Redbooks
ibm.com/redbooks

Building Cognitive Applications with IBM Watson Services: Volume 3 Visual Recognition

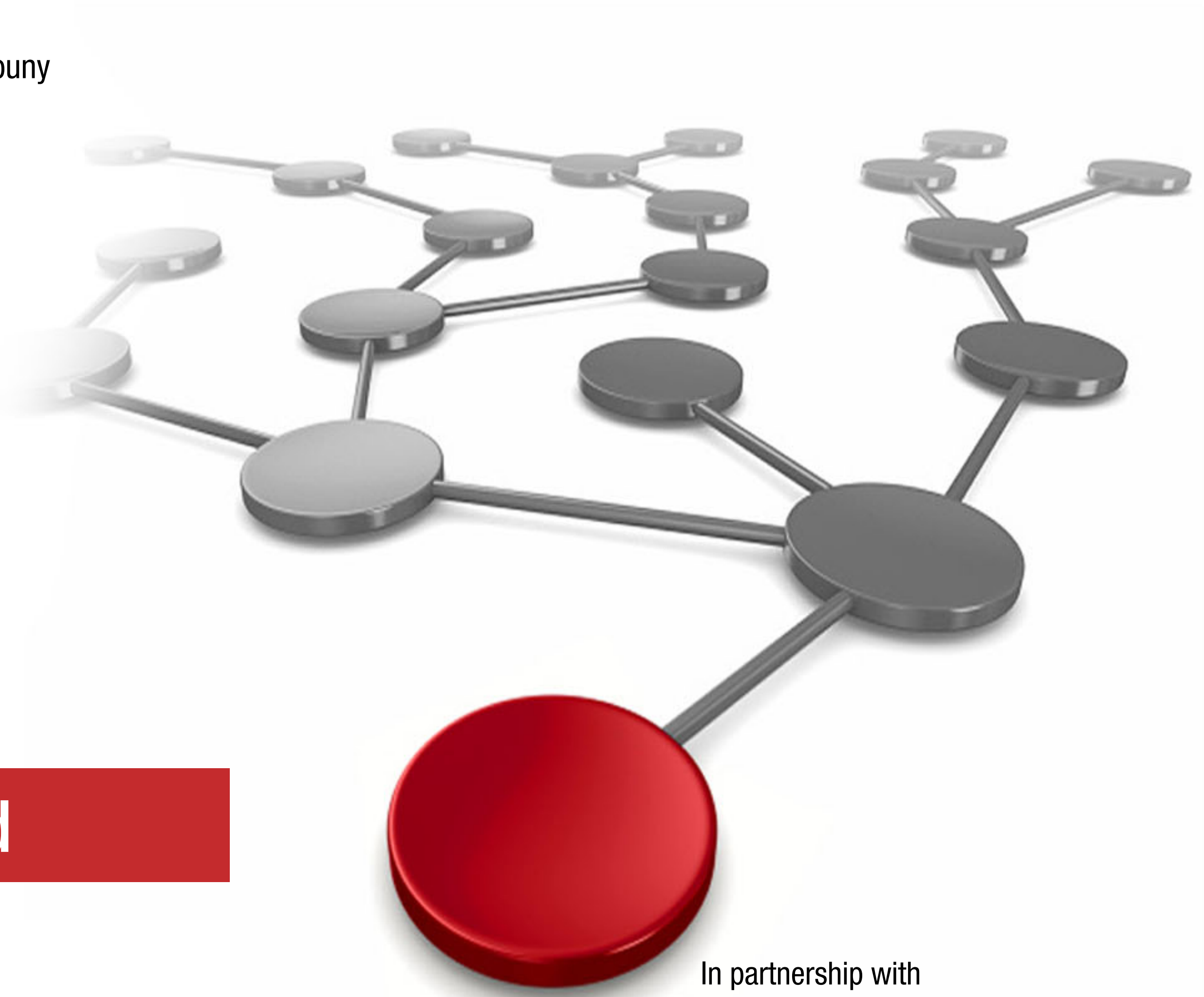
Dr. Azeddine Elhassouny

Dr. Le Nhan Tam

Dina Sayed

Bjoern Steffens

Lak Sri



Cloud

In partnership with
IBM **Skills Academy Program**

IBM

Redbooks

International Technical Support Organization

Building Cognitive Applications with IBM Watson Services: Volume 3 Visual Recognition

May 2017

SG24-8393-00

Note: Before using this information and the product it supports, read the information in “Notices” on page v.

First Edition (May 2017)

This edition applies to IBM Watson services in IBM Bluemix.

Contents

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

Bluemix®
developerWorks®
IBM®
IBM Watson®
IBM Watson IoT™
Redbooks®
Redbooks (logo) ®
Redpapers™
Tivoli®
Watson IoT™

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Preface

The *Building Cognitive Applications with IBM Watson Services* series is a seven-volume collection that introduces IBM® Watson cognitive computing services. The series includes an overview of specific Watson services with their associated architectures and simple code examples. Each volume describes how you can use and implement these services in your applications through practical use cases.

The series includes the following volumes:

- *Volume 1 Getting Started*, SG24-8387
- *Volume 2 Conversation*, SG24-8394
- *Volume 3 Visual Recognition*, SG24-8393
- *Volume 4 Natural Language Classifier*, SG24-8391
- *Volume 5 Language Translator*, SG24-8392
- *Volume 6 Speech to Text and Text to Speech*, SG24-8388
- *Volume 7 Natural Language Understanding*, SG24-8398

Whether you are a beginner or an experienced developer, this collection provides the information you need to start your research on Watson services. If your goal is to become more familiar with Watson in relation to your current environment, or if you are evaluating cognitive computing, this collection can serve as a powerful learning tool.

This IBM Redbooks® publication, Volume 3, introduces the IBM Watson® Visual Recognition service. The Watson Visual Recognition service uses deep learning algorithms to analyze images for scenes, objects, faces, and other content. This book introduces concepts that you need to understand in order to use this Watson service and provides simple code examples to illustrate the use of the APIs. This book includes examples of applications that demonstrate how to use the Watson Visual Recognition service in practical use cases. You can develop and deploy the sample applications by following along in a step-by-step approach and using provided code snippets. Alternatively, you can download an existing Git project to more quickly deploy the application.

Authors

This book was produced by a team of specialists from around the world working in collaboration with the IBM International Technical Support Organization.

Dr. Azeddine Elhassouny is a Professor at ENSIAS, IT College, Mohammed V University in Rabat, Morocco. He teaches multimedia indexing and retrieval, data visualization, and data science. His current research interests include deep learning in computer vision, multimedia signal processing, and pattern recognition and classification. His research also explores the connections between these areas and mathematical fields, such as neutrosophic field theory, fusion theory, and multiple criteria decision making (MCDM). He has published a book and several research papers about cognitive computing. Dr. Elhassouny holds a Ph.D. and M.S. in mathematics, computer science, and applications. He is a certified IT Specialist in IBM Big Data.

Dr. Le Nhan Tam is a Cloud Advisor in the IBM Cloud Advisor worldwide team and IBM Vietnam. His areas of expertise include cloud computing, Internet of Things (IoT), software engineering, business analytics, and cognitive computing. He has over 15 years of experience in the IT industry. Dr. Tam also is a visiting lecturer and a project advisor for students in universities in Hanoi. He supervised students in several projects to design and build cognitive applications that use IBM Bluemix® and IBM Watson Developer Cloud APIs, such as visual recognition and personal insight. Tam holds a Master degree in Computer Science from Asian Institute of Technology (AIT), Bangkok, Thailand, and a Ph.D degree in Computer Science from University of Rennes 1 and French National Institute for Research in Computer Science and Control (INRIA), France.

Dina Sayed is a Certified Expert IT Specialist in Watson Group, IBM Egypt. She joined IBM in 2004 as a software developer evolving to a technical lead role by leading the design and development of end-to-end solutions. She currently works in leading and supporting Watson multilingual cognitive solutions and services in the Middle East and North Africa (MENA) region. Her responsibilities include implementing proof of concepts, customer demonstrations, requirements analysis, solutions design, and development. She led Arabic support in IBM Social Media Analytic (SMA), which includes capabilities such as sentiment analysis, and demographic and behavioral analysis for telecommunication and government sectors. Dina also led implementation of the Watson conversation mobile app for different government entities in United Arab Emirates. Dina holds B.S. in Communication and Electronics Engineering from Cairo University and M.Sc. degree in Natural Language Processing from the University of Nottingham, UK Campus.

Bjoern Steffens is a certified IT Specialist in software services IBM Hybrid Cloud, IBM Switzerland Ltd. Bjoern has over 25 years of experience in the IT consulting field where his focus has been on data architectures, analytics, data science, and deriving insights, leveraging business intelligence and business analytics tools. Bjoern enables his customers to drive business decisions and strategy based on advanced statistics in combination with carefully designed dashboards and reports adding precision to information. He is also an expert in providing automated self-service information systems for audiences with varying technical and business requirements. Bjoern recently began his journey of applying cognitive tools and processes to his existing knowledge going forward as a Bluemix Developer.

Lak Sri currently serves as Program Director in IBM developerWorks® part of the IBM Digital Business Group organization. Lak leads innovation in the developer activation space. He was the Technical Leader for the Building Cognitive Applications with IBM Watson Services Redbooks series. Lak led the development of the IBM Cloud Application Developer Certification program and the associated course. Earlier he worked as Solution Architect for Enterprise Solutions in Fortune 500 companies using IBM Tivoli® products. He also built strategic partnerships in education and IBM Watson IoT™. Lak is an advocate and a mentor in several technology areas, and he volunteers to plan and support local community programs.

The project that produced this publication was managed by **Marcela Adan**, IBM Redbooks Project Leader, ITSO.

Thanks to the following people for their contributions to this project:

Swin Voon Cheok
Ecosystem Development (EcoD) Strategic Initiative, IBM Systems

Juan Pablo Napoli
Skills Academy Worldwide Leader, Global University Programs

Teja Tummalapalli
IBM Digital Business Group

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Basics of Watson Visual Recognition service

This chapter gets you started with using the Watson Visual Recognition service.

The Watson Visual Recognition service uses deep learning algorithms to analyze images for scenes, objects, faces, and other content. The response includes keywords that provide information about the content. A set of built-in classes provides highly accurate results without training. You can train custom classifiers to create specialized classes.

This chapter introduces the two main tasks that the IBM Watson Visual Recognition service performs:

- Classify a picture and get image details. For example, you might have an image of any entity, such as a cat, and use the Watson Visual Recognition `classify` method to get the details for that image. For more information, see the Classify an image topic in Watson Developer Cloud.
- Detect faces, gender, and age in a picture by using the Watson Visual Recognition `detectFaces` method. For more information, see the Detect faces topic in Watson Developer Cloud.

This chapter provides simple code examples in Java and Node.js that use the Watson SDKs and Eclipse IDE and Node.js Express framework.

The following topics are covered in this chapter:

- Use case examples
- Creating a Watson Visual Recognition service instance and getting the API key
- Image classification and face detection examples
- Classifying images and detecting faces: Use Watson Java SDK and Eclipse IDE
- Classifying images and detecting faces: Use Watson Node.js SDK and Node.js Express framework
- References

1.1 Use case examples

IBM Watson Visual Recognition is a service that allows users to understand the content of images and classify images into logical categories. In addition to classifying images, Visual Recognition also offers facial detection.

The Visual Recognition service can be used for diverse applications and industries, such as these:

- **Manufacturing**: Use images from a manufacturing setting to make sure products are being positioned correctly on an assembly line.
- **Visual Auditing**: Look for visual compliance or deterioration in a fleet of trucks, planes, or windmills in the field, train custom classifiers to understand what defects look like.
- **Insurance**: Rapidly process claims by using images to classify claims into different categories.
- **Social listening**: Use images from your product line or your logo to track buzz about your company on social media.
- **Social commerce**: Use an image of a plated dish to find out which restaurant serves it and find reviews, use a travel photo to find vacation suggestions based on similar experiences, use a house image to find similar homes that are for sale.
- **Retail**: Take a photo of a favorite outfit to find stores with those clothes in stock or on sale, use a travel image to find retail suggestions in that area, use the photo of an item to find out its price in different stores.
- **Education**: Create image-based applications to educate about taxonomies, use pictures to find educational material on similar subjects.
- **Public safety**: Automated, real-time video stream analysis to include targeted observations such as facial recognition and automated licence-plate reading, identify a suspect’s car with unknown whereabouts to locate instances of that model, parked or in motion, in any surveilled part of the country.

1.2 Creating a Watson Visual Recognition service instance and getting the API key

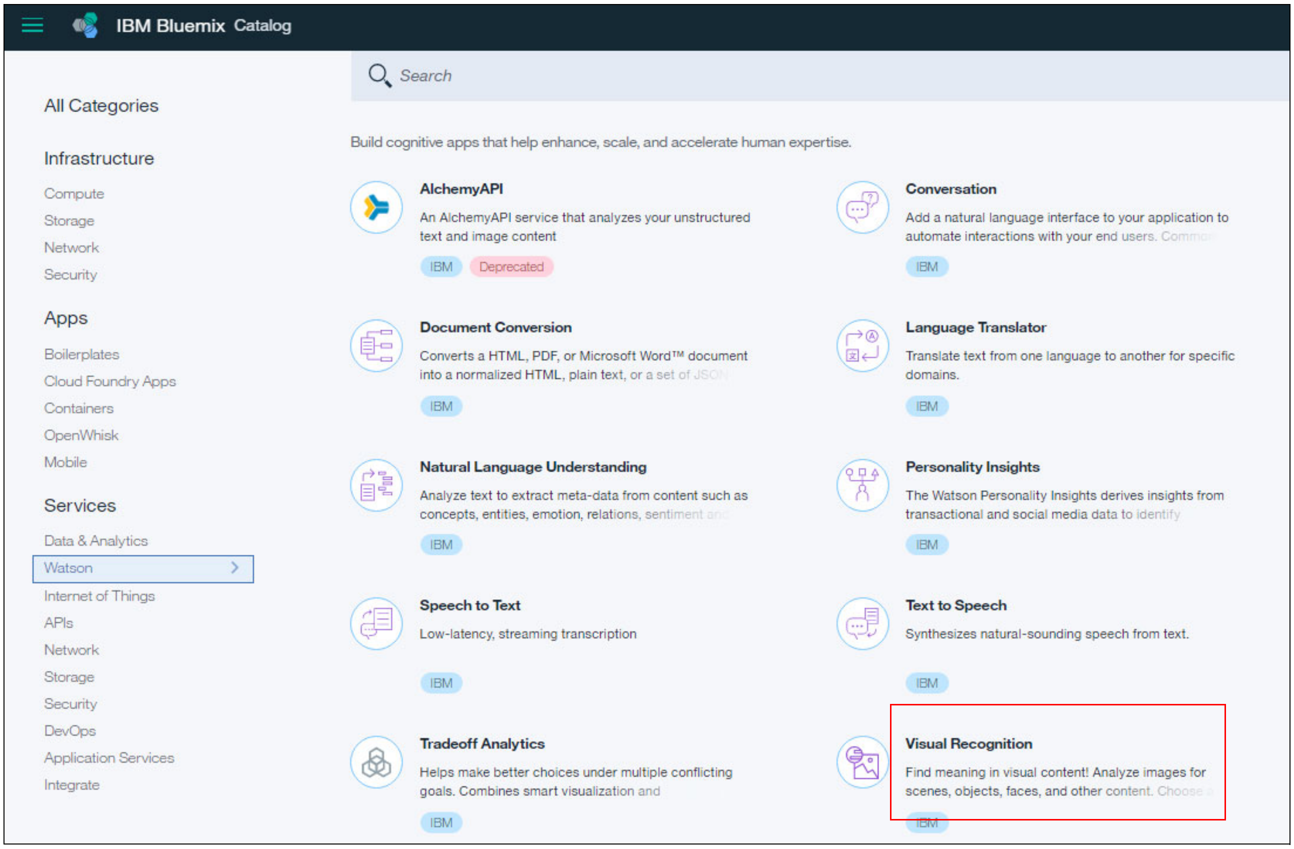
Bluemix provides resources to your applications through a service instance. Before you can use the Watson APIs, you must create an instance of the corresponding service; you will need to create a Watson Visual Recognition service instance for use in all the examples in this book.

To create an instance of the Visual Recognition service, complete these steps:

1. Create a Bluemix account.

   You must have a Bluemix account to access the Watson APIs. You can create a trial Bluemix account, valid for a specified number days.

2. Log in to Bluemix and click **Catalog**.
3. From the left menu, select **Services** → **Watson**.
4. Click **Visual Recognition** (Figure 1-1 on page 3).



Build cognitive apps that help enhance, scale, and accelerate human expertise.

AlchemyAPI
An AlchemyAPI service that analyzes your unstructured text and image content
IBM Deprecated

Conversation
Add a natural language interface to your application to automate interactions with your end users. Comm
IBM

Document Conversion
Converts a HTML, PDF, or Microsoft Word™ document into a normalized HTML, plain text, or a set of JSON
IBM

Language Translator
Translate text from one language to another for specific domains.
IBM

Natural Language Understanding
Analyze text to extract meta-data from content such as concepts, entities, emotion, relations, sentiment and
IBM

Personality Insights
The Watson Personality Insights derives insights from transactional and social media data to identify
IBM

Speech to Text
Low-latency, streaming transcription
IBM

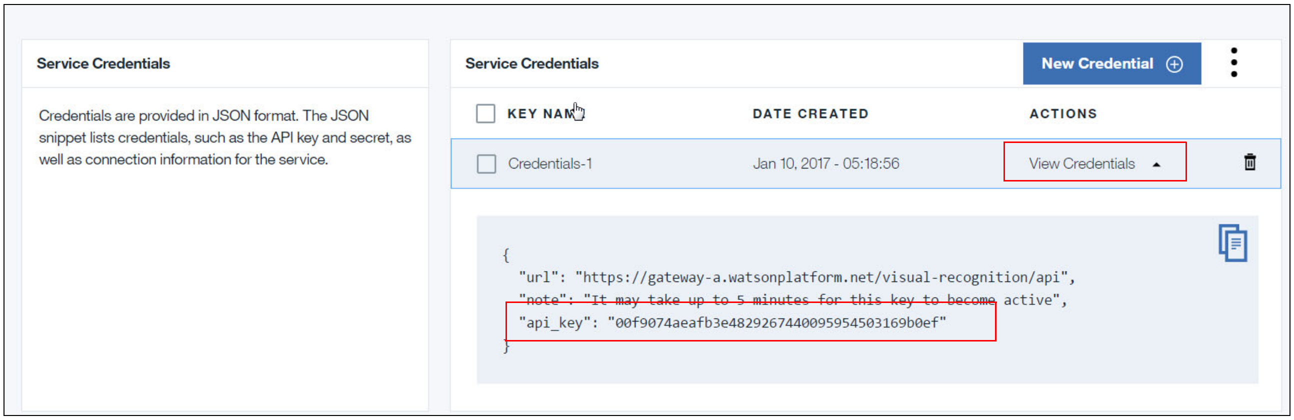
Text to Speech
Synthesizes natural-sounding speech from text.
IBM

Tradeoff Analytics
Helps make better choices under multiple conflicting goals. Combines smart visualization and
IBM

Visual Recognition
Find meaning in visual content! Analyze images for scenes, objects, faces, and other content. Choose
IBM

Figure 1-1 Create Visual Recognition service instance

5. Change the service and credential names or accept the default values. Confirm that the pricing plan **Free** is selected and click **Create**.
6. Select the **Service credentials** tab and click **View Credentials** (Figure 1-2).
7. Copy the API key for later use.


Service Credentials

Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service.

Service Credentials

□ KEY NAME	DATE CREATED	ACTIONS
□ Credentials-1	Jan 10, 2017 - 05:18:56	View Credentials ▲

```
{
  "url": "https://gateway-a.watsonplatform.net/visual-recognition/api",
  "note": "It may take up to 5 minutes for this key to become active",
  "api_key": "00f9074aeafb3e4829267440095954503169b0ef"
}
```

Figure 1-2 View Credential

1.3 Image classification and face detection examples

The examples in this chapter do the following tasks by using the Watson Visual Recognition service:

- Classify an image using the pre-trained classifier for general classification

  For each image, the response, in JSON format, describes the image content.

- Detect faces in an image

  Detect faces in image, analyze the detected faces, and get data about them, such as estimated age, gender. If a celebrity's face is detected, provide the names of celebrities. Images must be in JPEG or PNG file format.

For more information, see the Visual Recognition getting started tutorials.

1.3.1 Expected results

By following the examples in this chapter, you should be able to submit images to the application and obtain results after the image has been analyzed by the Watson Visual Recognition services.

Image classification results

Figure 1-3 represents the image used as input to the classification.

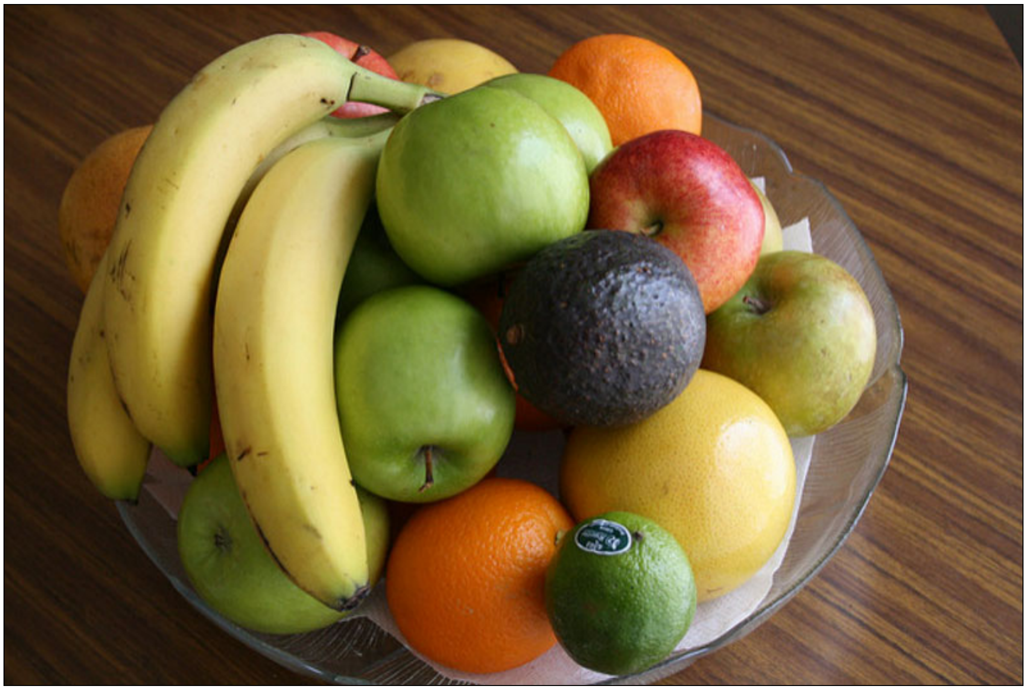
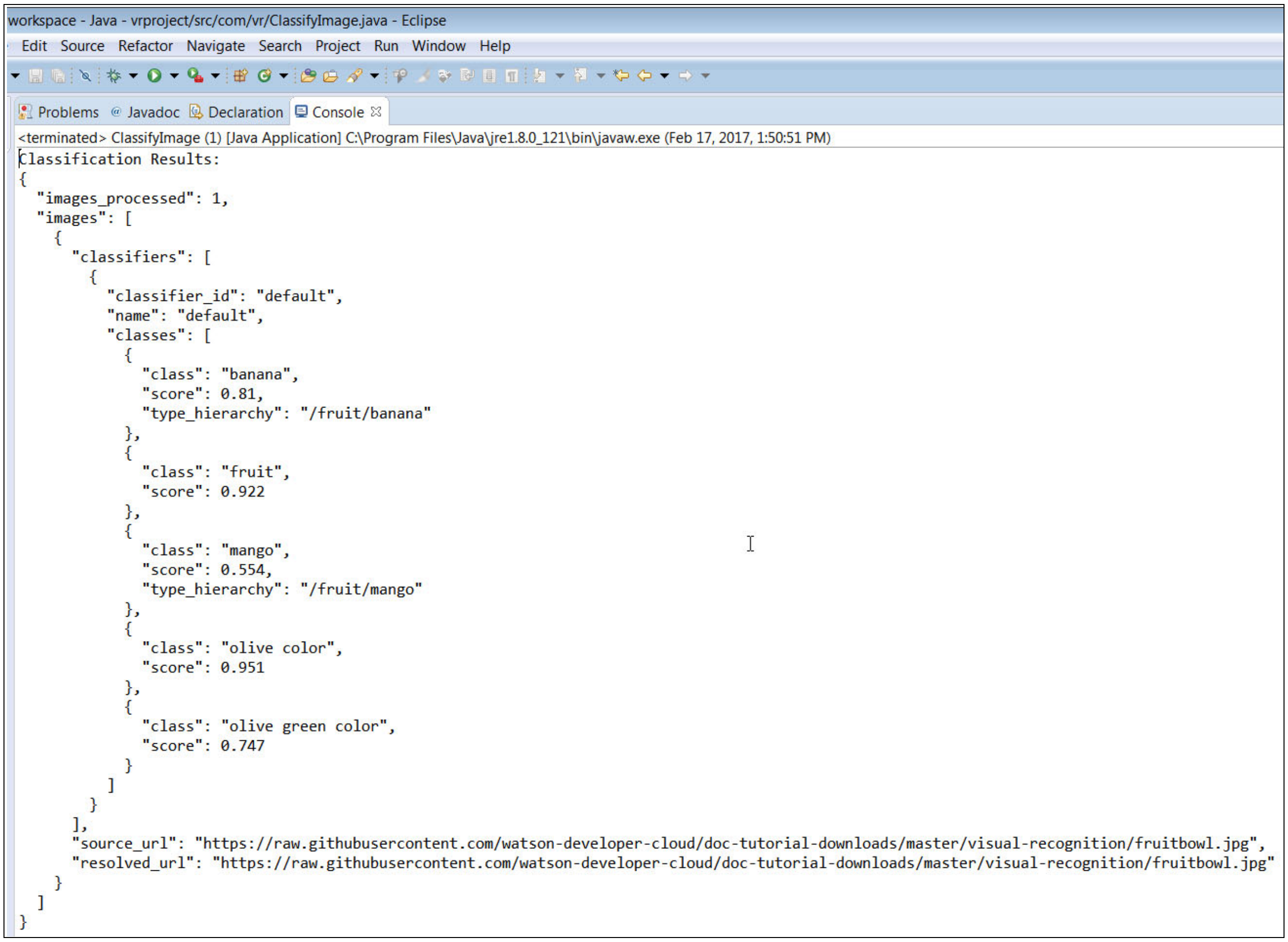


Figure 1-3 A sample for image classification: Fruit dish image

Figure 1-4 on page 5 shows the response, in JSON format. It describes the image content and for each image, the response includes a score for each class.


workspace - Java - vrproject/src/com/vr/ClassifyImage.java - Eclipse
Edit Source Refactor Navigate Search Project Run Window Help
Problems
Javadoc
Declaration
Console

```
<terminated> ClassifyImage (1) [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Feb 17, 2017, 1:50:51 PM)
Classification Results:
{
  "images_processed": 1,
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "default",
          "name": "default",
          "classes": [
            {
              "class": "banana",
              "score": 0.81,
              "type_hierarchy": "/fruit/banana"
            },
            {
              "class": "fruit",
              "score": 0.922
            },
            {
              "class": "mango",
              "score": 0.554,
              "type_hierarchy": "/fruit/mango"
            },
            {
              "class": "olive color",
              "score": 0.951
            },
            {
              "class": "olive green color",
              "score": 0.747
            }
          ]
        }
      ],
      "source_url": "https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/fruitbowl.jpg",
      "resolved_url": "https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/fruitbowl.jpg"
    }
  ]
}
```

Figure 1-4 Image classification results

Face detection results

Figure 1-5 represents the image used as input in face detection.

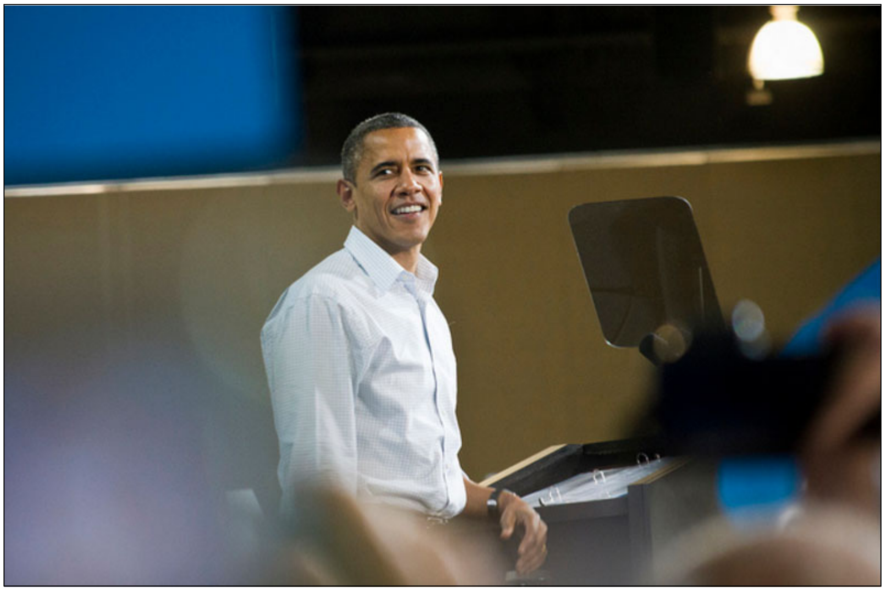
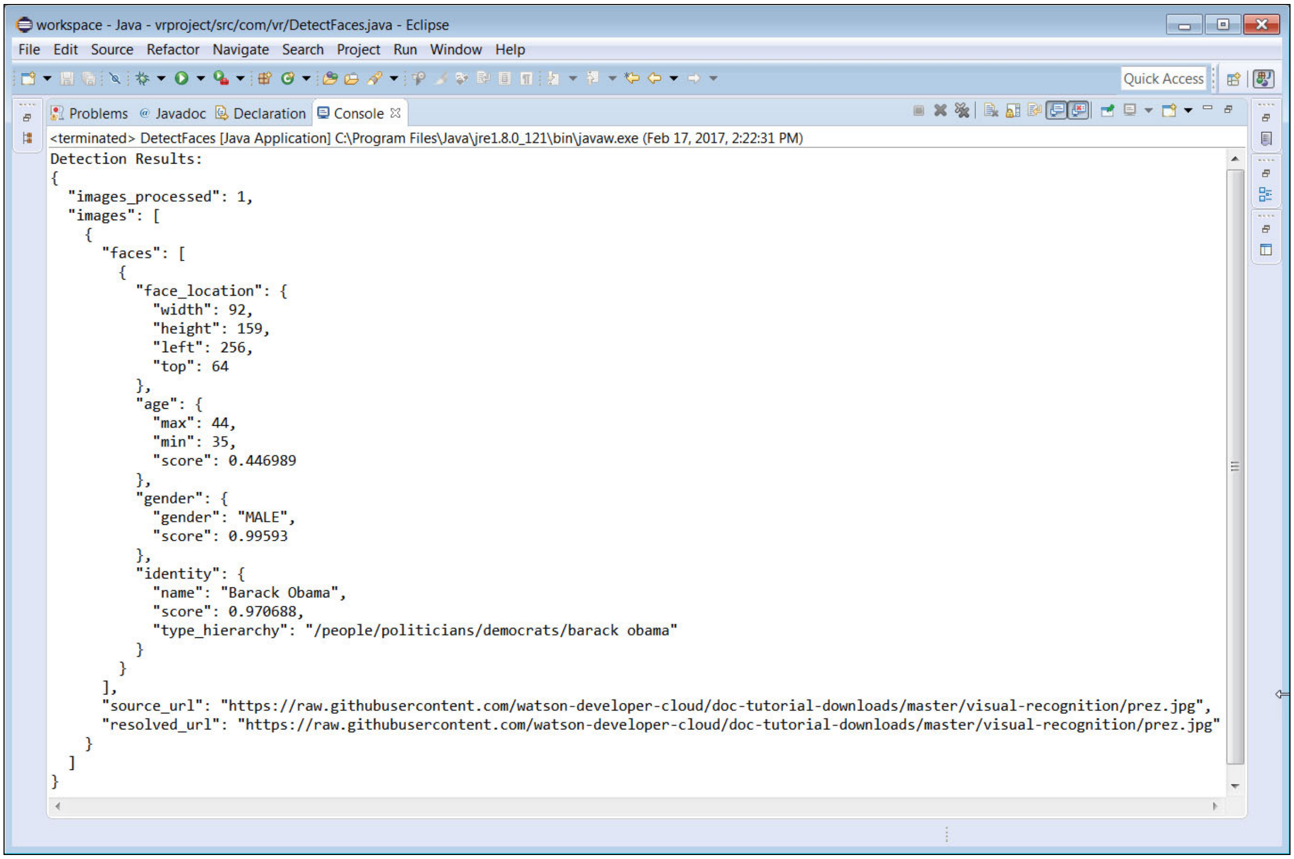


Figure 1-5 Sample image for face detection: Barak Obama

Figure 1-6 shows the response in JSON format; it shows that a face was detected and recognized it as an image of a celebrity, former President Barak Obama. It also detected gender as Male and the estimated age.


workspace - Java - vrproject/src/com/vr/DetectFaces.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Problems
Javadoc
Declaration
Console

```
<terminated> DetectFaces [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Feb 17, 2017, 2:22:31 PM)
Detection Results:
{
  "images_processed": 1,
  "images": [
    {
      "faces": [
        {
          "face_location": {
            "width": 92,
            "height": 159,
            "left": 256,
            "top": 64
          },
          "age": {
            "max": 44,
            "min": 35,
            "score": 0.446989
          },
          "gender": {
            "gender": "MALE",
            "score": 0.99593
          },
          "identity": {
            "name": "Barack Obama",
            "score": 0.970688,
            "type_hierarchy": "/people/politicians/democrats/barack obama"
          }
        }
      ],
      "source_url": "https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/prez.jpg",
      "resolved_url": "https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/prez.jpg"
    }
  ]
}
```

Figure 1-6 Results of running the face detection service

1.4 Classifying images and detecting faces: Use Watson Java SDK and Eclipse IDE

By the end of this section, you should be able to accomplish these objectives:

- Use the Watson Java SDK to call Watson APIs for image classification.
- Use the Watson Java SDK to call Watson APIs to detect faces and get additional data about them such as gender and estimated age.

Implementing this use case using the Watson Java SDK and Eclipse IDE involves the following steps:

1. Creating a Bluemix account (see step 1 on page 2)
2. Creating a Watson Visual Recognition service instance and getting the API key
3. Getting started with Eclipse and Java
4. Downloading the Watson Java SDK
5. Classifying images
6. Detecting faces

1.4.1 Getting started with Eclipse and Java

In this use case, Eclipse IDE is used to build the Java application. Install and become familiar with Eclipse and Java before you follow the implementation steps:

- Download Eclipse:

  `https://eclipse.org/downloads/`

- Getting Started with Eclipse:

  `https://eclipse.org/users/`

- Getting Started with Java Programming:

  `http://www.oracle.com/technetwork/topics/newtojava/learn-141096.html`

1.4.2 Downloading the Watson Java SDK

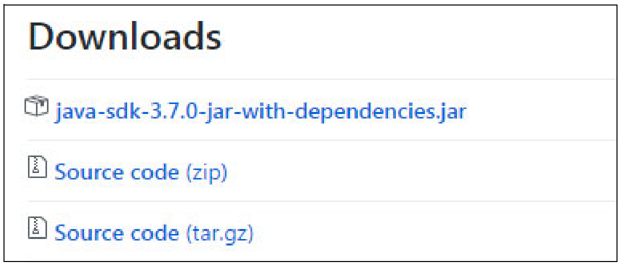
IBM Watson services offer Software Development Kits (SDKs) that simplify application development for a variety of programming languages and platforms.

In this chapter, the focus is on developing a Java sample application. Therefore, the Watson Java SDK must be downloaded:

1. Go to GitHub:

   `https://github.com/watson-developer-cloud/java-sdk/releases`

2. Scroll to the **Downloads** section and click **java-sdk-3.7.0-jar-with-dependencies.jar** (Figure 1-7).



Downloads

java-sdk-3.7.0-jar-with-dependencies.jar

Source code (zip)

Source code (tar.gz)

Figure 1-7 Download Watson Java SDK

1.4.3 Classifying images

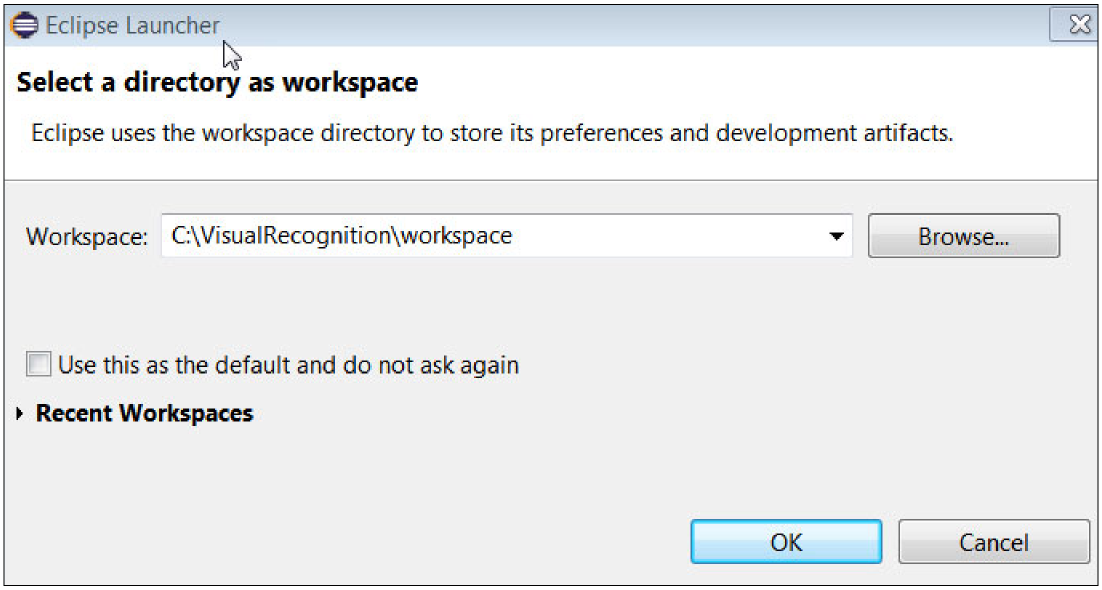
In this section, you will use the Watson Java SDK to classify image content. It describes how to call the Watson service and how to interpret the response.

Complete these steps:

1. Launch Eclipse.

   After you complete 1.4.1, “Getting started with Eclipse and Java” on page 7, you should have Eclipse installed in your workstation. Launch Eclipse by double-clicking the application icon.

2. Select a workspace directory and click **OK** (Figure 1-8 on page 8).



Select a directory as workspace

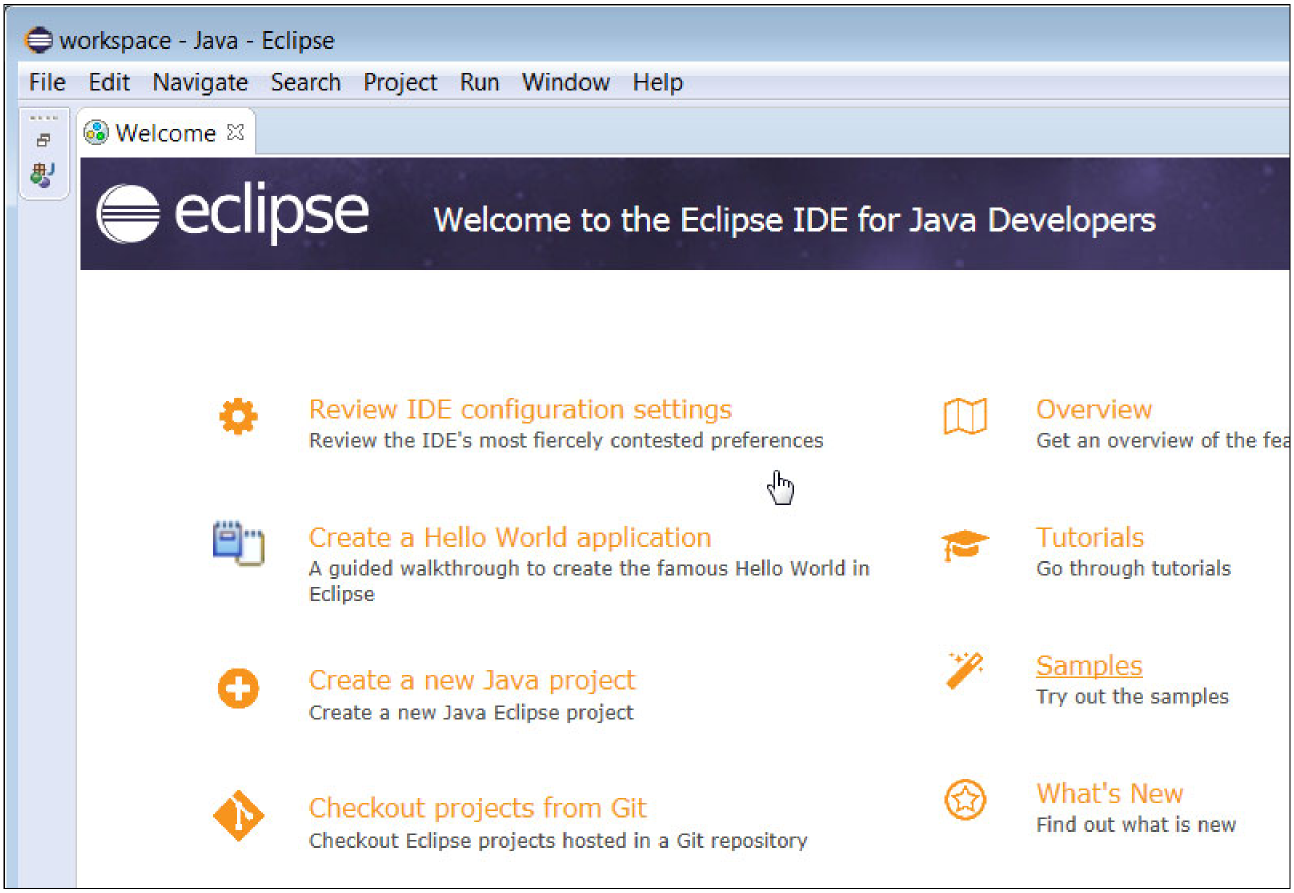
Eclipse uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\VisualRecognition\workspace

□ Use this as the default and do not ask again

Figure 1-8 Select an Eclipse workspace

The Eclipse Welcome page opens (Figure 1-9).



Welcome to the Eclipse IDE for Java Developers

Review IDE configuration settings
Review the IDE's most fiercely contested preferences

Create a Hello World application
A guided walkthrough to create the famous Hello World in Eclipse

Create a new Java project
Create a new Java Eclipse project

Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository

Overview
Get an overview of the fea

Tutorials
Go through tutorials

Samples
Try out the samples

What's New
Find out what is new

Figure 1-9 Eclipse Welcome page

3. Create a new Java project. Select **File** → **New** → **Java Project** (Figure 1-10).

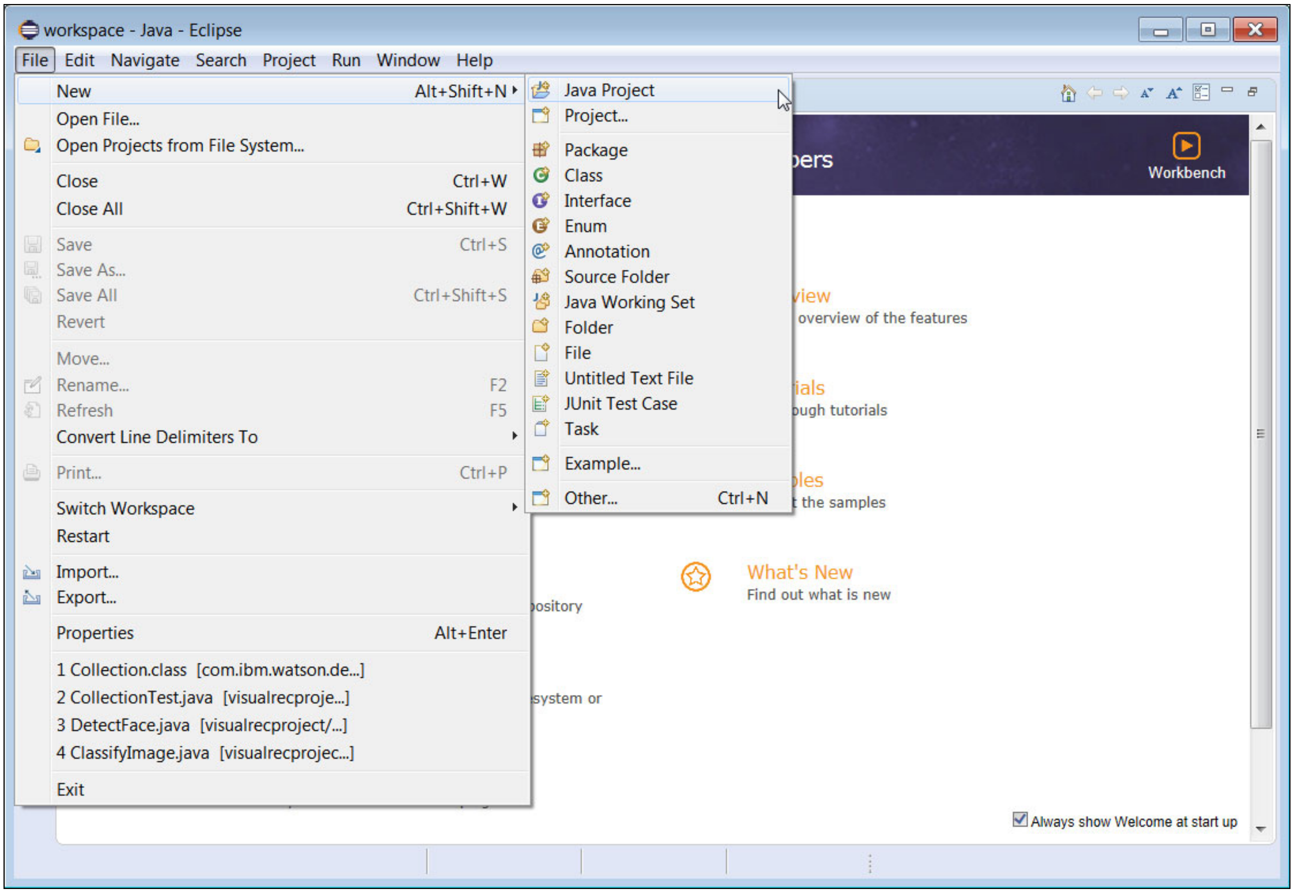
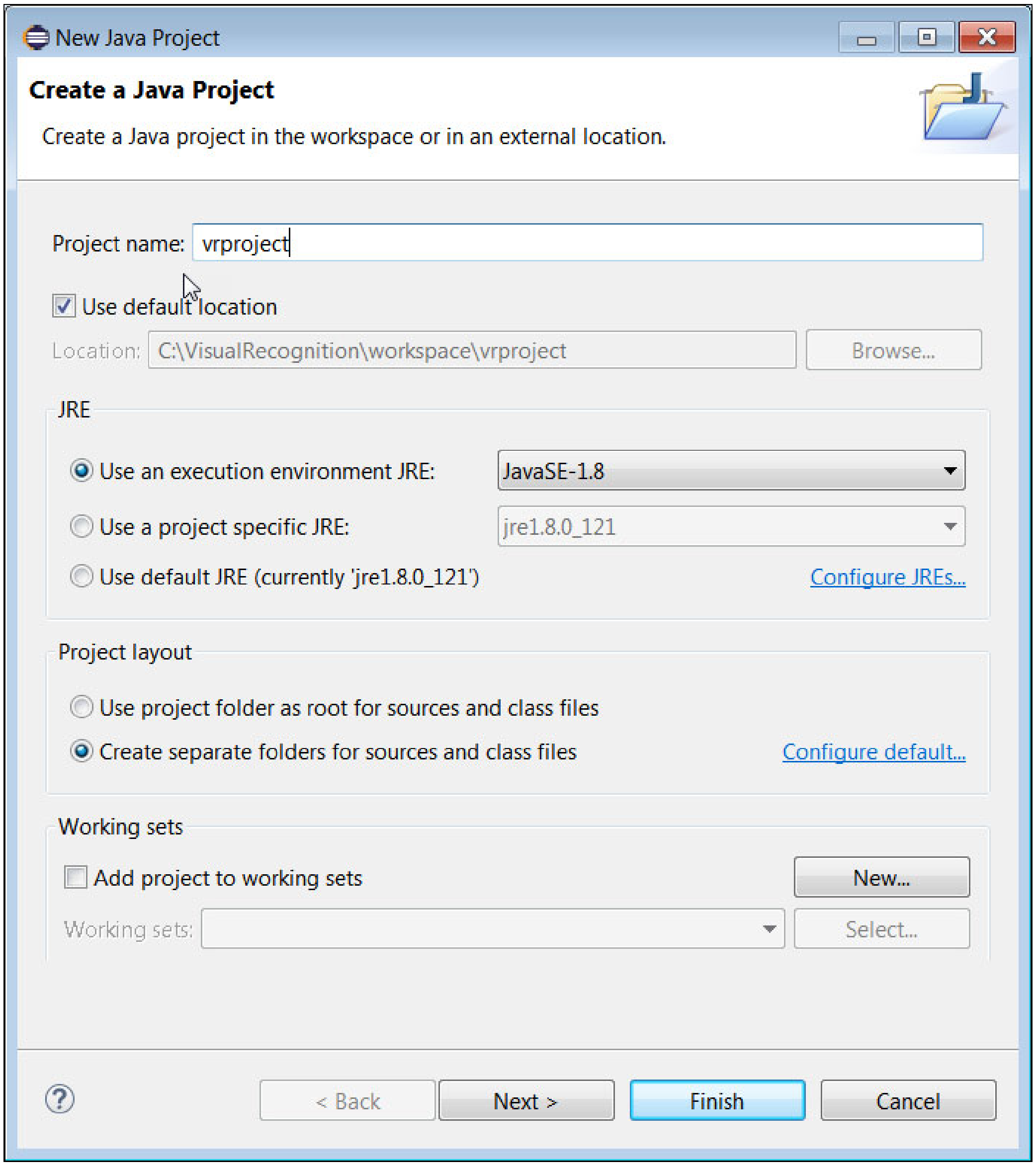


Figure 1-10 Create new Java project

4. Enter the project name (`vrproject` in this example), accept the default values for other fields, and click **Finish** (Figure 1-11).



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: vrproject

☑ Use default location

Location: C:\VisualRecognition\workspace\vrproject

JRE

● Use an execution environment JRE: JavaSE-1.8

○ Use a project specific JRE: jre1.8.0_121

○ Use default JRE (currently 'jre1.8.0_121')

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

Working sets

☐ Add project to working sets

Working sets:

Figure 1-11 Setting your Java project name

5. Close the Welcome page in order to view your project in Package Explorer.

6. Import the Watson Java SDK so you can use it in your application. Right-click the **vrproject** project and select **Build Path** → **Configure Build Path** (Figure 1-12).

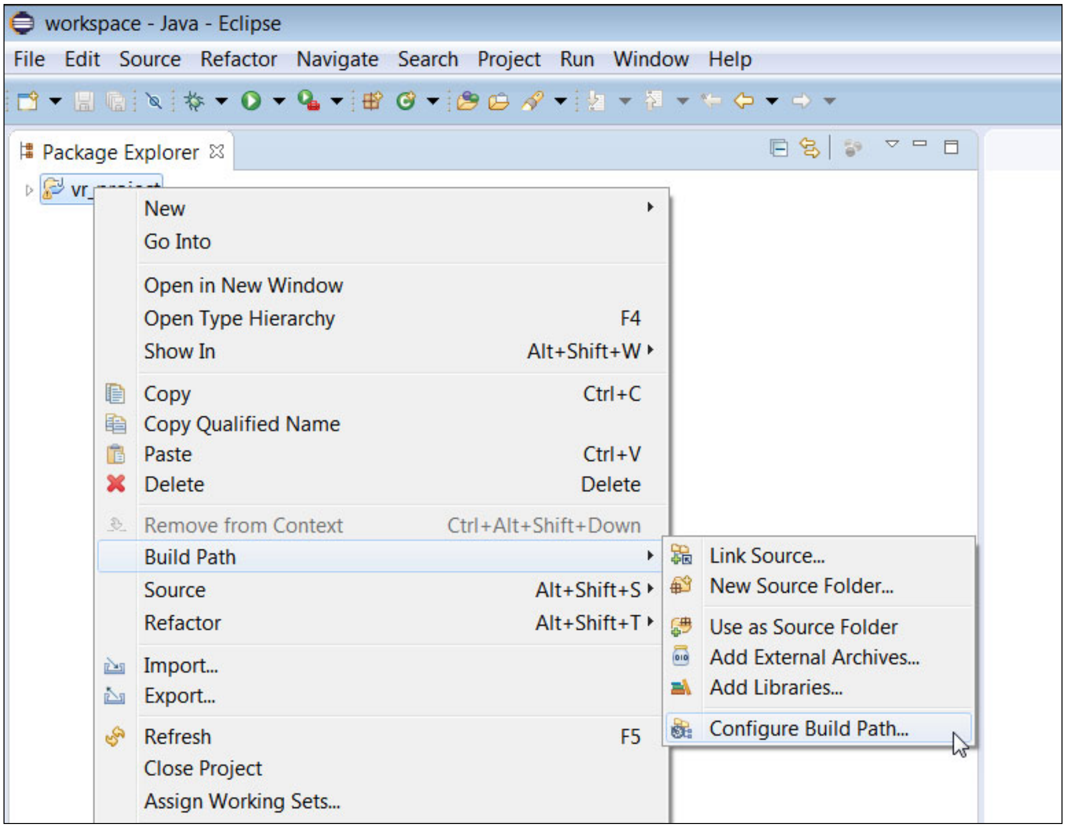
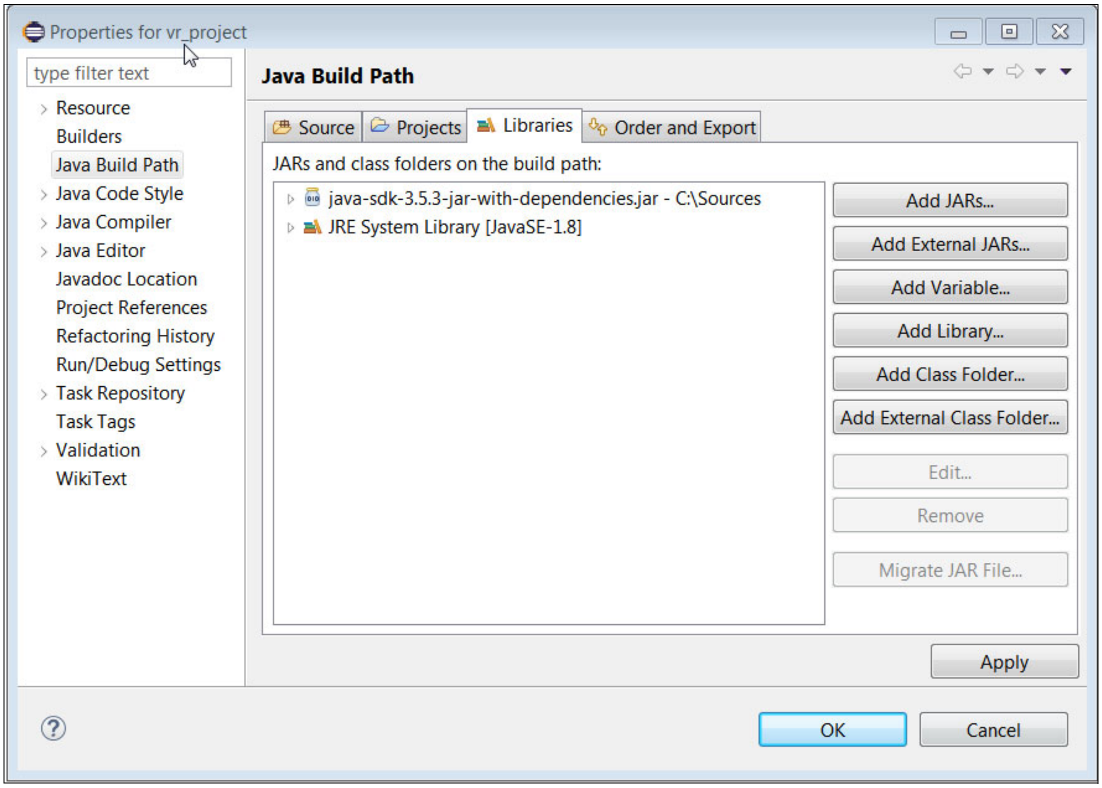


Figure 1-12 Go to your project build path

7. Select the **Libraries** tab and click **Add External JARs**.

   Browse to and select the Watson Java SDK (JAR file) that you downloaded in 1.4.2, “Downloading the Watson Java SDK” on page 7. Click **OK**.

   The Watson Java SDK is successfully added to your project (Figure 1-13).



JARs and class folders on the build path:

java-sdk-3.5.3-jar-with-dependencies.jar - C:\Sources

JRE System Library [JavaSE-1.8]

Figure 1-13 Import the Watson Java SDK to the Eclipse project

8. Create a Java class to classify your image. Right-click the **vrproject** project and select **New** → **Class** (Figure 1-14).

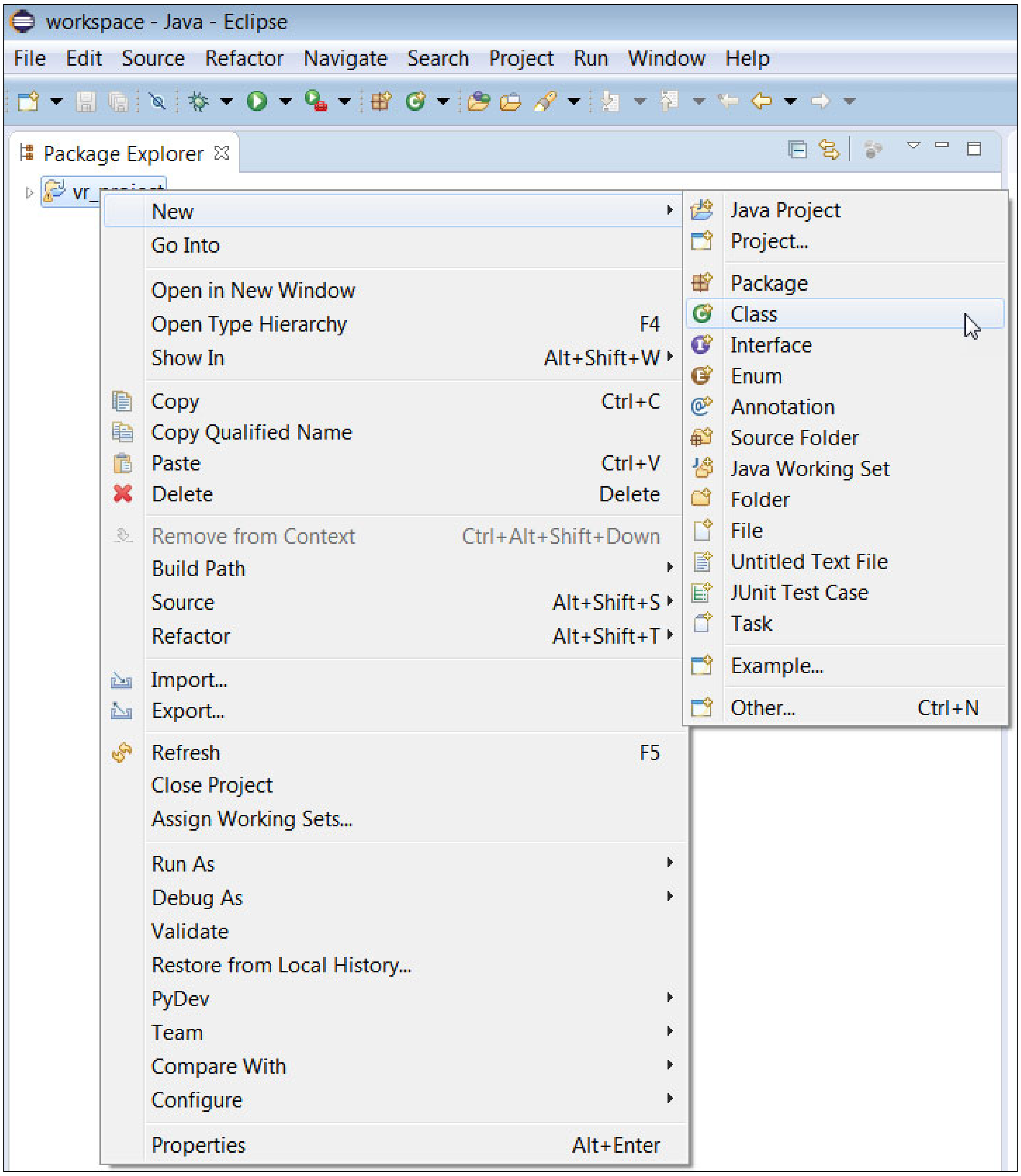
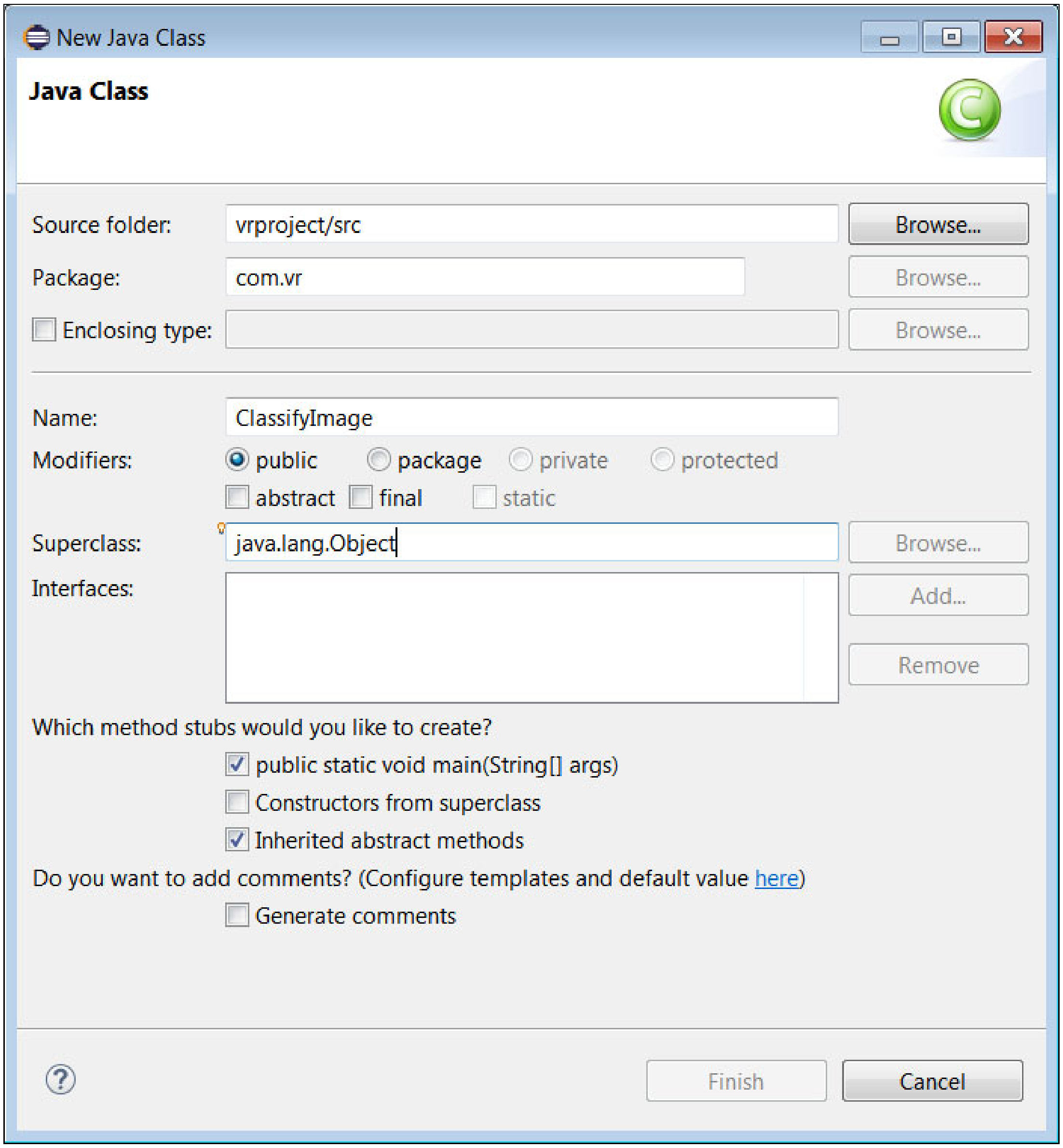


Figure 1-14 Create a new Java class

9. The New Java Class window opens (Figure 1-15). Provide the class details: Add a class name (`ClassifyImage` in this example) and select the **public static void main(String[] args)** check box. Click **Finish**.



New Java Class

Java Class

Source folder: vrproject/src

Package: com.vr

□ Enclosing type:

Name: ClassifyImage

Modifiers: ● public ○ package ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

□ Constructors from superclass

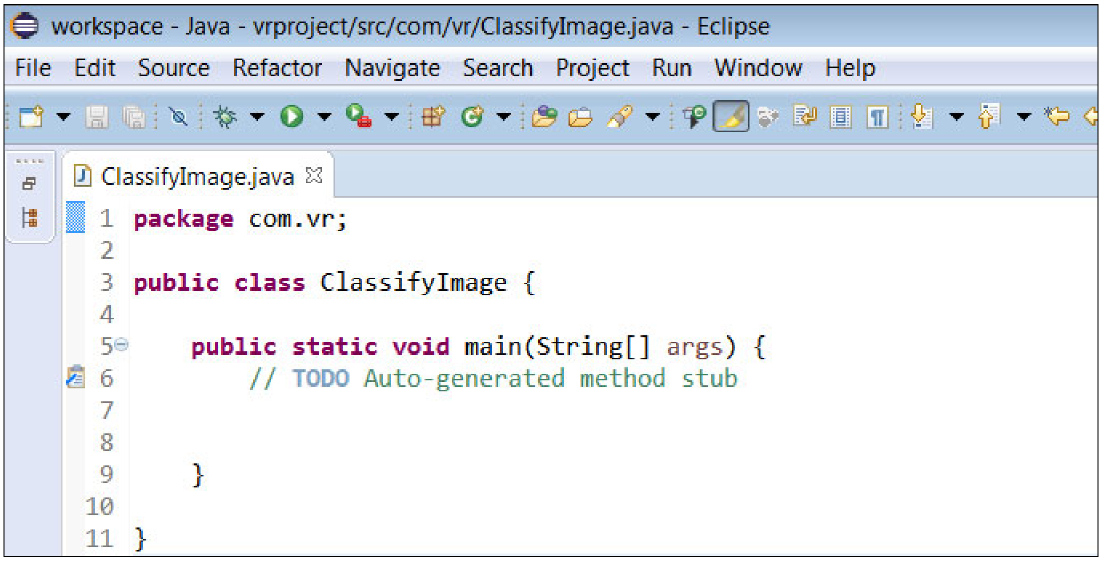
☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

□ Generate comments

Figure 1-15 Set your Java class name

The ClassifyImage class is created (Figure 1-16).



```
package com.vr;

public class ClassifyImage {

    public static void main(String[] args) {
        // TODO Auto-generated method stub


    }

}
```

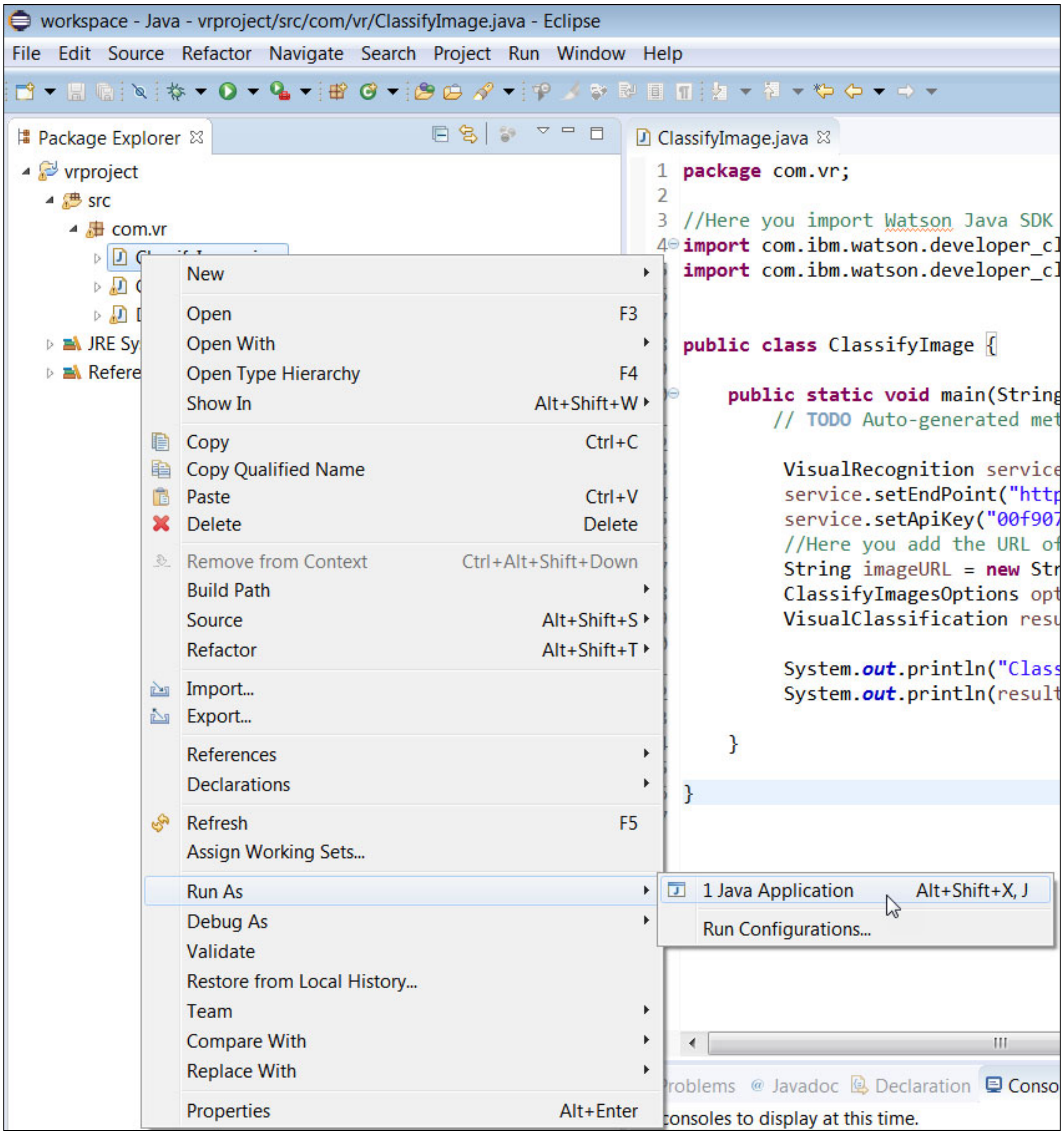
Figure 1-16   ClassifyImage Java class

10. Edit the `ClassifyImage.java` content by adding the code in Example 1-1. Spend several minutes to read through the code snippet to understand it.

Example 1-1   Code snippet for image classification

```
package com.vr;

//Here you import Watson Java SDK to make it available in your code.
import com.ibm.watson.developer_cloud.visual_recognition.v3.*;
import com.ibm.watson.developer_cloud.visual_recognition.v3.model.*;


public class ClassifyImage {

public static void main(String[] args) {

VisualRecognition service = new    VisualRecognition(VisualRecognition.VERSION_DATE_2016_05_20);
       service.setEndPoint("https://gateway-a.watsonplatform.net/visual-recognition/api");

//Here you replace "your_api_key_here" by the API Key you created in "Creating //a Watson Visual
Recognition service instance and getting the API key"
service.setApiKey("your_api_key_here");

//Here you add the URL of your image. The image size should not exceed 2MB.
 String imageURL = new
String("https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recog
nition/fruitbowl.jpg");
       ClassifyImagesOptions options = new   ClassifyImagesOptions.Builder().url(imageURL).build();
VisualClassification result = service.classify(options).execute();
System.out.println("Classification Results:");
System.out.println(result);

   }

}
```

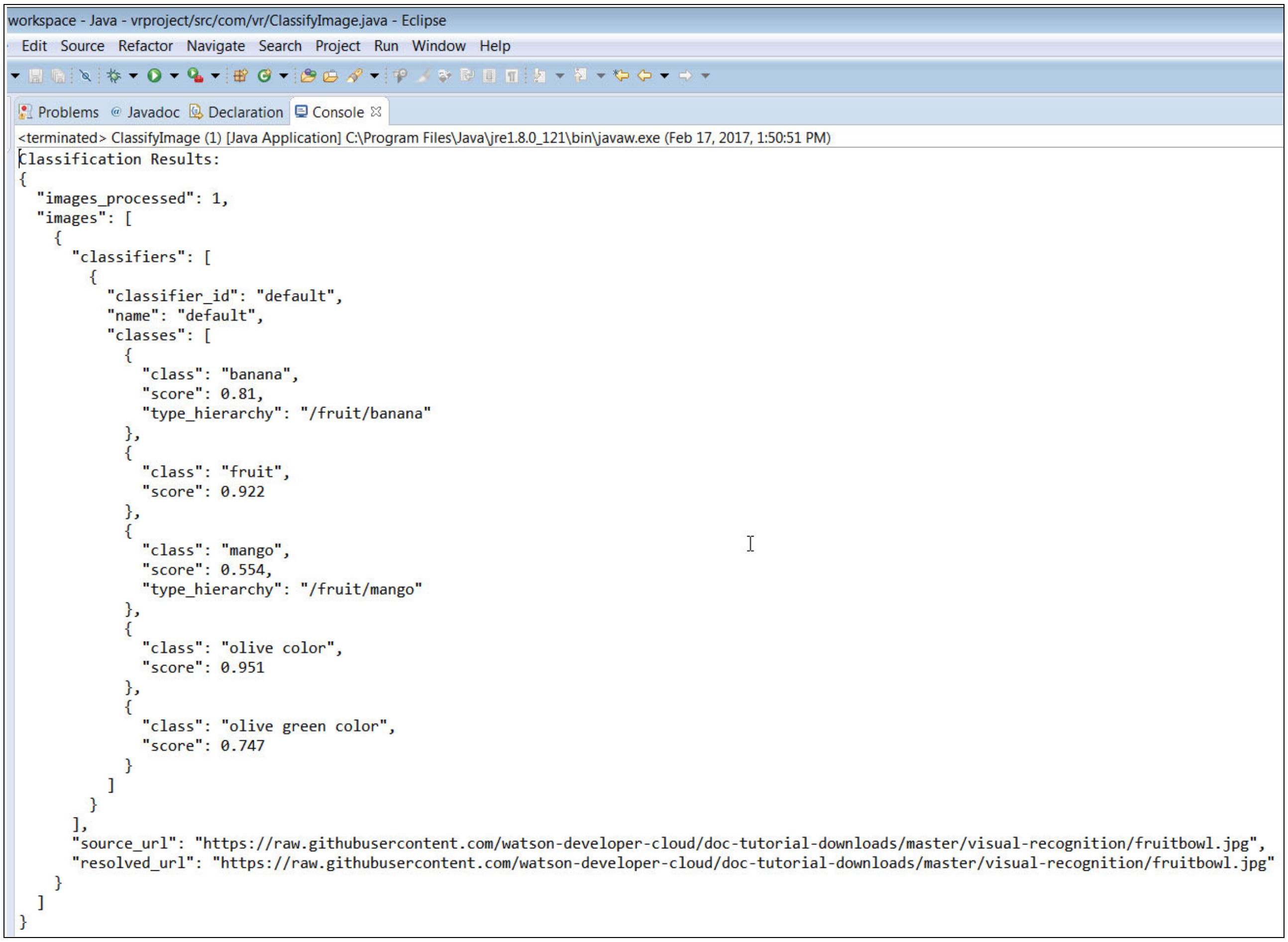
11. Run the code and check results. Right-click **ClassImage.java** and then select **Run As** → **Java Application** (Figure 1-17).



```
package com.vr;

//Here you import Watson Java SDK
import com.ibm.watson.developer_cl
import com.ibm.watson.developer_cl


public class ClassifyImage {

    public static void main(String
        // TODO Auto-generated met

         VisualRecognition service
         service.setEndPoint("http
         service.setApiKey("00f907
         //Here you add the URL of
         String imageURL = new Str
         ClassifyImagesOptions opt
         VisualClassification resu

         System.out.println("Class
         System.out.println(result

    }

}
```

Figure 1-17 Run ClassifyImage

12. View the results in the Console, which by default is under `ClassifyImage`. You can double-click the **Console** tab to maximize the view and check the results (Figure 1-18).



<terminated> ClassifyImage (1) [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Feb 17, 2017, 1:50:51 PM)

```
Classification Results:
{
  "images_processed": 1,
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "default",
          "name": "default",
          "classes": [
            {
              "class": "banana",
              "score": 0.81,
              "type_hierarchy": "/fruit/banana"
            },
            {
              "class": "fruit",
              "score": 0.922
            },
            {
              "class": "mango",
              "score": 0.554,
              "type_hierarchy": "/fruit/mango"
            },
            {
              "class": "olive color",
              "score": 0.951
            },
            {
              "class": "olive green color",
              "score": 0.747
            }
          ]
        }
      ],
      "source_url": "https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/fruitbowl.jpg",
      "resolved_url": "https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/fruitbowl.jpg"
    }
  ]
}
```

Figure 1-18 Console view displays classification results

The response, in JSON format, describes the image content. For each image, the response includes a score for each class.

Figure 1-19 represents the image used as input to the classification.

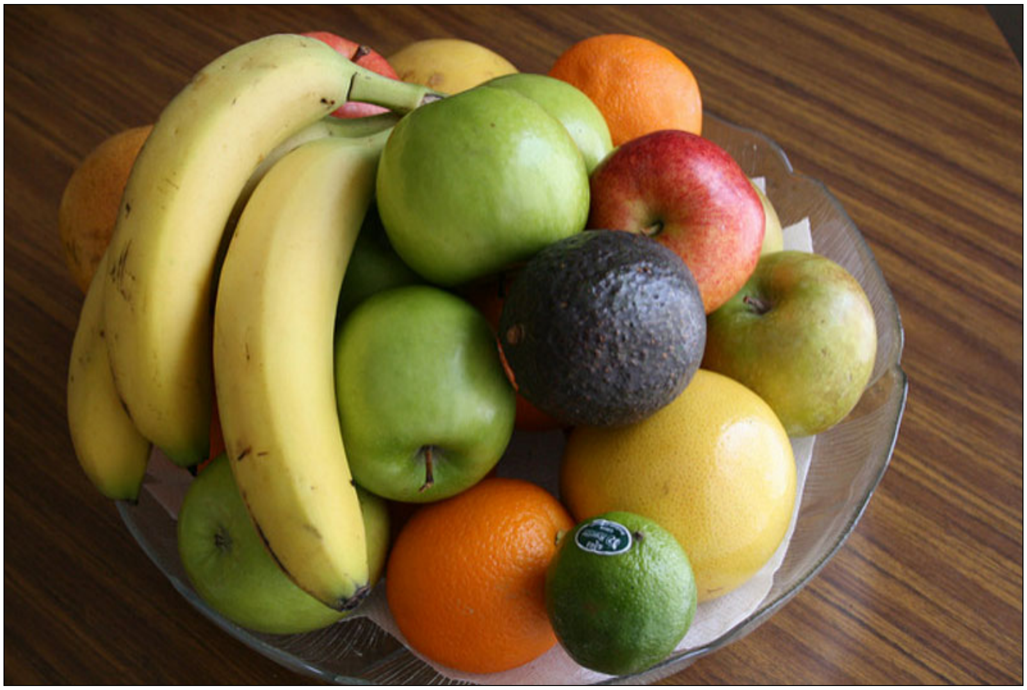
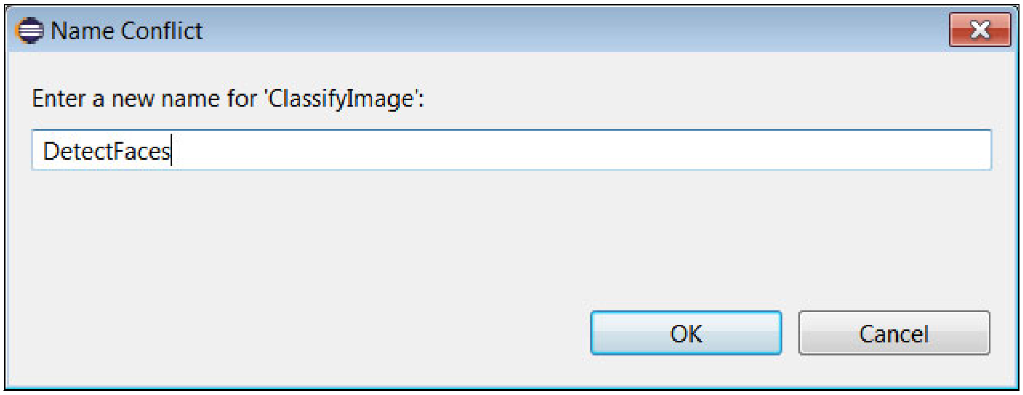


Figure 1-19 A sample for image classification: Fruit dish image

1.4.4 Detecting faces

In this section, you use the Watson Java SDK to detect faces in an image. The API also provides data about the detected faces, such as estimated age, gender, and names of celebrities.

1. Right-click **ClassifyImage.java** (Figure 1-14 on page 13), click **Copy** and **Paste** in the same directory.
2. The Name Conflict dialog opens (Figure 1-20). Enter `DetectFaces` as the new class name and click **OK**.



Name Conflict

Enter a new name for 'ClassifyImage':

DetectFaces

Figure 1-20 Create DetectFaces class

The new class `DetectFaces.java` is listed in Package Explorer (Figure 1-21).

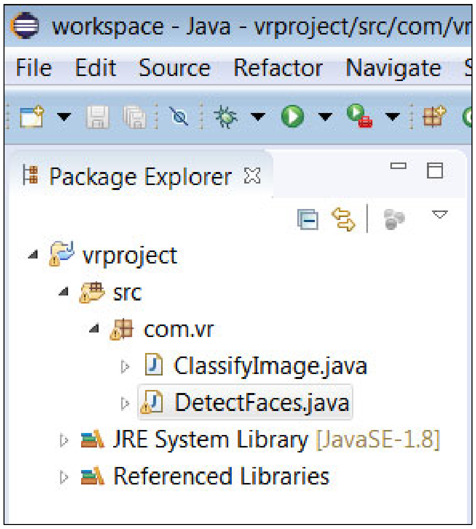

workspace - Java - vrproject/src/com/vr
File Edit Source Refactor Navigate S
Package Explorer
vrproject
src
com.vr
ClassifyImage.java
DetectFaces.java
JRE System Library [JavaSE-1.8]
Referenced Libraries

Figure 1-21 DetectFaces java class in Package Explorer

3. Update the code to call the face detection Watson API.

   Double-click **DetectFace.java** to open the class.

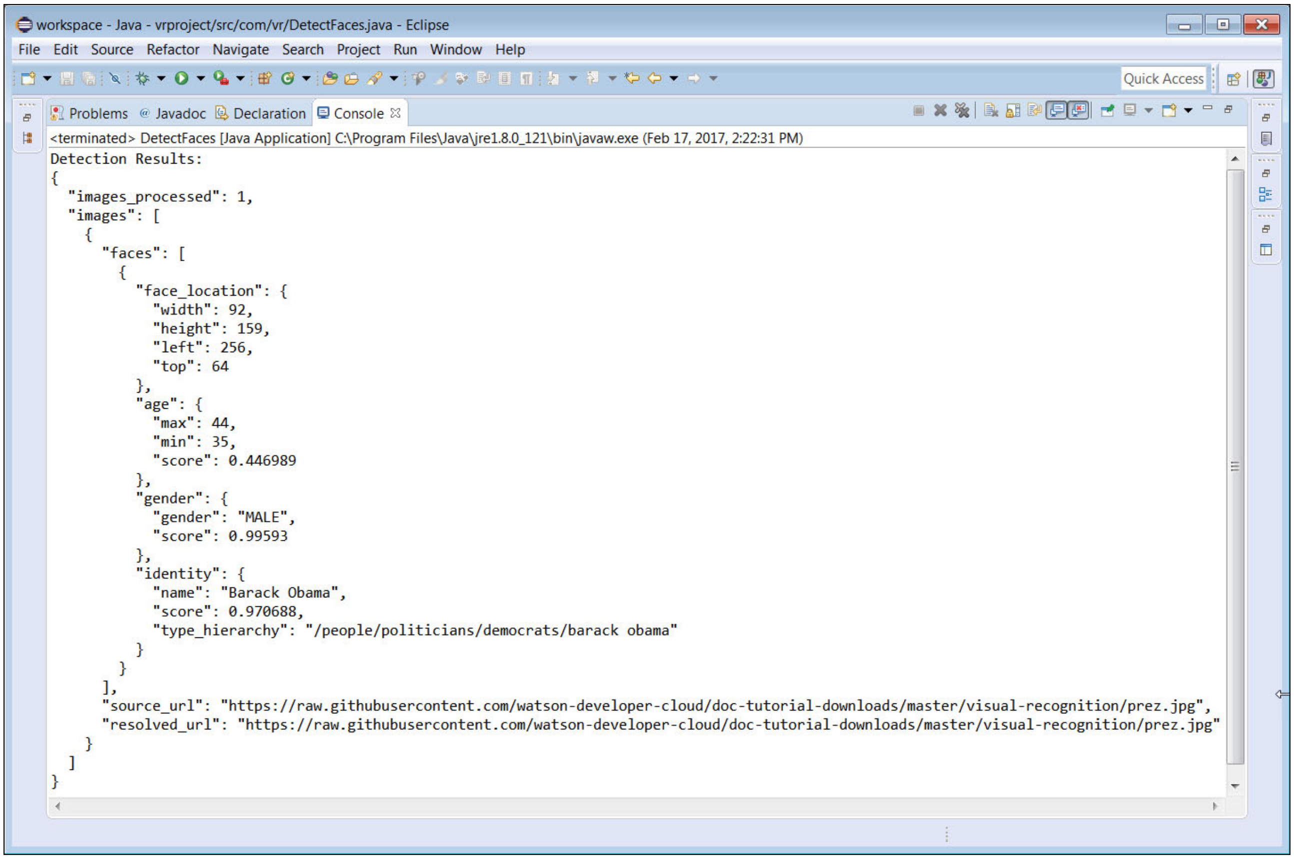
Apply the changes that are highlighted (outlined in red in Example 1-2).

Example 1-2 Code changes to perform face detection

```
package com.vr;

import com.ibm.watson.developer_cloud.visual_recognition.v3.*;
import com.ibm.watson.developer_cloud.visual_recognition.v3.model.*;


public class ClassifyImage {

public static void main(String[] args) {

VisualRecognition service = new
VisualRecognition(VisualRecognition.VERSION_DATE_2016_05_20);

service.setEndPoint("https://gateway-a.watsonplatform.net/visual-recognition/ap
i");

//Here you replace "your_api_key_here" by the API Key you created in "Creating
a //Watson Visual Recognition service instance and getting the API key"

 service.setApiKey("your_api_key_here");
//Here you add the URL of your image. The image size should not exceed 2MB.
 String imageURL = new
String("https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-d
ownloads/master/visual-recognition/prez.jpg");
       VisualRecognitionOptions options = new
VisualRecognitionOptions.Builder().url(imageURL).build(); DetectedFaces result
= service.detectFaces(options).execute();
System.out.println("Detections Results:");
System.out.println(result);
  }

}
```

4. Run code and check the results. Right-click **DetectFaces.java** and select **Run As** → **Java Application** (see Figure 1-17 on page 16).

5. View the results on your Console, which by default is under `DetectFaces`. To maximize the view and see the results (Figure 1-22), double-click the **Console** tab. It shows the response is in JSON format and that a face was detected and recognized as an image of former President Barak Obama. It also detected gender as Male and the estimated age.


workspace - Java - vrproject/src/com/vr/DetectFaces.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Problems
Javadoc
Declaration
Console

<terminated> DetectFaces [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Feb 17, 2017, 2:22:31 PM)

```
Detection Results:
{
  "images_processed": 1,
  "images": [
    {
      "faces": [
        {
          "face_location": {
            "width": 92,
            "height": 159,
            "left": 256,
            "top": 64
          },
          "age": {
            "max": 44,
            "min": 35,
            "score": 0.446989
          },
          "gender": {
            "gender": "MALE",
            "score": 0.99593
          },
          "identity": {
            "name": "Barack Obama",
            "score": 0.970688,
            "type_hierarchy": "/people/politicians/democrats/barack obama"
          }
        }
      ],
      "source_url": "https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/prez.jpg",
      "resolved_url": "https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/prez.jpg"
    }
  ]
}
```

Figure 1-22   Results of running the face detection service

Figure 1-23 represents the image used in face detection.

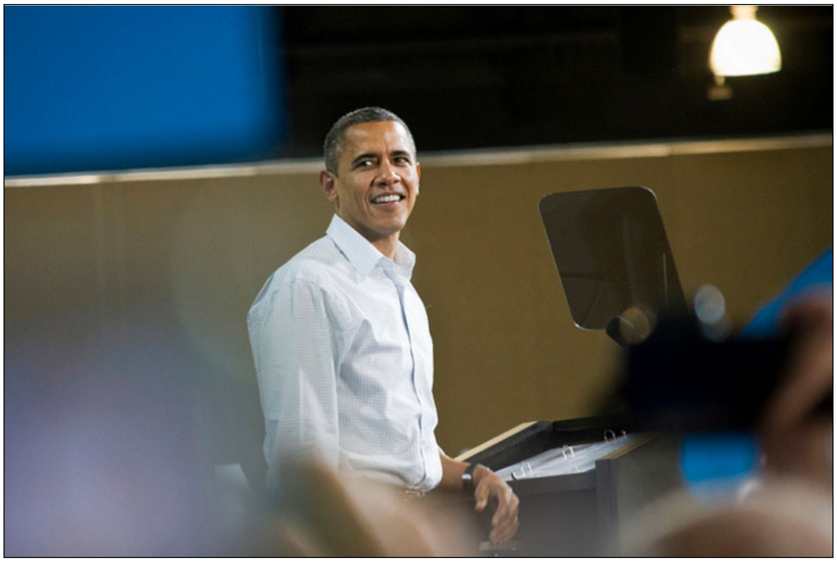


Figure 1-23   Sample image for face detection: Barak Obama

1.5 Classifying images and detecting faces: Use Watson Node.js SDK and Node.js Express framework

By the end of this section, you should be able to accomplish these objectives:

- Use the Watson Node.js SDK to call Watson APIs for image classification.
- Use the Watson Node.js SDK to call Watson APIs to detect faces, gender, and age in an image.

Implementing this use case using the Watson Node.js SDK involves the following steps:

1. Creating a Watson Visual Recognition service instance and getting the API key.
2. Installing the Watson Node.js SDK into your project.
3. Classifying images.
4. Detecting faces.

For more information about the Node.js client library to use the Watson services, see the Watson Developer Cloud Node.js SDK web page.

You can find several Node.js usage examples of the Watson APIs on GitHub.

1.5.1 Installing the Watson Node.js SDK into your project

For Node.js you need to *enable* the Watson API by installing the SDK into your local Node.js installation and the project you are currently working on:

1. You need a text editor to enter and edit the code. Use your favorite text editor or download Brackets or Atom, which are two very popular code editors.
2. Install Node.js runtime and node package manager (npm) on your system from the Node.js website.
3. After you initiate your Node.js project, install the Watson Node SDK into your local installation and Node.js project:

```
npm install -g watson-developer-cloud
npm install --save watson-developer-cloud
```

1.5.2 Classifying images

The Node.js sample code in Example 1-3 does the following tasks:

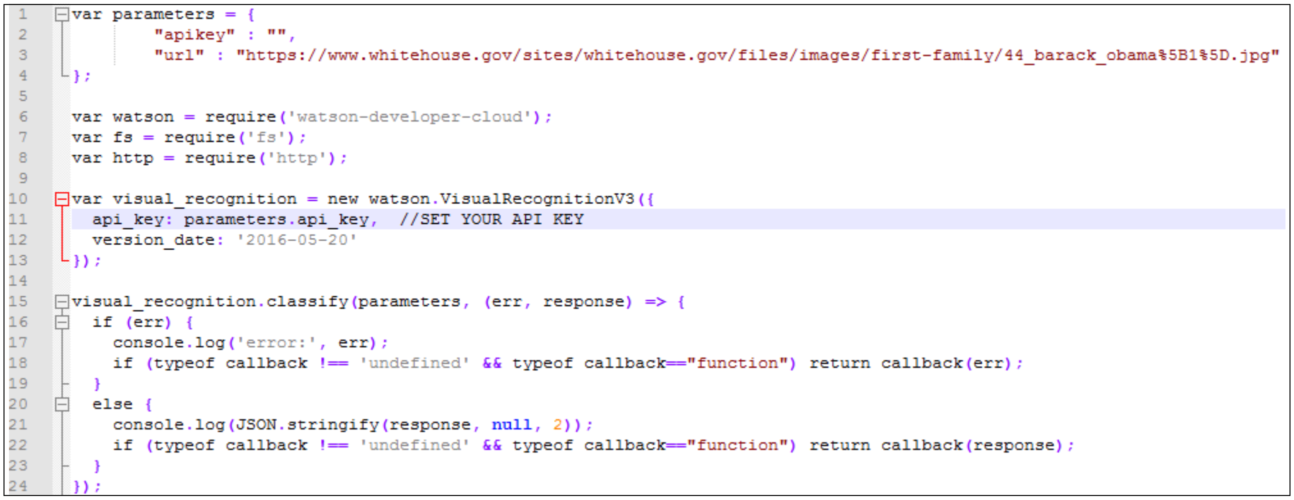
1. Gets an image from a website URL
2. Sets the API key of the Visual Recognition service
3. Sends the image to the `classify` method of the Visual Recognition service for processing.
4. Returns the results in JSON format.

Example 1-3 Image classification: Node.js sample code

```
var parameters = {
     "apikey" : "",
     "url" :
"https://www.whitehouse.gov/sites/whitehouse.gov/files/images/first-family/44_bara
ck_obama%5B1%5D.jpg"
};

var watson = require('watson-developer-cloud');
var fs = require('fs');
var http = require('http');

var visual_recognition = new watson.VisualRecognitionV3({
  api_key: parameters.api_key,  //SET YOUR API KEY
  version_date: '2016-05-20'
});

visual_recognition.classify(parameters, (err, response) => {
  if (err) {
    console.log('error:', err);
    if (typeof callback !== 'undefined' && typeof callback=="function") return 
callback(err);
  }
  else {
    console.log(JSON.stringify(response, null, 2));
    if (typeof callback !== 'undefined' && typeof callback=="function") return 
callback(response);
  }
});
```

Note the following important lines in the Node.js code snippet, shown in Figure 1-24:

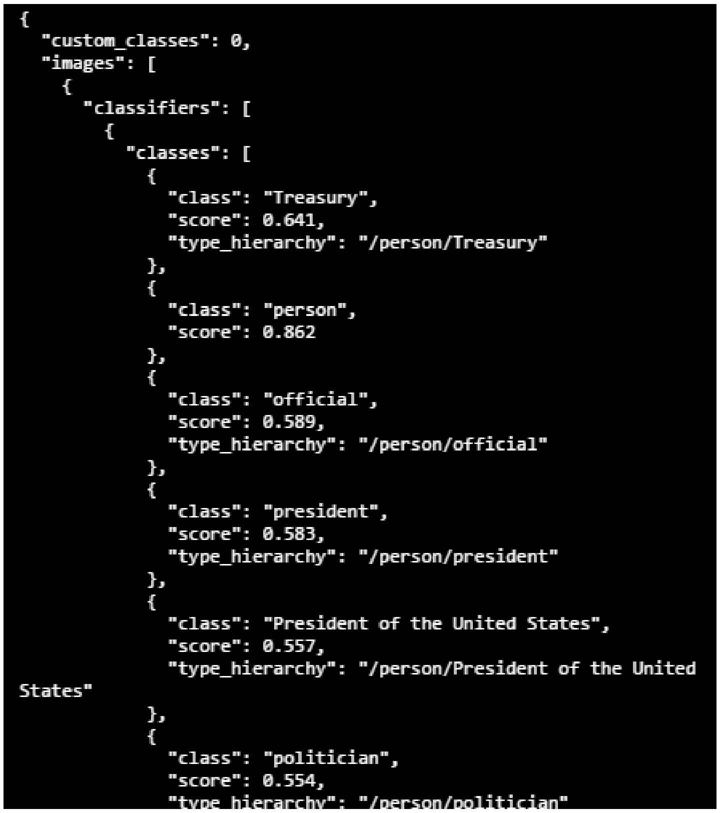
- Line 3: The URL that supplies the image as input for processing.
- Line 11: Set your `api_key` of the Visual Recognition service created in 1.2, “Creating a Watson Visual Recognition service instance and getting the API key” on page 2.
- Line 15: Call the `classify` method passing the image `url` and `api_key`.



```
1  var parameters = {
2          "apikey" : "",
3          "url" : "https://www.whitehouse.gov/sites/whitehouse.gov/files/images/first-family/44_barack_obama%5B1%5D.jpg"
4  };
5
6  var watson = require('watson-developer-cloud');
7  var fs = require('fs');
8  var http = require('http');
9
10 var visual_recognition = new watson.VisualRecognitionV3({
11   api_key: parameters.api_key,  //SET YOUR API KEY
12   version_date: '2016-05-20'
13 });
14
15 visual_recognition.classify(parameters, (err, response) => {
16   if (err) {
17     console.log('error:', err);
18     if (typeof callback !== 'undefined' && typeof callback=="function") return callback(err);
19   }
20   else {
21     console.log(JSON.stringify(response, null, 2));
22     if (typeof callback !== 'undefined' && typeof callback=="function") return callback(response);
23   }
24 });
```

Figure 1-24   Classify object: JSON snippet highlights

Figure 1-25 shows the response, in JSON format. It describes the image content and includes a score for each class.



```
{
  "custom_classes": 0,
  "images": [
    {
      "classifiers": [
        {
          "classes": [
            {
              "class": "Treasury",
              "score": 0.641,
              "type_hierarchy": "/person/Treasury"
            },
            {
              "class": "person",
              "score": 0.862
            },
            {
              "class": "official",
              "score": 0.589,
              "type_hierarchy": "/person/official"
            },
            {
              "class": "president",
              "score": 0.583,
              "type_hierarchy": "/person/president"
            },
            {
              "class": "President of the United States",
              "score": 0.557,
              "type_hierarchy": "/person/President of the United
States"
            },
            {
              "class": "politician",
              "score": 0.554,
              "type_hierarchy": "/person/politician"
```

Figure 1-25 Results

1.5.3 Detecting faces

In this section, you use the Watson Node.js SDK to detect faces in an image. The API also provides data about the detected faces, such as estimated age, gender, and names of celebrities.

The Node.js sample code in Example 1-4 on page 25 performs the following tasks:

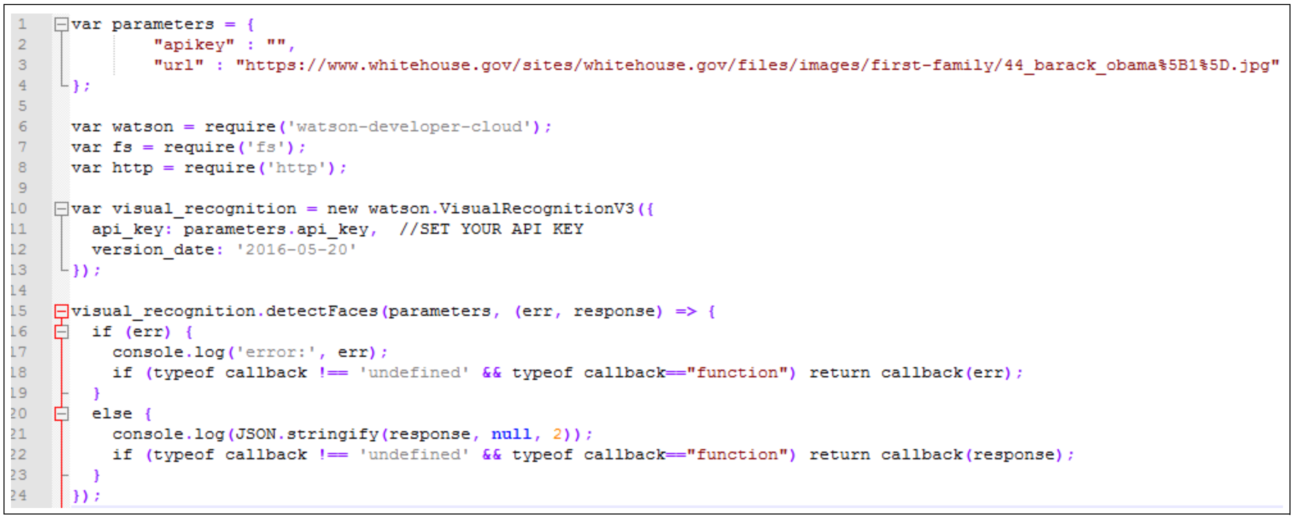
1. Gets an image from a website URL.
2. Sets the API key of the Visual Recognition service.
3. Sends the image to the `detectFaces` method of the Visual Recognition service for processing.
4. Returns the results in JSON format.

Example 1-4   Face detection: Node.js sample code

```
var parameters = {
      "apikey" : "",
      "url" :
"https://www.whitehouse.gov/sites/whitehouse.gov/files/images/first-family/44_bara
ck_obama%5B1%5D.jpg"
};

var watson = require('watson-developer-cloud');
var fs = require('fs');
var http = require('http');

var visual_recognition = new watson.VisualRecognitionV3({
  api_key: parameters.api_key,  //SET YOUR API KEY
  version_date: '2016-05-20'
});

visual_recognition. detectFaces (parameters, (err, response) => {
  if (err) {
    console.log('error:', err);
    if (typeof callback !== 'undefined' && typeof callback=="function") return
callback(err);
  }
  else {
    console.log(JSON.stringify(response, null, 2));
    if (typeof callback !== 'undefined' && typeof callback=="function") return
callback(response);
  }
});
```

Note the following important lines in the Node.js code snippet, shown in Figure 1-26:

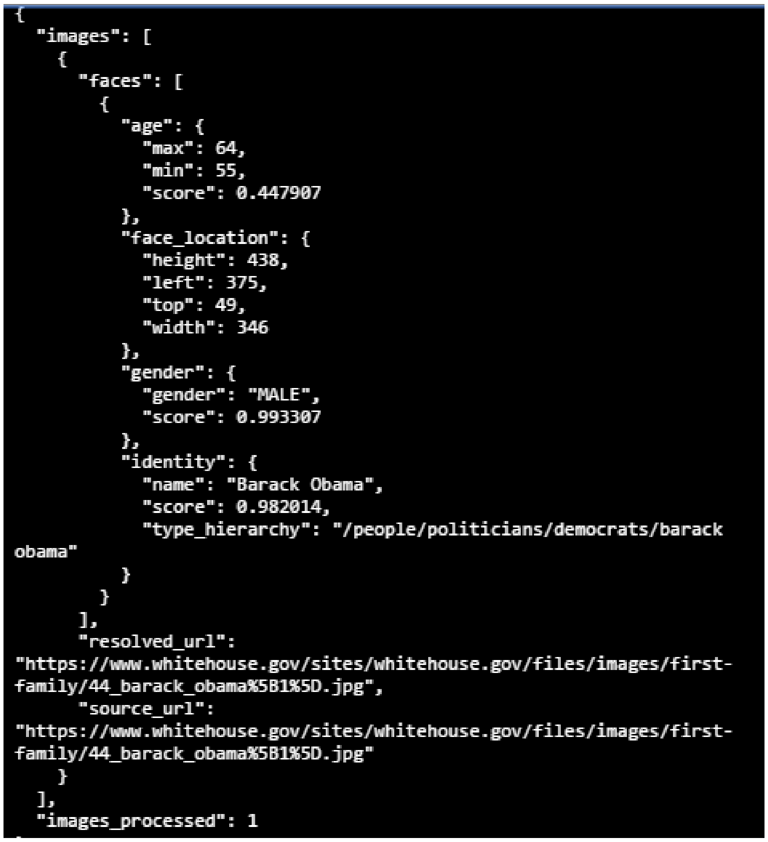
- Line 3: The URL that supplies the image as input for processing.
- Line 11: Set your `api_key` of Visual Recognition service created in 1.2, “Creating a Watson Visual Recognition service instance and getting the API key” on page 2.
- Line 15: Call the `detectFaces` method passing the image `url` and `api_key`.



```
1   var parameters = {
2           "apikey" : "",
3           "url" : "https://www.whitehouse.gov/sites/whitehouse.gov/files/images/first-family/44_barack_obama%5B1%5D.jpg"
4   };
5
6   var watson = require('watson-developer-cloud');
7   var fs = require('fs');
8   var http = require('http');
9
10  var visual_recognition = new watson.VisualRecognitionV3({
11    api_key: parameters.api_key,  //SET YOUR API KEY
12    version_date: '2016-05-20'
13  });
14
15  visual_recognition.detectFaces(parameters, (err, response) => {
16    if (err) {
17      console.log('error:', err);
18      if (typeof callback !== 'undefined' && typeof callback=="function") return callback(err);
19    }
20    else {
21      console.log(JSON.stringify(response, null, 2));
22      if (typeof callback !== 'undefined' && typeof callback=="function") return callback(response);
23    }
24  });
```

Figure 1-26   Face detection: JSON snippet highlights

Figure 1-27 shows the results in JSON format. The face of a celebrity, former President Barak Obama, was detected and data about the face is provided (gender, estimated age).



```
{
  "images": [
    {
      "faces": [
        {
          "age": {
            "max": 64,
            "min": 55,
            "score": 0.447907
          },
          "face_location": {
            "height": 438,
            "left": 375,
            "top": 49,
            "width": 346
          },
          "gender": {
            "gender": "MALE",
            "score": 0.993307
          },
          "identity": {
            "name": "Barack Obama",
            "score": 0.982014,
            "type_hierarchy": "/people/politicians/democrats/barack
obama"
          }
        }
      ],
      "resolved_url":
"https://www.whitehouse.gov/sites/whitehouse.gov/files/images/first-
family/44_barack_obama%5B1%5D.jpg",
      "source_url":
"https://www.whitehouse.gov/sites/whitehouse.gov/files/images/first-
family/44_barack_obama%5B1%5D.jpg"
    }
  ],
  "images_processed": 1
```

Figure 1-27 Expected output

1.6 References

See the following resources:

- Overview of the IBM Watson Visual Recognition service:

  https://www.ibm.com/watson/developercloud/doc/visual-recognition/index.html

- Watson Developer Cloud: Visual Recognition:

  https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/

- Classify an image:

  https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/#classify_an_image

- Detect faces:

  https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/#detect_faces

- Visual Recognition getting started tutorials:

  https://www.ibm.com/watson/developercloud/doc/visual-recognition/getting-started.html

- Watson Developer Cloud Node.js SDK:

  https://www.npmjs.com/package/watson-developer-cloud

- Node.js usage examples of the Watson APIs:

  https://github.com/watson-developer-cloud/node-sdk

2

Classify images with a custom classifier

The examples in Chapter 1, “Basics of Watson Visual Recognition service” on page 1, use the pre-trained classifier to classify images.

You can also train and create a custom classifier. With a custom classifier, you can train the Visual Recognition service to classify images to suit your business needs. By creating a custom classifier, you can use the Visual Recognition service to recognize images that are *not* available with pre-trained classification.

This chapter shows you how to create and train a custom classifier and use it to classify a new image.

The following topics are covered in this chapter:

- Visual Recognition custom classifier overview
- Train, create, and use a custom classifier
- References

2.1 Visual Recognition custom classifier overview

The Watson Visual Recognition service can learn from example images that you upload to create a new classifier. Each example file is trained against the other files uploaded when you create the classifier and positive examples are stored as classes. These classes are grouped to define a single classifier, but return their own scores.

Figure 2-1 shows an overview of the process to use the Watson Visual Recognition service with a custom classifier.

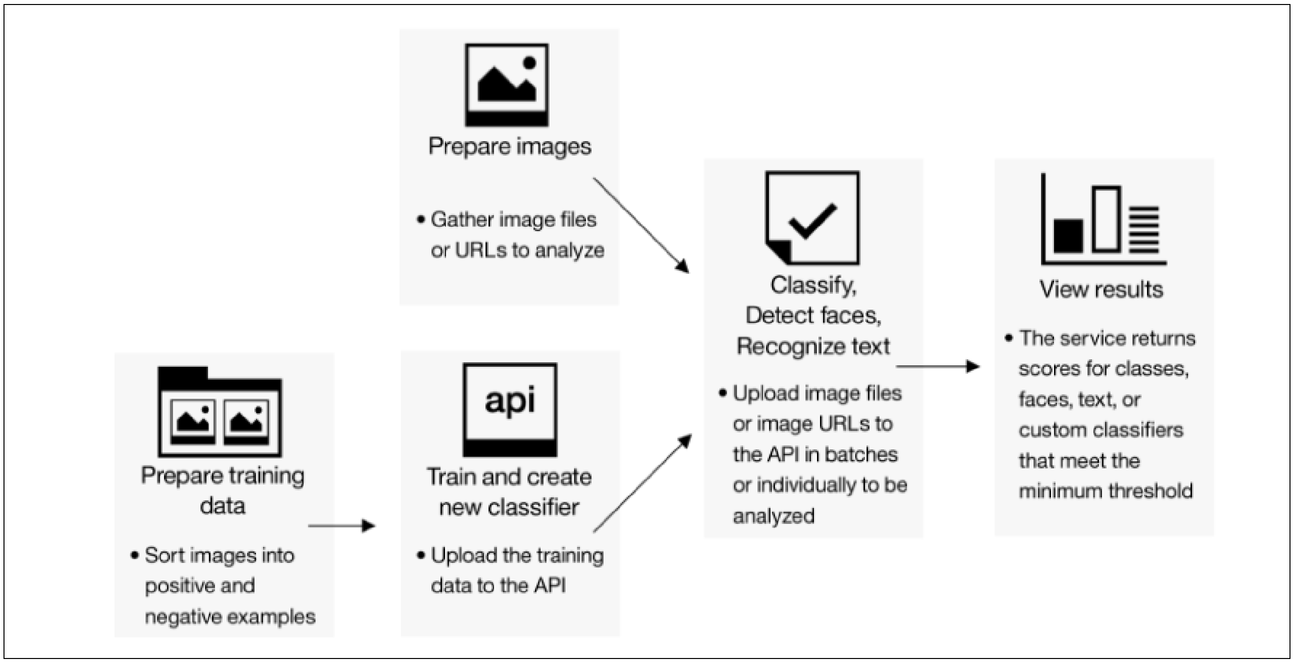

Prepare images
• Gather image files or URLs to analyze
Classify, Detect faces, Recognize text
• Upload image files or image URLs to the API in batches or individually to be analyzed
View results
• The service returns scores for classes, faces, text, or custom classifiers that meet the minimum threshold
Prepare training data
• Sort images into positive and negative examples
api
Train and create new classifier
• Upload the training data to the API

Figure 2-1 Visual Recognition process with custom classifier

A new custom classifier can be trained by several compressed (.zip) files, including files containing positive or negative examples of images (.jpg or .png). You must supply at least two compressed files, either two positive example files or one positive and one negative example file.

Compressed files containing positive examples are used to create *classes* that define what the new classifier is. The prefix that you specify for each positive example parameter is used as the class name within the new classifier. The _positive_examples suffix is required. There is no limit on the number of positive example files that you can upload in a single call.

The compressed file containing negative examples is not used to create a class within the created classifier, but does define what the new classifier is not. Negative example files should contain images that do not depict the subject of any of the positive examples. You can specify only one negative example file in a single call. For more information, see these web pages:

- Overview of the IBM Watson Visual Recognition service

  https://www.ibm.com/watson/developercloud/doc/visual-recognition/index.html

- Guidelines for training classifiers

  https://www.ibm.com/watson/developercloud/doc/visual-recognition/customizing.html

2.2 Train, create, and use a custom classifier

By the end of this chapter, you should be able to accomplish these objectives:

- Create a custom classifier and upload positive and negative image files examples.
- Get the custom classifier ID.
- Classify a new image using a newly trained custom classifier.
- Get results in JSON format containing `class`, `score`, and `type hierarchy.`

To accomplish these objectives, you will do the following steps:

- Prepare training data.
- Create a Watson Visual Recognition service instance and getting the API key as described in 1.2, “Creating a Watson Visual Recognition service instance and getting the API key” on page 2.
- Create and train the classifier.
- Classify an image with a custom classifier.

2.2.1 Prepare training data

Gather image files to use as positive and negative example training data. Download the following ZIP files:

- `beagle.zip` (positive example)

  https://watson-developer-cloud.github.io/doc-tutorial-downloads/visual-recognition/beagle.zip

- `husky.zip` (positive example)

  https://watson-developer-cloud.github.io/doc-tutorial-downloads/visual-recognition/husky.zip

- `golden-retriever.zip` (positive example)

  https://watson-developer-cloud.github.io/doc-tutorial-downloads/visual-recognition/golden-retriever.zip

- `cats.zip` (negative example)

  https://watson-developer-cloud.github.io/doc-tutorial-downloads/visual-recognition/cats.zip

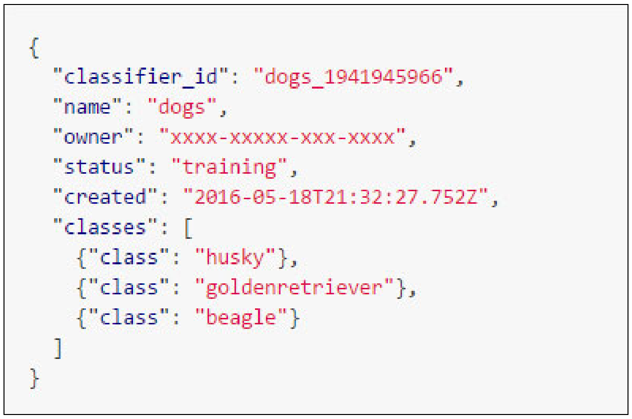
2.2.2 Create and train the classifier

The sample code in Example 2-1 specifies the location of the training images and creates the custom classifier. Positive example file names require the suffix `_positive_examples`; the prefix (beagle, golden_retriever, and husky) is returned as the name of the class. Notice that a negative example file is also provided.

Example 2-1 Specify location of training images and create classifier

```
var watson = require('watson-developer-cloud');
var fs = require('fs');

var visual_recognition = watson.visual_recognition({
  api_key: '{api_key}',
  version: 'v3',
  version_date: '2016-05-19'
});

var params = {
   name: 'dog',
   beagle_positive_examples: fs.createReadStream('./ public/resource/beagle.zip'),
    husky_positive_examples: fs.createReadStream('./ public/resource/husky.zip'),
    golden_retriever_positive_examples: fs.createReadStream('./
public/resource/golden-retriever.zip'),
   negative_examples: fs.createReadStream('./ public/resource/cats.zip')
};

visual_recognition.createClassifier(params,
   function(err, response) {
    if (err)
      console.log(err);
     else
      console.log(JSON.stringify(response, null, 2));
});
```

The sample output in Figure 2-2 shows that the `classifier_id` is returned.



```
{
  "classifier_id": "dogs_1941945966",
  "name": "dogs",
  "owner": "xxxx-xxxxx-xxx-xxxx",
  "status": "training",
  "created": "2016-05-18T21:32:27.752Z",
  "classes": [
    {"class": "husky"},
    {"class": "goldenretriever"},
    {"class": "beagle"}
  ]
}
```

Figure 2-2 Returned classifier_id

2.2.3 Classify an image with a custom classifier

The code snippet shown in Example 2-2 is used to classify a new image with the custom classifier. Compare this example to Example 1-3 on page 22. The difference is that in Example 2-2 you specify the `classifier_id` of the the custom classifier created in 2.2.2, “Create and train the classifier” on page 32.

1. Download the following image file to use as the input image to classify:

   https://raw.githubusercontent.com/watson-developer-cloud/doc-tutorial-downloads/master/visual-recognition/dogs.jpg

2. Enter the code from Example 2-2 to classify the image. Make these changes:

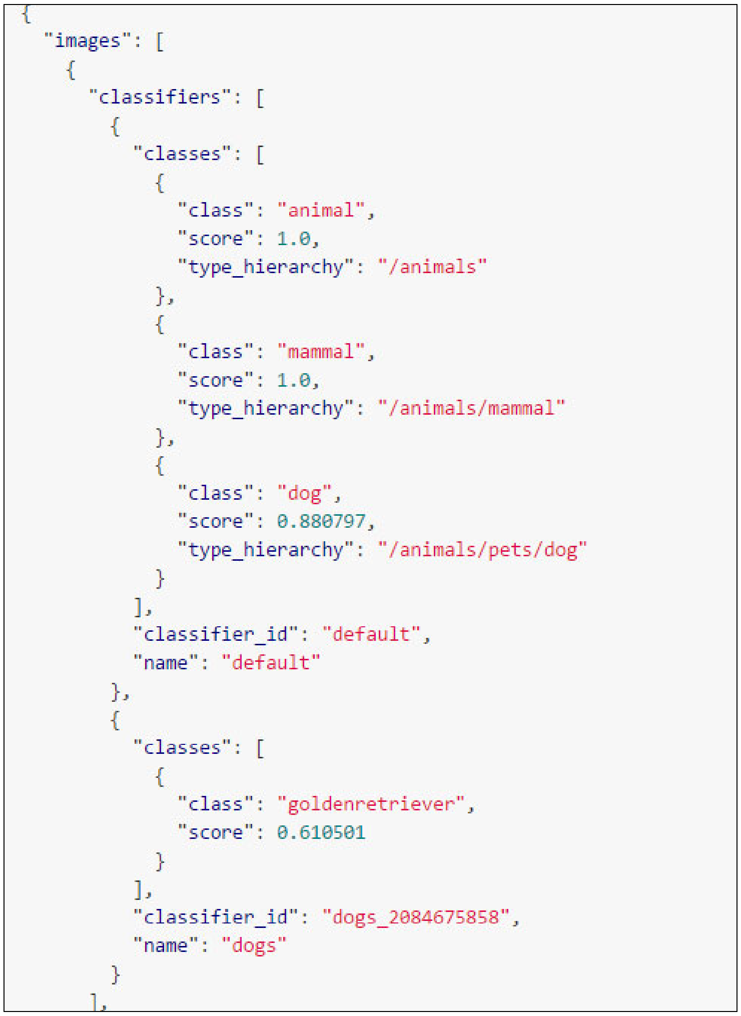
   – Replace `api_key` with the key that you obtained when creating the Visual Recognition service, as described in 1.2, “Creating a Watson Visual Recognition service instance and getting the API key” on page 2.

   – Replace `custom_classifer_id` with the ID that you obtained when you created the custom classifier in 2.2.2, “Create and train the classifier” on page 32.

Example 2-2   Code snippet to classify a new image with a custom classifier

```
var watson = require('watson-developer-cloud');
var fs = require('fs');

var visual_recognition = watson.visual_recognition({
  api_key: '<api_key>',
  version: 'v3',
  version_date: '2016-05-20'
});

var params = {
  images_file: fs.createReadStream('./public/resource/dogs.jpg'),
  classifier_ids: ["<custom_classifer_id", "default"]
};

visual_recognition.classify(params, function(err, res) {
  if (err)
    console.log(err);
  else
    console.log(JSON.stringify(res, null, 2));
});
```

The sample output is shown in Figure 2-3.



```
{
  "images": [
    {
      "classifiers": [
        {
          "classes": [
            {
              "class": "animal",
              "score": 1.0,
              "type_hierarchy": "/animals"
            },
            {
              "class": "mammal",
              "score": 1.0,
              "type_hierarchy": "/animals/mammal"
            },
            {
              "class": "dog",
              "score": 0.880797,
              "type_hierarchy": "/animals/pets/dog"
            }
          ],
          "classifier_id": "default",
          "name": "default"
        },
        {
          "classes": [
            {
              "class": "goldenretriever",
              "score": 0.610501
            }
          ],
          "classifier_id": "dogs_2084675858",
          "name": "dogs"
        }
      ],
```

Figure 2-3 Sample output

2.3 References

See the following resources:

- Overview of the IBM Watson Visual Recognition service:

  https://www.ibm.com/watson/developercloud/doc/visual-recognition/index.html

- Guidelines for training classifiers:

  https://www.ibm.com/watson/developercloud/doc/visual-recognition/customizing.html

3

Image Content Description

This chapter focuses on the development of Java programs using the Watson Visual Recognition service, which uses deep learning algorithms to analyze images, to generate image content description.

In this chapter, you review the source code for a sample application, Image Content Description, which is a program written in Java and uses the Watson Visual Recognition services. You can also run the program in Eclipse on Linux or Windows. The majority of steps are similar for both systems.

The following topics are covered in this chapter:

- Getting started
- Architecture
- Implementation
- Deploy a Java application to Bluemix
- References

3.1 Getting started

To start, read through the objectives, prerequisites, and expected results of this use case.

3.1.1 Objectives

By the end of this chapter, you should be able to write a Java program that uses the Java classes that are provided with the Watson Visual Recognition service:

- To access the service:
  - VisualRecognition
- To classify and describe objects in an image:
  - ClassifyImagesOptions
  - VisualClassification
- To recognize celebrity faces in images, analyze them, and get data about the person:
  - DetectedFaces
  - VisualRecognitionOptions

3.1.2 Prerequisites

You must have the following accounts, resources, knowledge, and experiences:

- An IBM Bluemix account (register for a new account or log in to Bluemix if you already have an account)
- Eclipse IDE Luna
- Java 8
- The Cloud Foundry command-line interface (CLI)

3.1.3 Expected results

By following the steps in this chapter, you should be able to submit images to the application and obtain results after the image is analyzed by the Watson Visual Recognition services:

1. Input the image shown in Figure 3-1 on page 37.

   The program results are shown in Figure 3-2 on page 37 and Figure 3-3 on page 38.

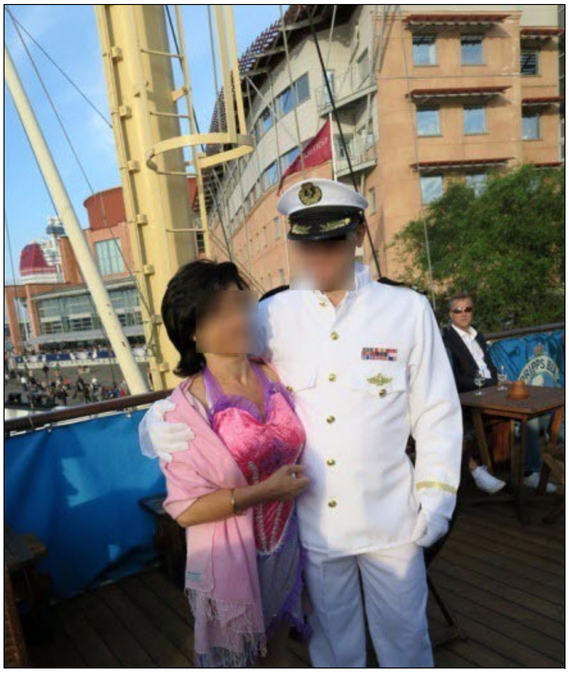
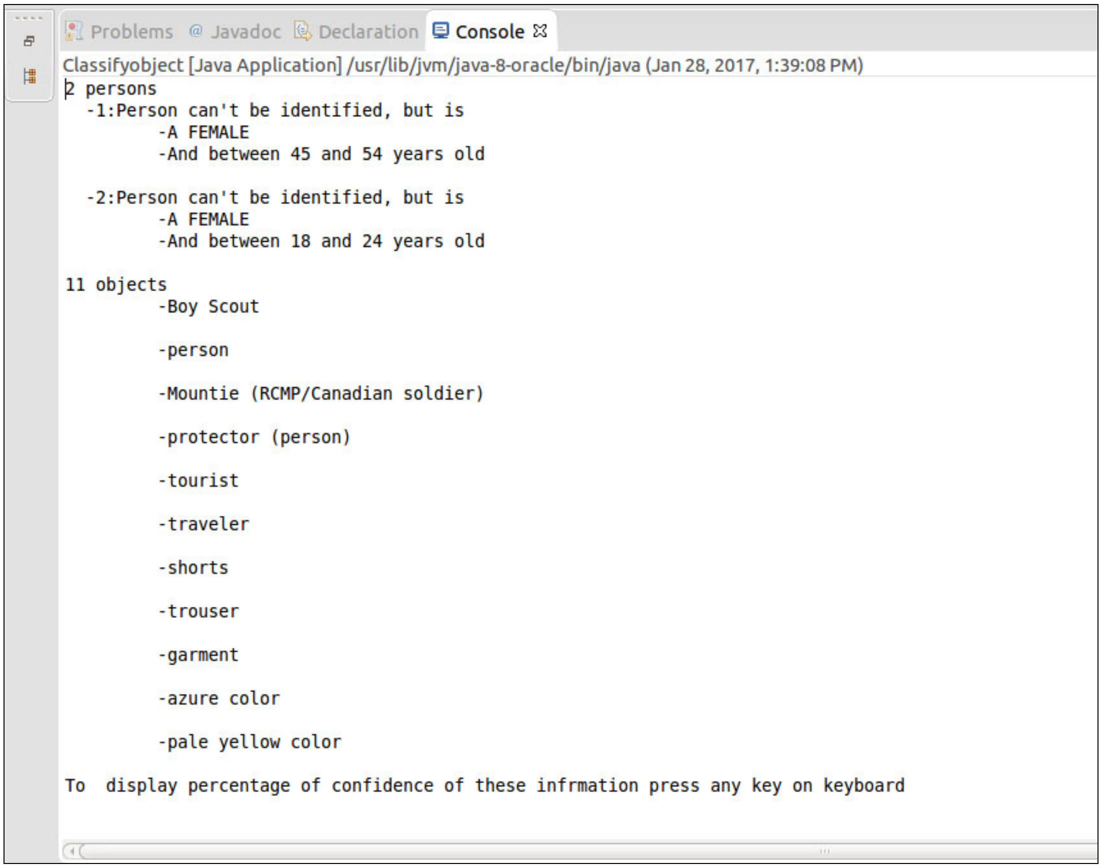


Figure 3-1 The input image to be described

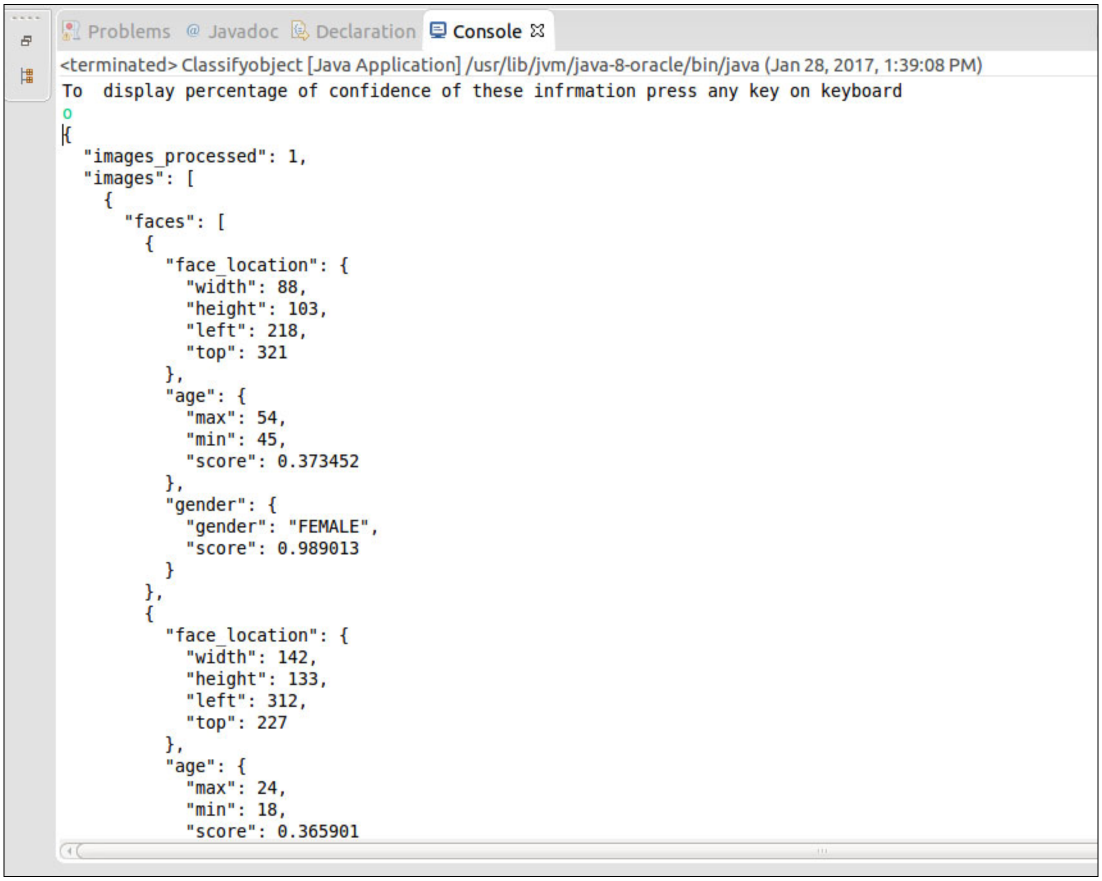
2. Maximize the console window to show the details (Figure 3-2).



```
Classifyobject [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Jan 28, 2017, 1:39:08 PM)
2 persons
  -1:Person can't be identified, but is
         -A FEMALE
         -And between 45 and 54 years old

  -2:Person can't be identified, but is
         -A FEMALE
         -And between 18 and 24 years old

11 objects
         -Boy Scout

         -person

         -Mountie (RCMP/Canadian soldier)

         -protector (person)

         -tourist

         -traveler

         -shorts

         -trouser

         -garment

         -azure color

         -pale yellow color

To  display percentage of confidence of these infrmation press any key on keyboard
```

Figure 3-2 Expected results for input image (part 1 of 2)

3. To display JSON files with more details, press any key (Figure 3-3).



```
<terminated> Classifyobject [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Jan 28, 2017, 1:39:08 PM)
To  display percentage of confidence of these infrmation press any key on keyboard
o
{
  "images_processed": 1,
  "images": [
    {
      "faces": [
        {
          "face_location": {
            "width": 88,
            "height": 103,
            "left": 218,
            "top": 321
          },
          "age": {
            "max": 54,
            "min": 45,
            "score": 0.373452
          },
          "gender": {
            "gender": "FEMALE",
            "score": 0.989013
          }
        },
        {
          "face_location": {
            "width": 142,
            "height": 133,
            "left": 312,
            "top": 227
          },
          "age": {
            "max": 24,
            "min": 18,
            "score": 0.365901
```

Figure 3-3 Expected results for input image (part 2 of 2)

Another function is the capability to recognize celebrity faces demonstrated by using an image of former President Obama (Figure 3-4).

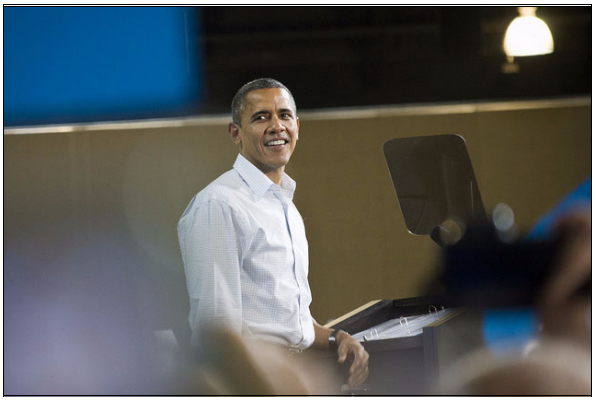
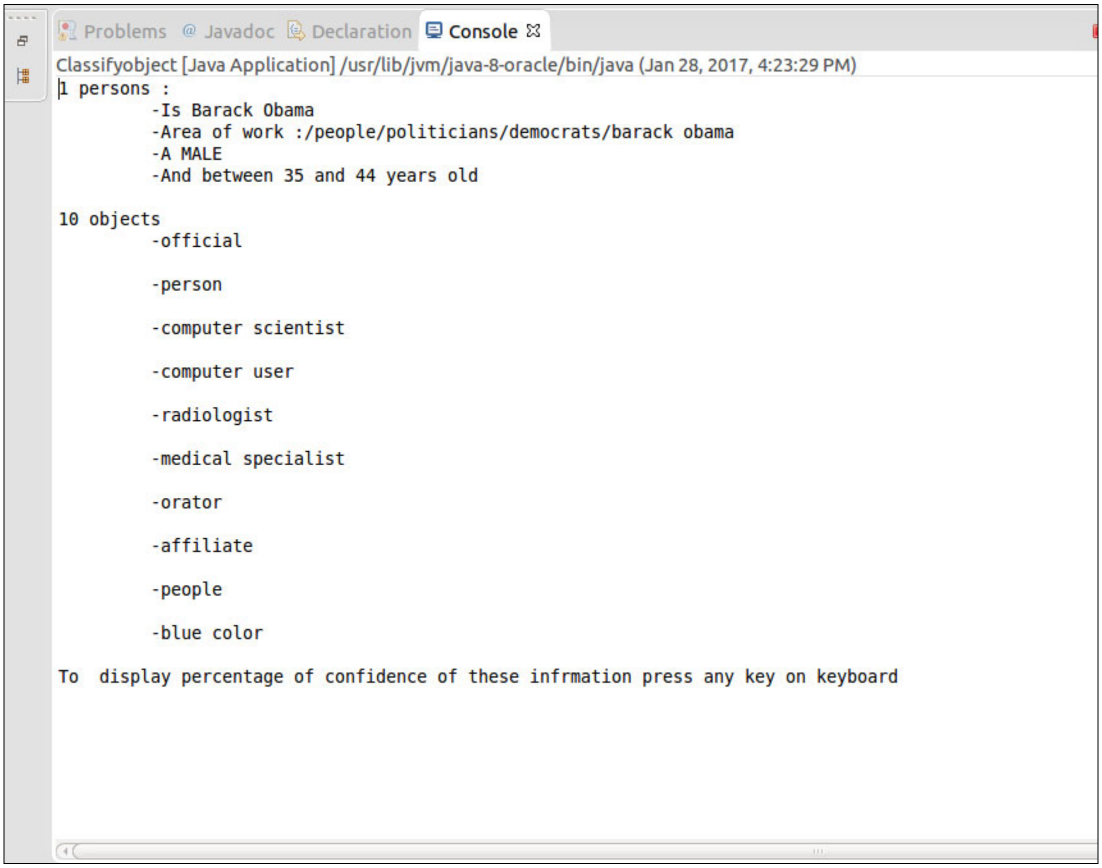


Figure 3-4 Photograph to analyze

The result of analyzing the photograph with the Watson Visual Recognition service is shown in Figure 3-5.



```
Classifyobject [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Jan 28, 2017, 4:23:29 PM)
1 persons :
         -Is Barack Obama
         -Area of work :/people/politicians/democrats/barack obama
         -A MALE
         -And between 35 and 44 years old

10 objects
         -official

         -person

         -computer scientist

         -computer user

         -radiologist

         -medical specialist

         -orator

         -affiliate

         -people

         -blue color

To  display percentage of confidence of these infrmation press any key on keyboard
```

Figure 3-5   Expected result of photograph

3.1.4 Creating, deploying, and running applications that use Bluemix services

To create, deploy, and run an application that uses Bluemix services, you have the following options:

- Create, deploy, and run the application in the Bluemix cloud environment.
- Create and run the application locally by using Bluemix services on the cloud. For example, create a Java application with Eclipse or download the code from GitHub, add the API key and URL endpoint of a Bluemix service instance and run the application as a Java application, after deploying it to Bluemix. This chapter uses this scenario.
- Use the hybrid scenario (Bluemix cloud and local). In this scenario, create the application on Bluemix (cloud) and import it to the local system, modify it, and then deploy to Bluemix.

3.2 Architecture

The flow chart shown in Figure 3-6 summarizes the main activities of the Image Content Description sample program.

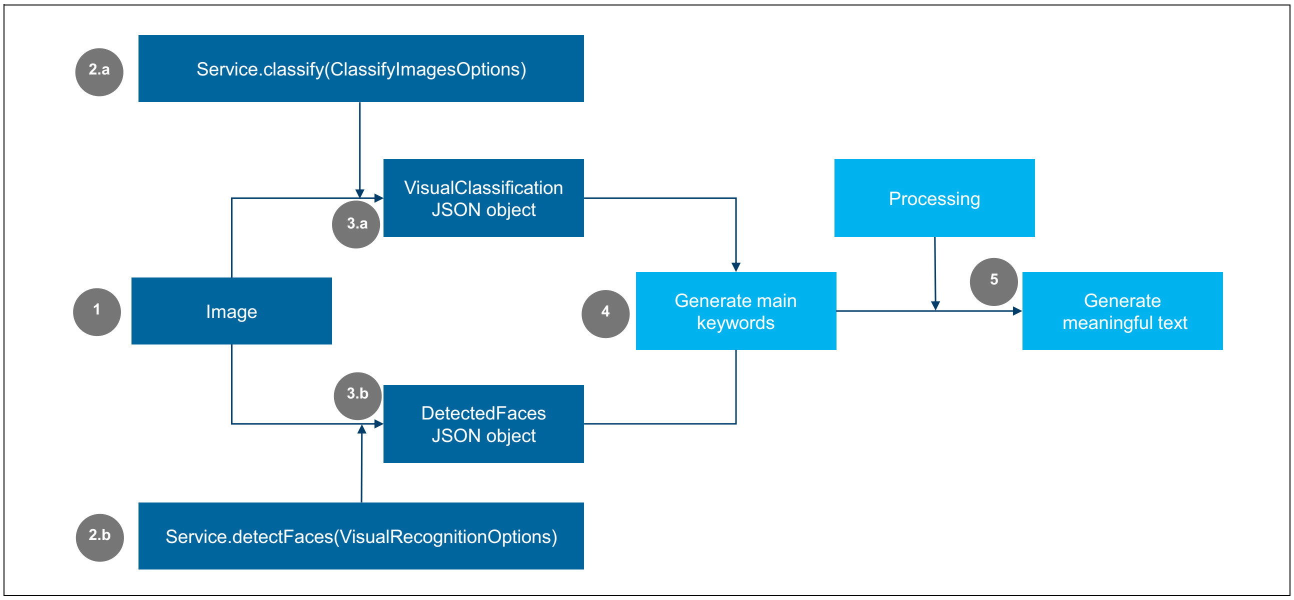

2.a
Service.classify(ClassifyImagesOptions)
VisualClassification JSON object
3.a
Processing
1
Image
4
Generate main keywords
5
Generate meaningful text
3.b
DetectedFaces JSON object
2.b
Service.detectFaces(VisualRecognitionOptions)

Figure 3-6 Flow diagram of the Image Content Description program

The program reads an input image and displays text that describes the image content. Figure 3-6 shows the following flow:

1. Input an image.
2. This step has two activities on the image:
   a. Call the Watson service to classify objects in an image.
   b. Call the Watson service to detect faces in an image.
3. This step has two activities:
   a. VisualClassification contains the JSON representation of the classified objects.
   b. DetectedFaces contains the JSON representation of the faces detected in the image.
4. Generate main keywords, to produce a summary of the image such as number of persons, number of objects, and so on.
5. Process the two obtained JSON objects (DetectedFaces and VisualClassification) to display meaningful text that describes the image content.

3.3 Implementation

Implementing this use case involves the following steps:

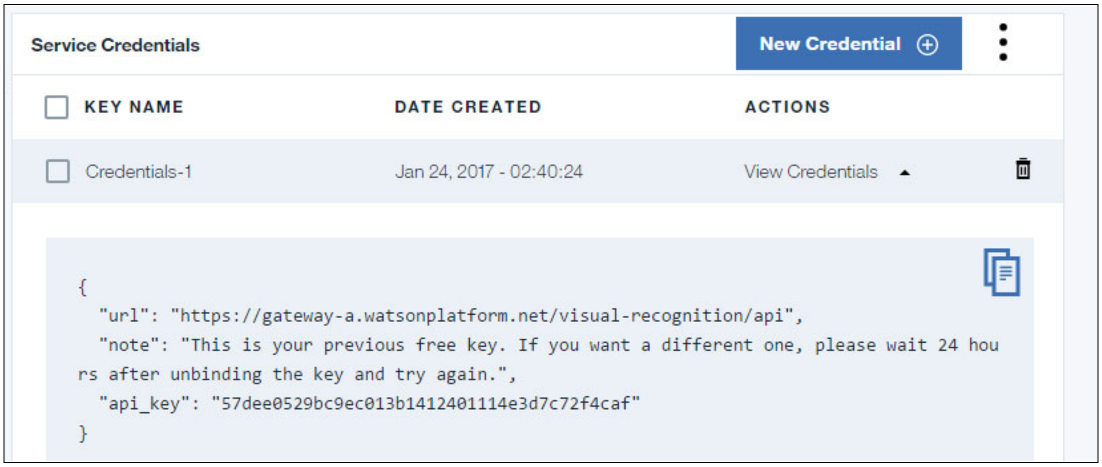
1. Creating a Visual Recognition service instance.
2. Downloading the project from Git.
3. Importing the project into Eclipse.
4. Importing Watson Java SDK.
5. Exploring the sample code provided with the use case.
6. Running the application.

3.3.1 Creating a Visual Recognition service instance

Before you can use the Watson services, you must create an instance of the service in Bluemix. For this use case, create a Visual Recognition service instance as described in 1.2, “Creating a Watson Visual Recognition service instance and getting the API key” on page 2.

After creating the service instance, view the credentials (Figure 3-7). Copy and save the following values for later use:

- `url`, which is the API endpoint
- `api-key`, which is the API key



□ KEY NAME	DATE CREATED	ACTIONS
□ Credentials-1	Jan 24, 2017 - 02:40:24	View Credentials ▲

```
{
  "url": "https://gateway-a.watsonplatform.net/visual-recognition/api",
  "note": "This is your previous free key. If you want a different one, please wait 24 hou
rs after unbinding the key and try again.",
  "api_key": "57dee0529bc9ec013b1412401114e3d7c72f4caf"
}
```

Figure 3-7 Credentials of Visual Recognition service instance

3.3.2 Downloading the project from Git

For this use case, a Git repository is provided, which includes the code to implement the ImageContentDescription application with comments to help you more easily understand. Complete these steps:

1. Download the repository from the GitHub location:

   `https://github.com/snippet-java/redbooks-vis-301-ImageContentDescription`

2. Download the `ImageContentDescription_full.zip` file.
3. Extract the file, which then creates a Java Eclipse Project folder.

3.3.3 Importing the project into Eclipse

In this section, you import the `ImageContentDescription` project into the Eclipse workspace as an existing project.

After you extract the project, complete these steps:

1. Launch the Eclipse IDE. When prompted for a workspace, keep the existing workspace or change the workspace if you want, and click **OK**.
2. In the Eclipse environment, select **File** → **Import** (Figure 3-8).

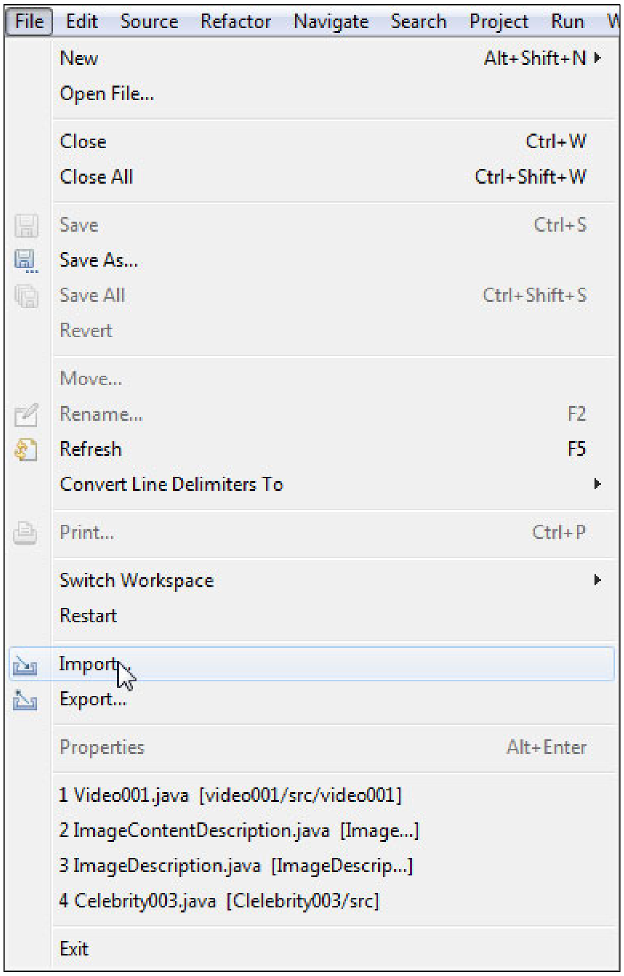
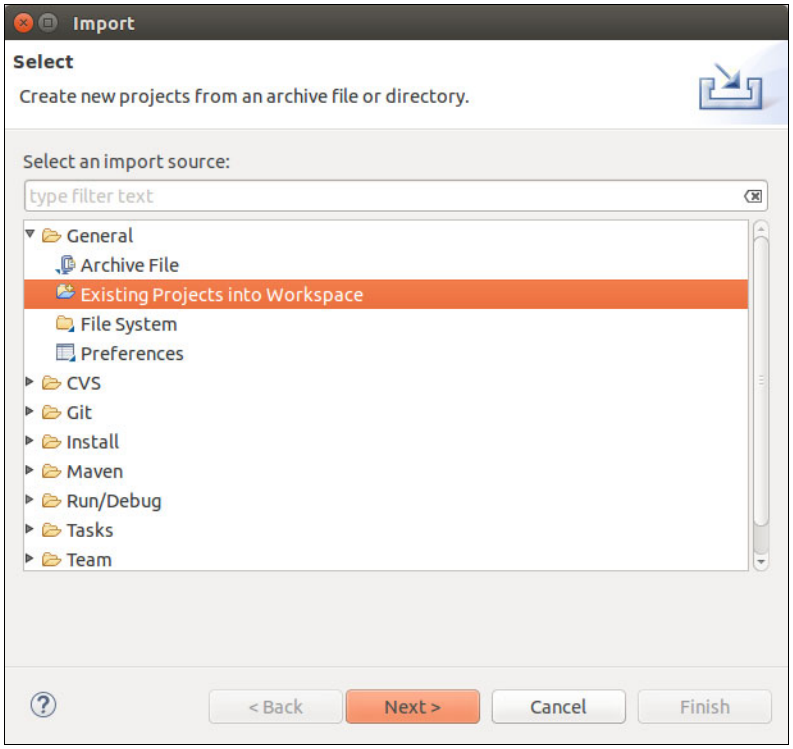


Figure 3-8 Import project menu

3. Select **General** → **Existing Projects into Workspace** (Figure 3-9) and click **Next**. The import process has three pages.



Import

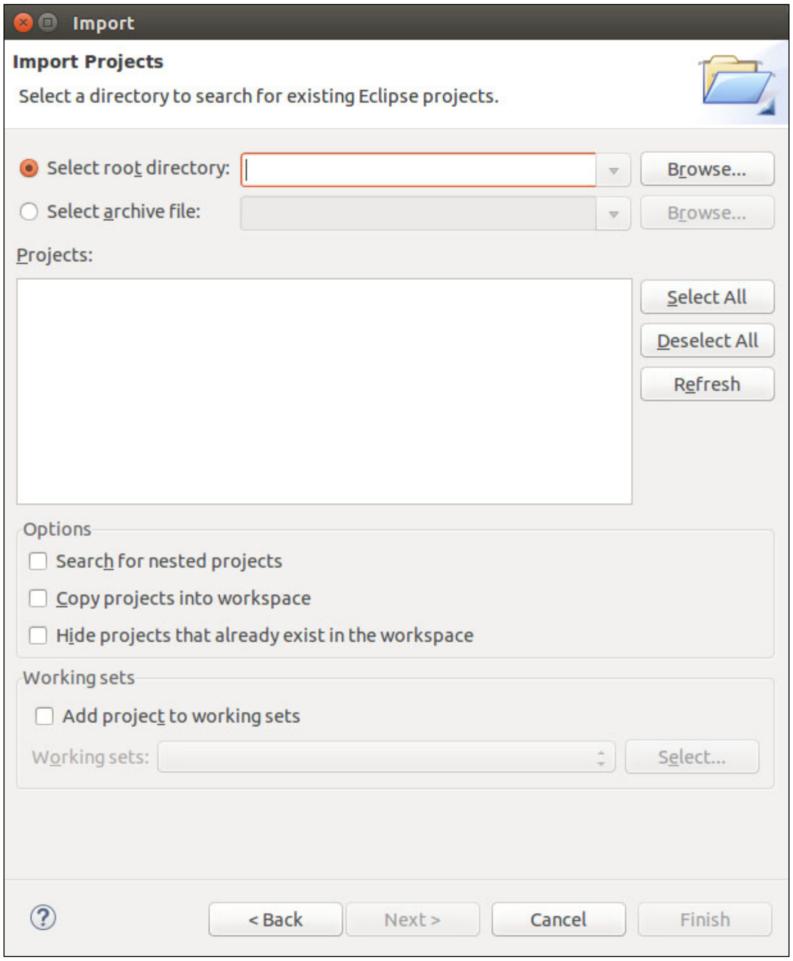
Select

Create new projects from an archive file or directory.

Select an import source:

Figure 3-9 Type imported project dialog

4. Select a root directory. Click **Browse** to navigate to your project’s directory (Figure 3-10).



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

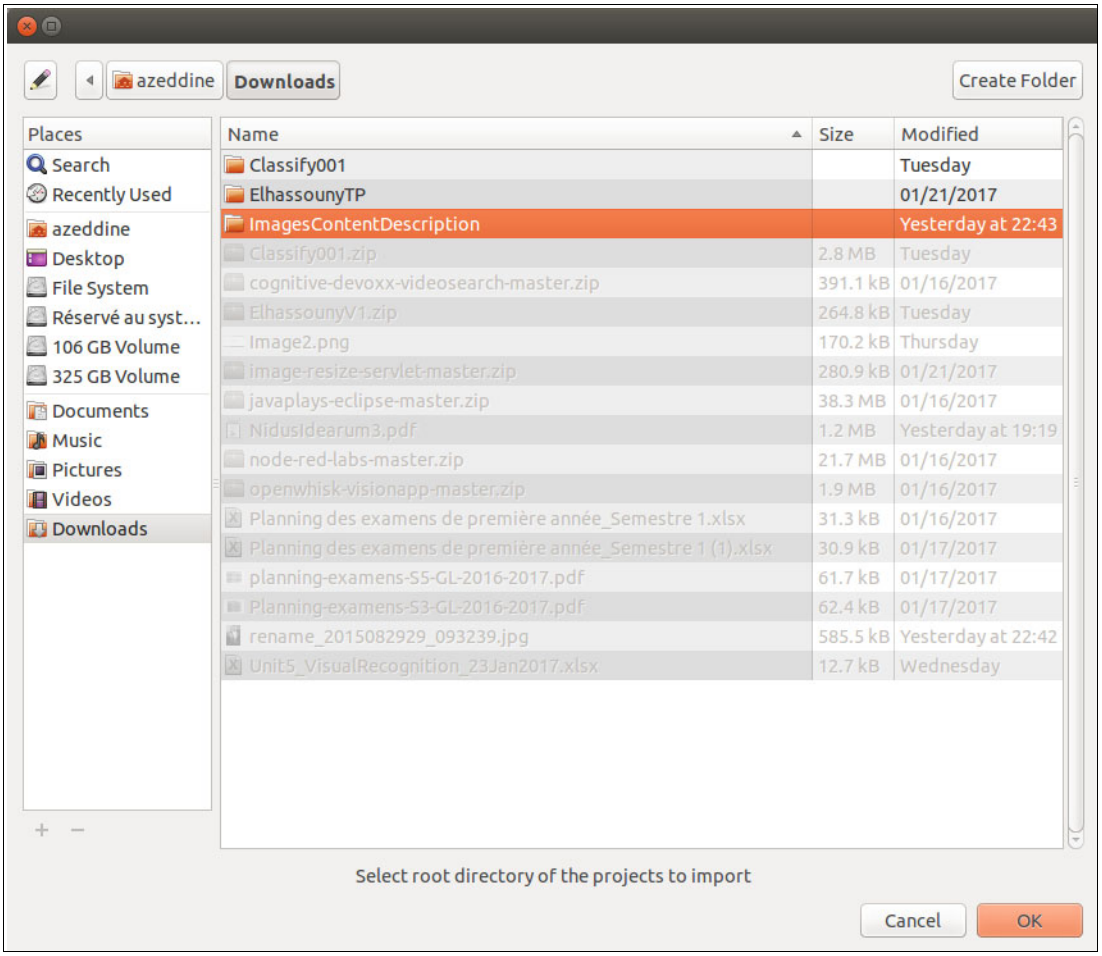
Working sets

□ Add project to working sets

Working sets:

Figure 3-10 Select root directory

5. Find and select the **ImageContentDescription** folder (Figure 3-11), and then click **OK**.

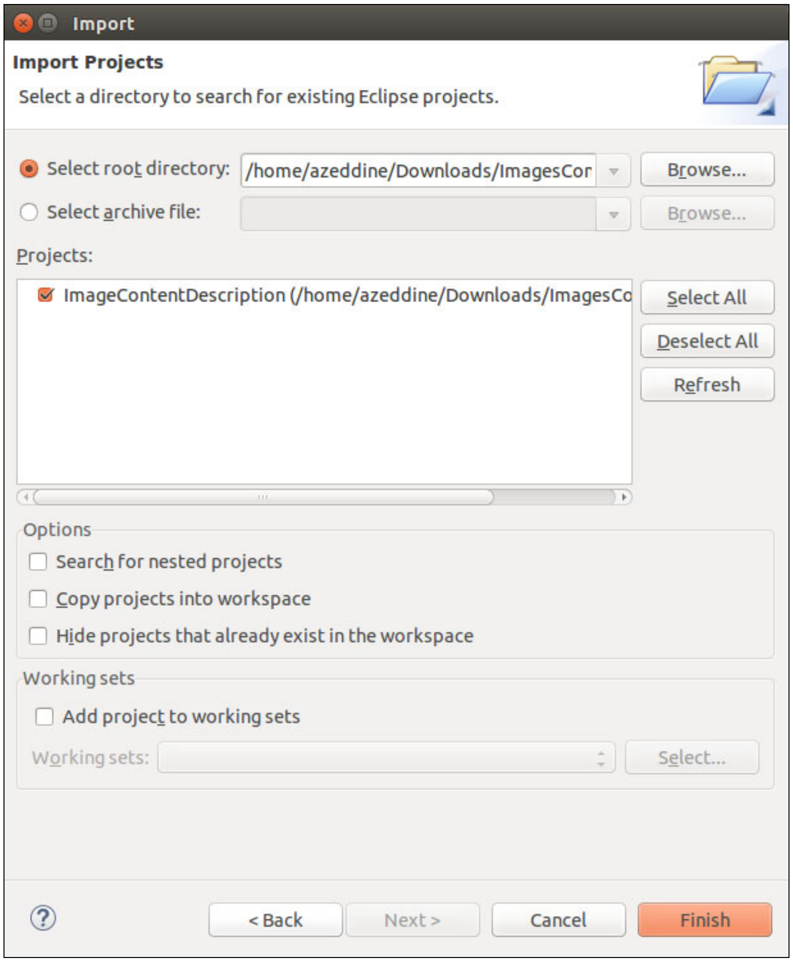


Name	Size	Modified
Classify001		Tuesday
ElhassounyTP		01/21/2017
ImagesContentDescription		Yesterday at 22:43
Classify001.zip	2.8 MB	Tuesday
cognitive-devoxx-videosearch-master.zip	391.1 kB	01/16/2017
ElhassounyV1.zip	264.8 kB	Tuesday
Image2.png	170.2 kB	Thursday
image-resize-servlet-master.zip	280.9 kB	01/21/2017
javaplays-eclipse-master.zip	38.3 MB	01/16/2017
NidusIdearum3.pdf	1.2 MB	Yesterday at 19:19
node-red-labs-master.zip	21.7 MB	01/16/2017
openwhisk-visionapp-master.zip	1.9 MB	01/16/2017
Planning des examens de première année_Semestre 1.xlsx	31.3 kB	01/16/2017
Planning des examens de première année_Semestre 1 (1).xlsx	30.9 kB	01/17/2017
planning-examens-S5-GL-2016-2017.pdf	61.7 kB	01/17/2017
Planning-examens-S3-GL-2016-2017.pdf	62.4 kB	01/17/2017
rename_2015082929_093239.jpg	585.5 kB	Yesterday at 22:42
Unit5_VisualRecognition_23Jan2017.xlsx	12.7 kB	Wednesday

Select root directory of the projects to import

Figure 3-11 Navigation window to import project

6. Under Projects, select the **ImageContentDescription** check box, deselect any other check boxes, and click **Finish** (Figure 3-12).



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: /home/azeddine/Downloads/ImagesCor

○ Select archive file:

Projects:

☑ ImageContentDescription (/home/azeddine/Downloads/ImagesCo

Options

☐ Search for nested projects

☐ Copy projects into workspace

☐ Hide projects that already exist in the workspace

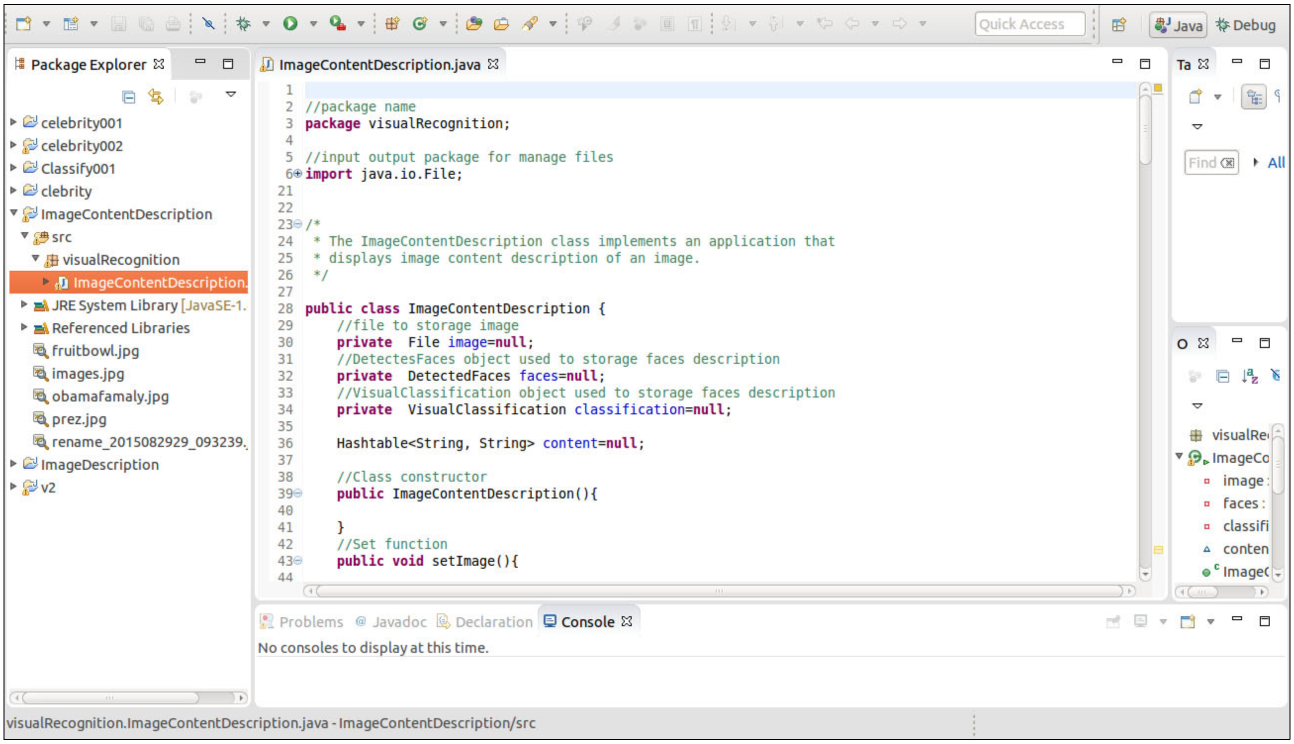
Working sets

☐ Add project to working sets

Working sets:

Figure 3-12 Last import project dialog

7. Verify that the `ImageContentDescription` project folder is imported to Eclipse Package Explorer (Figure 3-13) and explore its structure (for more details, see the `README.txt` file).



```

//package name
package visualRecognition;

//input output package for manage files
import java.io.File;


/*
 * The ImageContentDescription class implements an application that
 * displays image content description of an image.
 */

public class ImageContentDescription {
    //file to storage image
    private  File image=null;
    //DetectesFaces object used to storage faces description
    private  DetectedFaces faces=null;
    //VisualClassification object used to storage faces description
    private  VisualClassification classification=null;

    Hashtable<String, String> content=null;

    //Class constructor
    public ImageContentDescription(){

    }
    //Set function
    public void setImage(){

```

```
No consoles to display at this time.
```

visualRecognition.ImageContentDescription.java - ImageContentDescription/src

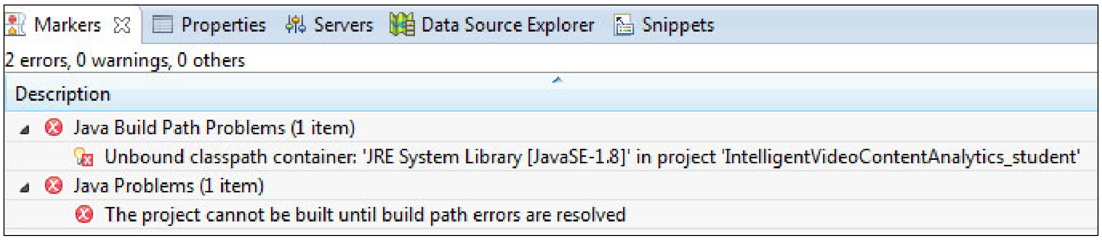
Figure 3-13 Eclipse Package Explorer dialog

3.3.4 Importing Watson Java SDK

You might notice some errors when you import the source code. Correcting those errors requires adding an extra dependency and libraries.

Fix Java problems

Figure 3-14 shows Java problems that you might see.



2 errors, 0 warnings, 0 others

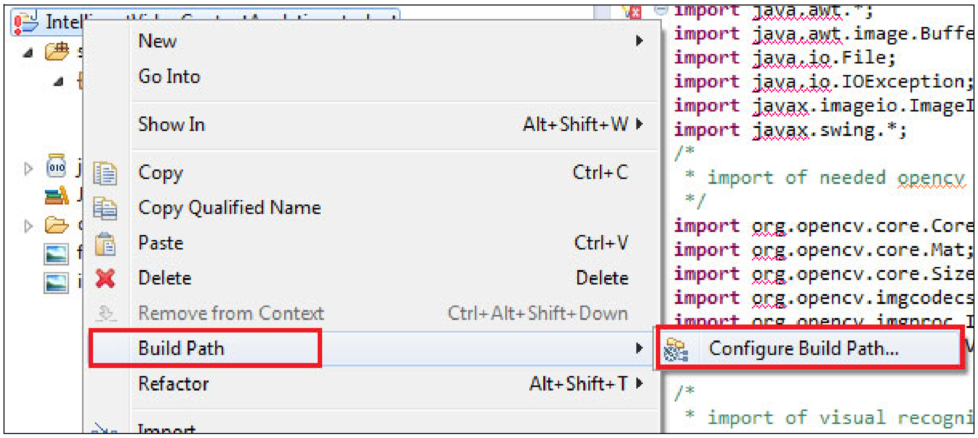
Description

- Java Build Path Problems (1 item)
  - Unbound classpath container: 'JRE System Library [JavaSE-1.8]' in project 'IntelligentVideoContentAnalytics_student'
- Java Problems (1 item)
  - The project cannot be built until build path errors are resolved

Figure 3-14 Java problems

To correct the problems, complete these steps:

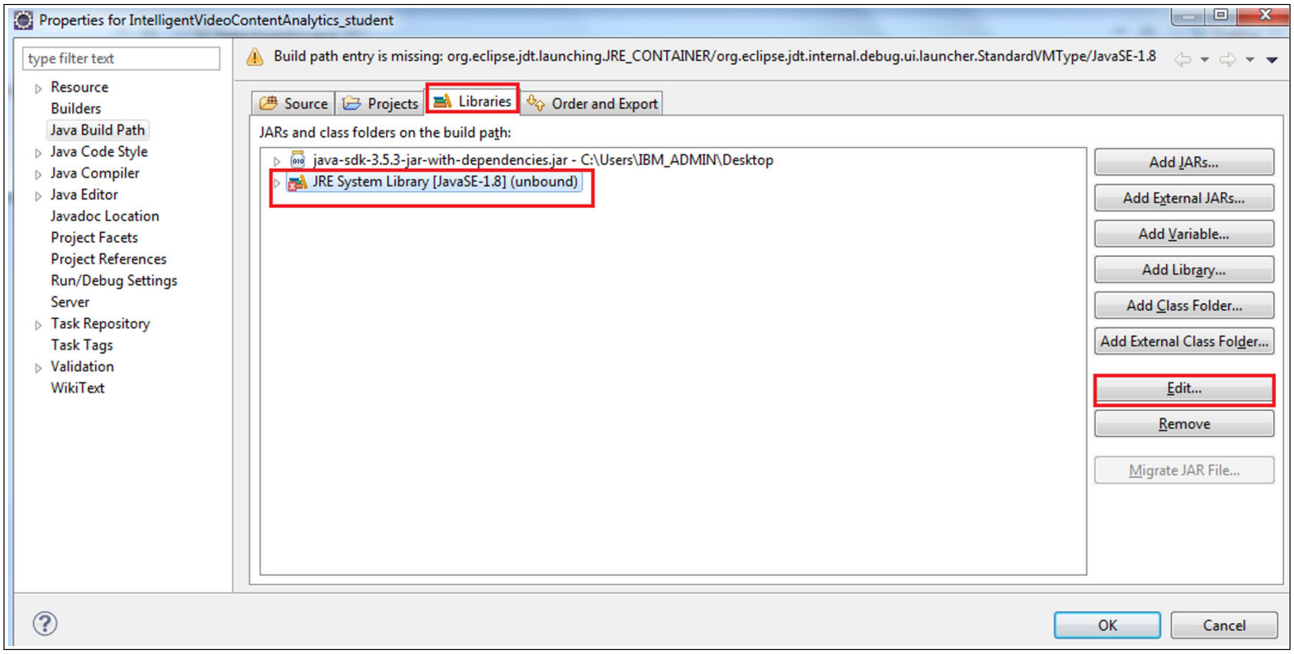
1. Right-click the **ImageContentDescription** project, and select **Build Path** → **Configure Build Path** (Figure 3-15).



```
import java.awt.*;
import java.awt.image.Buffe
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageI
import javax.swing.*;
/*
 * import of needed opencv
 */
import org.opencv.core.Core
import org.opencv.core.Mat;
import org.opencv.core.Size
import org.opencv.imgcodecs
import org.opencv.imgproc.I

/*
 * import of visual recogni
```

Figure 3-15 Configure Build Path

2. Select the **Libraries** tab, click the library showing the error, and click **Edit** (Figure 3-16).



Properties for IntelligentVideoContentAnalytics_student

Build path entry is missing: org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/JavaSE-1.8

JARs and class folders on the build path:

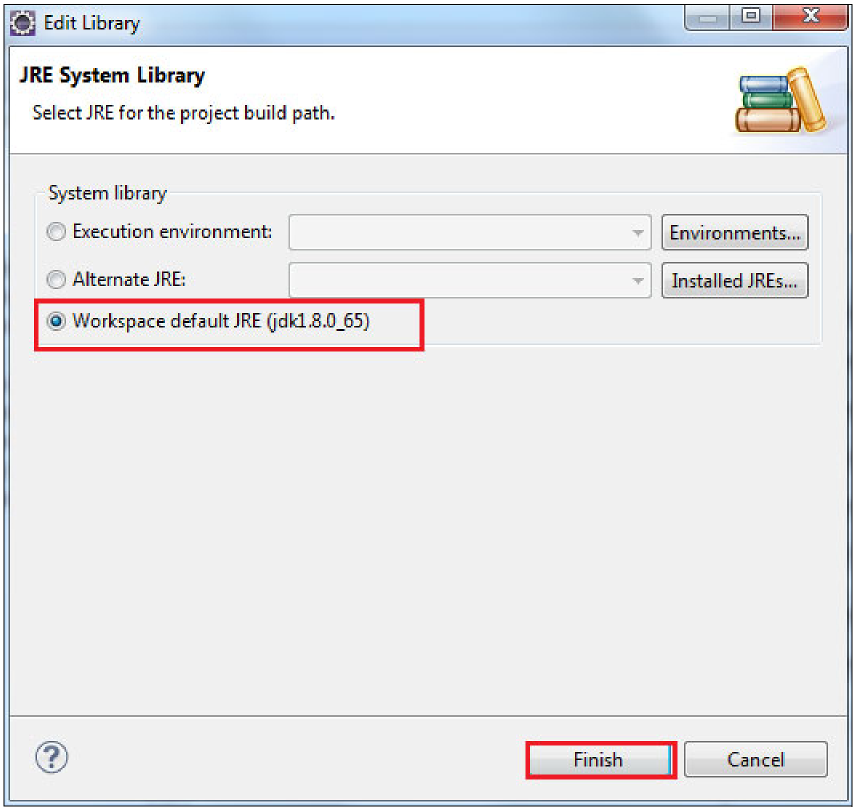
- java-sdk-3.5.3-jar-with-dependencies.jar - C:\Users\IBM_ADMIN\Desktop
- JRE System Library [JavaSE-1.8] (unbound)

Figure 3-16 Select the library showing an error to edit

3. Do one of the following steps:

   – If *no* default JRE was previously defined: Skip to step 4 on page 50.

   – If a default JRE was previously defined: Select **Workspace default JRE**, and click **Finish** (Figure 3-17). You can now skip to “Add Watson Java SDK with dependencies to your project” on page 54.



Edit Library

JRE System Library

Select JRE for the project build path.

System library

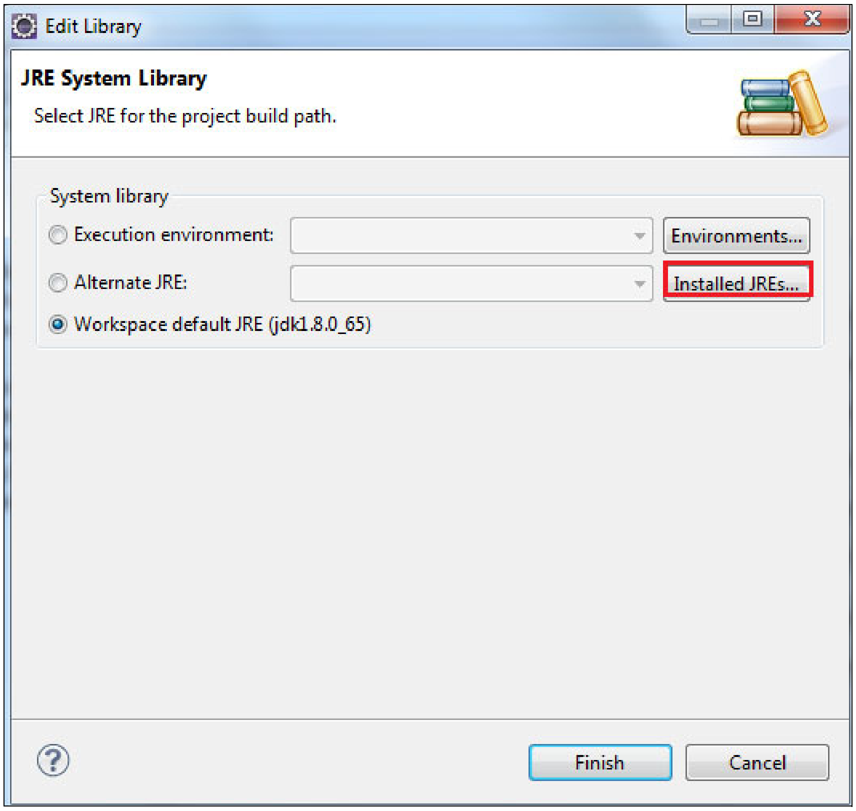
○ Execution environment:

○ Alternate JRE:

● Workspace default JRE (jdk1.8.0_65)

Figure 3-17 Select Workspace default JRE, if one was previously defined

4. This step through step 9 on page 54 are needed *only* if no default JRE was installed previously. Click **Installed JREs** (Figure 3-18).



JRE System Library

Select JRE for the project build path.

System library

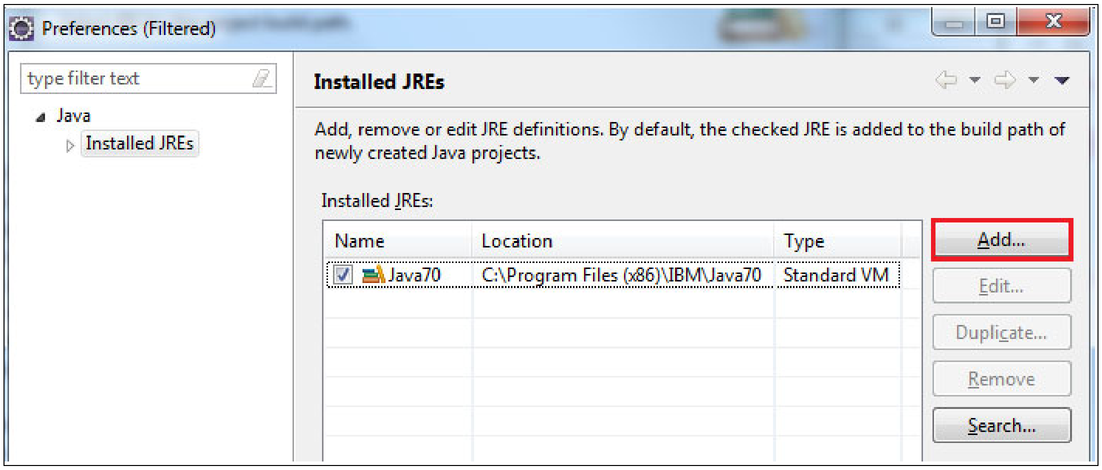
○ Execution environment:

○ Alternate JRE:

● Workspace default JRE (jdk1.8.0_65)

Figure 3-18 Installed JREs

5. Click **Add** (Figure 3-19).



Installed JREs

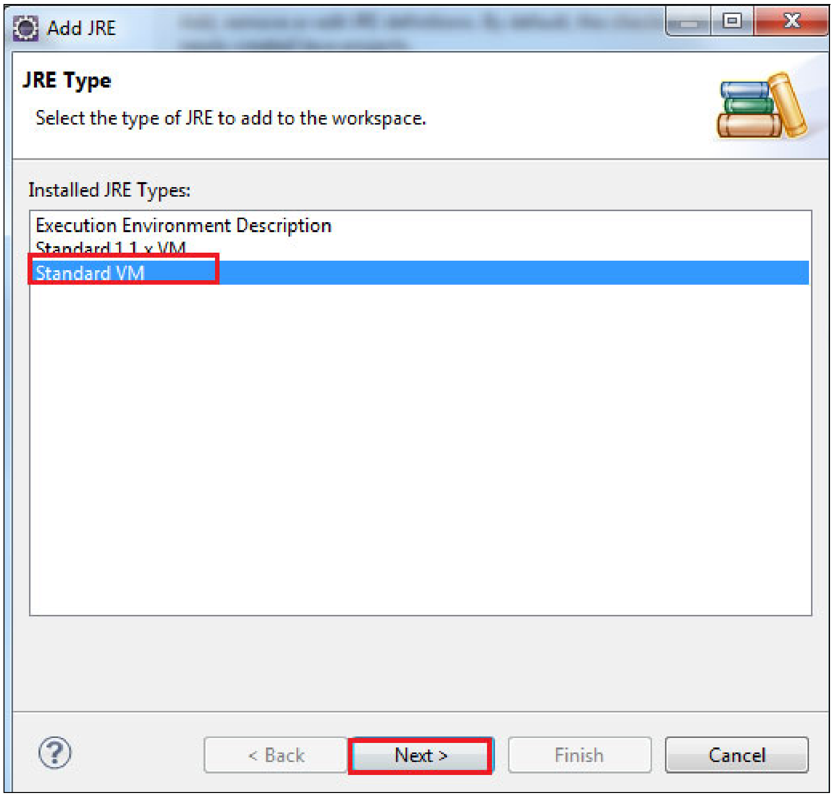
Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

Name	Location	Type
☑ Java70	C:\Program Files (x86)\IBM\Java70	Standard VM

Figure 3-19 Add a JRE definition

6. Select **Standard VM** and click **Next** (Figure 3-20).



Add JRE

JRE Type

Select the type of JRE to add to the workspace.

Installed JRE Types:

Execution Environment Description
Standard 1.1.x VM
Standard VM

Figure 3-20 Standard VM installed JRE type

7. Click **Directory**, select a JDK installation path, and click **OK** (Figure 3-21).

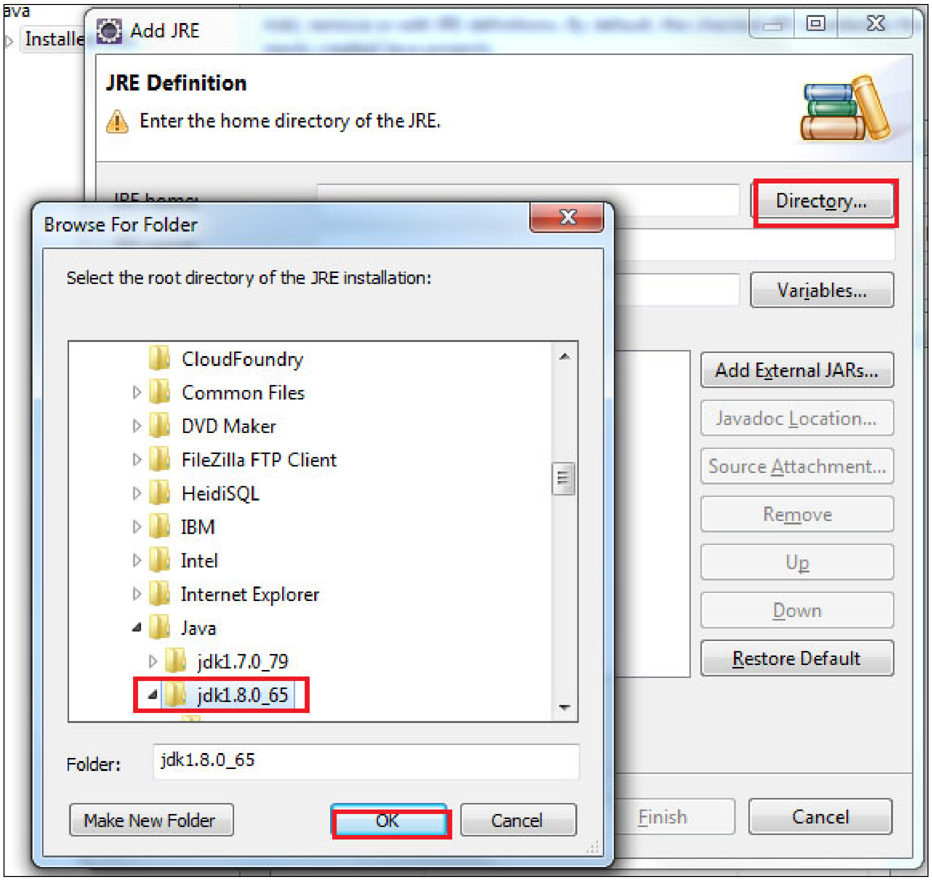
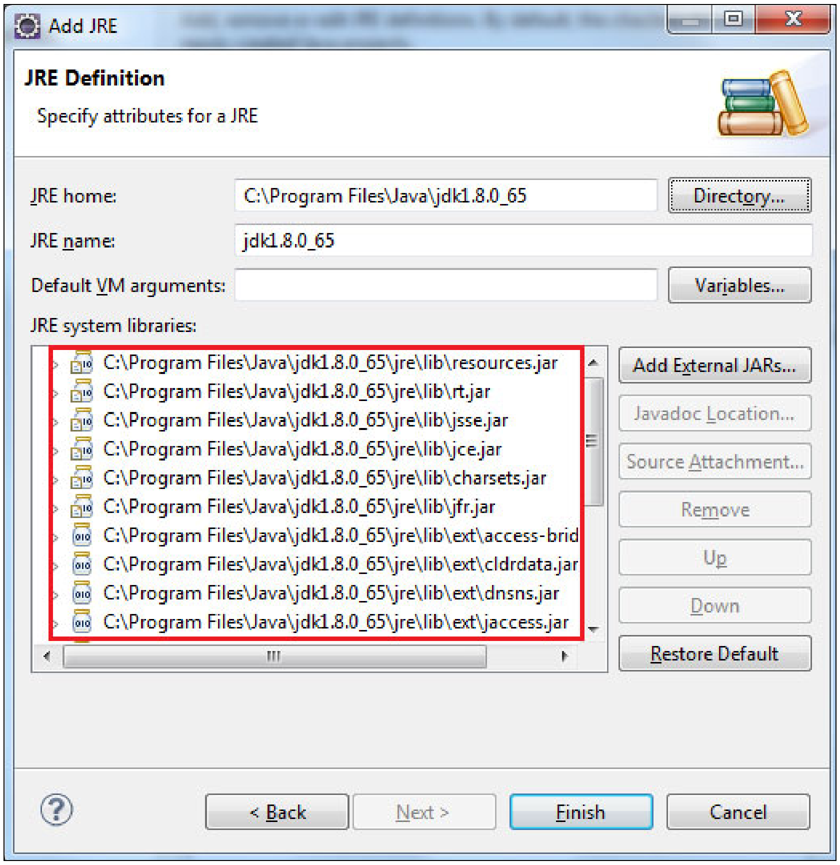


Figure 3-21 Select root directory of JRE installation

8. Your panel should look similar to the one shown in Figure 3-22. Click **Finish**.



Add JRE

JRE Definition

Specify attributes for a JRE

JRE home: C:\Program Files\Java\jdk1.8.0_65

JRE name: jdk1.8.0_65

Default VM arguments:

JRE system libraries:

C:\Program Files\Java\jdk1.8.0_65\jre\lib\resources.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\rt.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\jsse.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\jce.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\charsets.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\jfr.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\ext\access-brid
C:\Program Files\Java\jdk1.8.0_65\jre\lib\ext\cldrdata.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\ext\dnsns.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\ext\jaccess.jar

Figure 3-22 Sample JRE system libraries

9. Select **Workspace default JRE**, and click **Finish** (Figure 3-23).

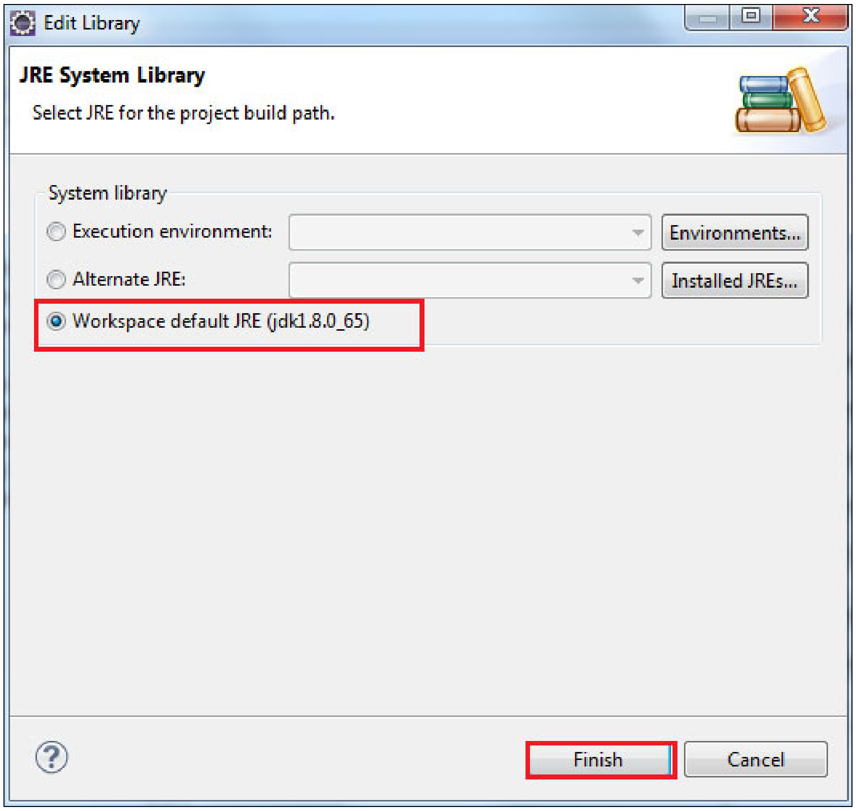


Figure 3-23 Select Workspace default JRE

Add Watson Java SDK with dependencies to your project

Complete the following steps:

1. Download the Watson Java SDK dependencies JAR (with dependencies) files:

   `https://github.com/watson-developer-cloud/java-sdk/releases`

2. Scroll to the **Downloads** section and click **java-sdk-3.7.0-jar-with-dependencies.jar** (Figure 3-24).

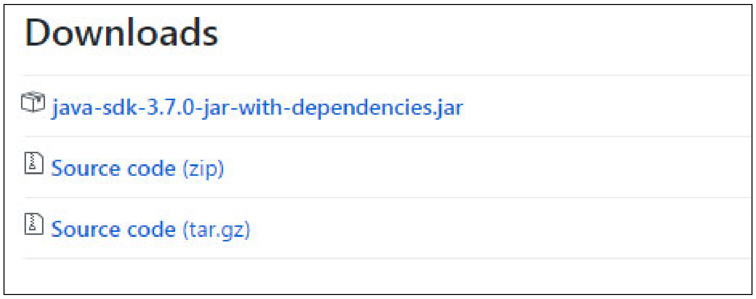


Figure 3-24 Download Watson Java SDK

3. After the JAR file is downloaded, open Eclipse, right-click the project name, and select **Build Path** → **Configure Build Path** (Figure 3-25).

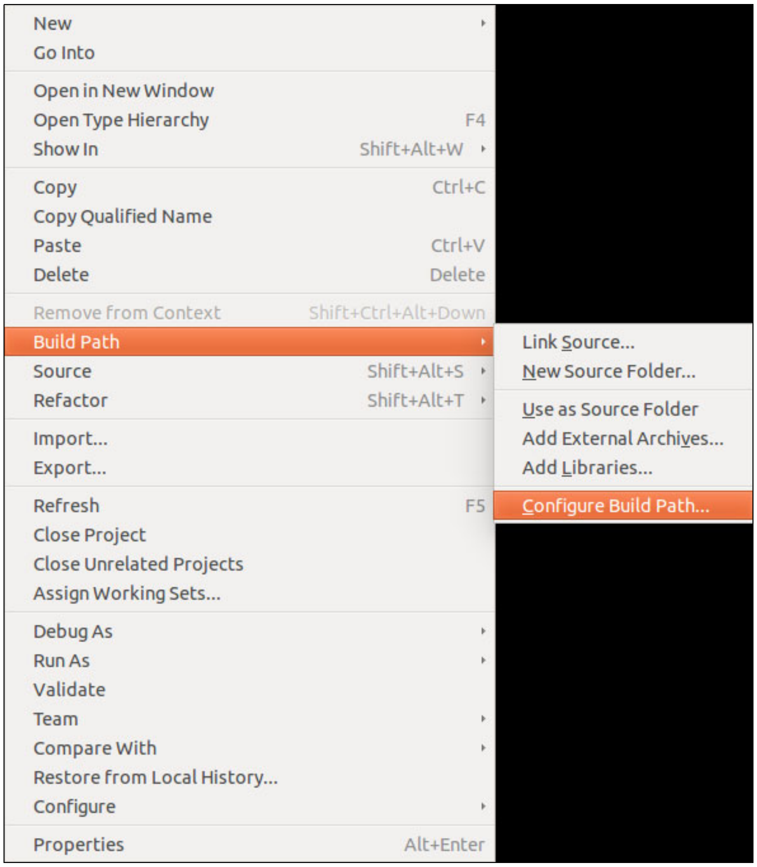
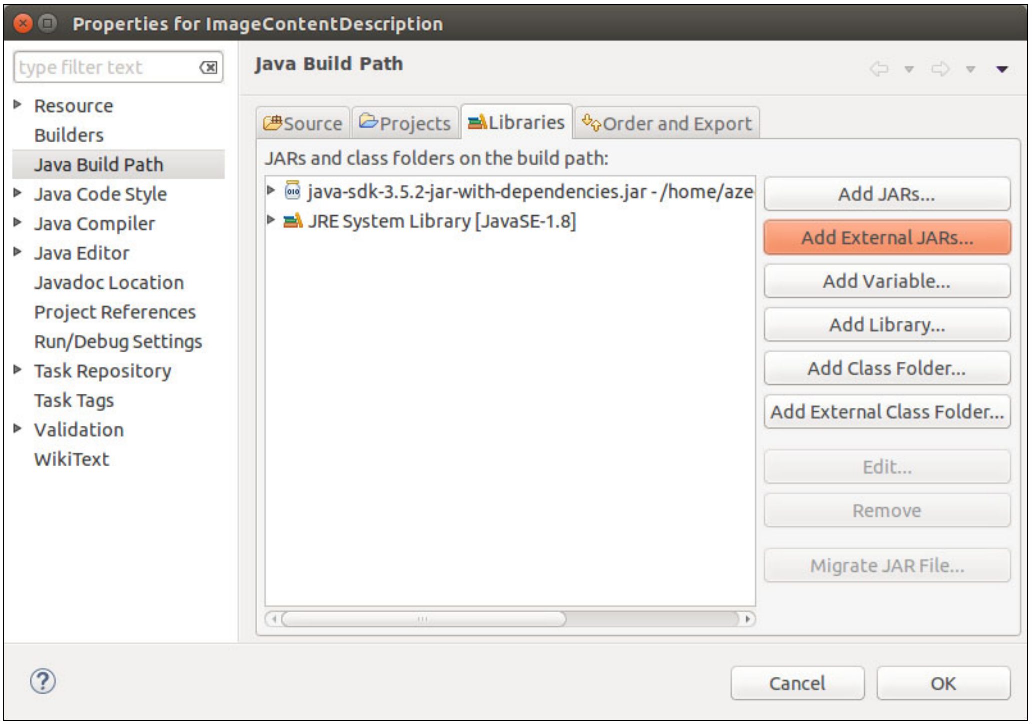


Figure 3-25 Configure Build Path

4. Open the **Libraries** tab, and then click **Add External JARs** (Figure 3-26).



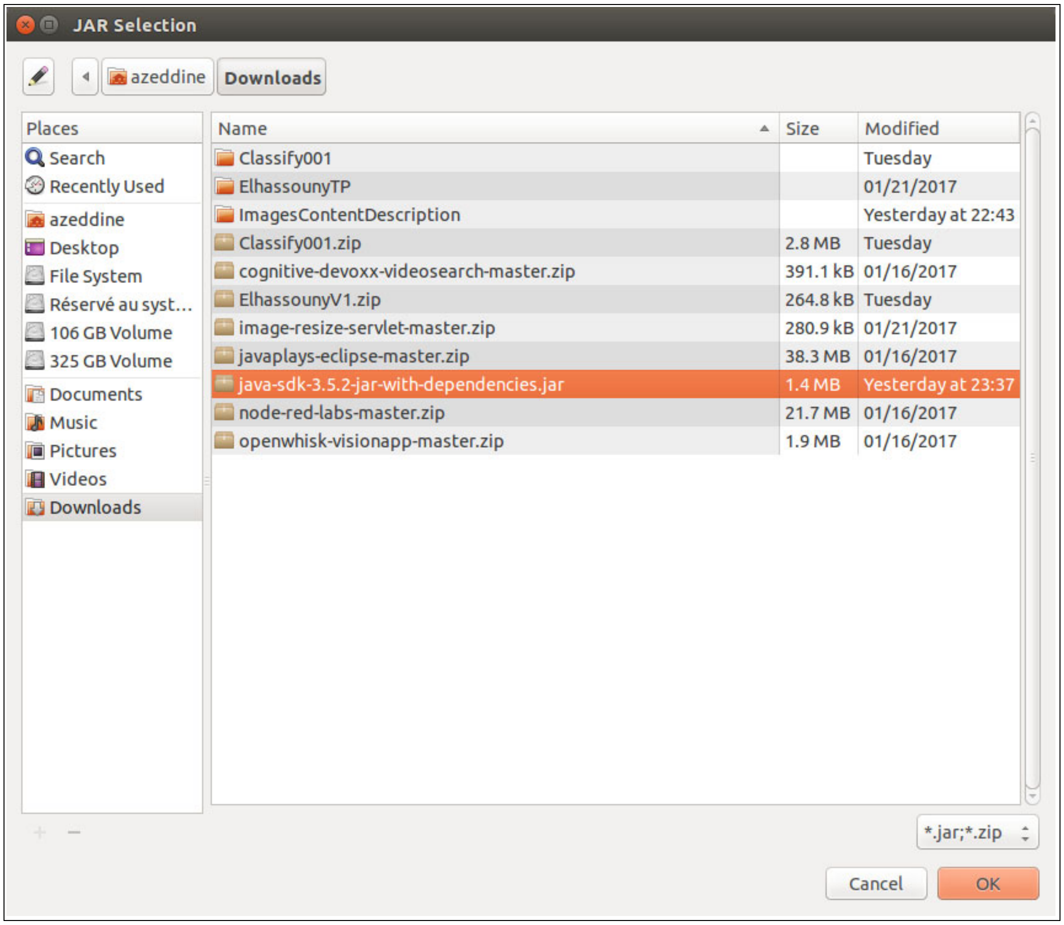
JARs and class folders on the build path:

- java-sdk-3.5.2-jar-with-dependencies.jar - /home/aze
- JRE System Library [JavaSE-1.8]

Figure 3-26 Java Built Path dialog

5. Navigate to the JAR file (`java-sdk-3.5.2-jar-with-dependencies.jar`), select it, and then click **OK** (Figure 3-27).

Note: The JAR file name (`java-sdk-x.x.x-jar-with-dependencies.jar`) will vary depending on the version available when you download it.

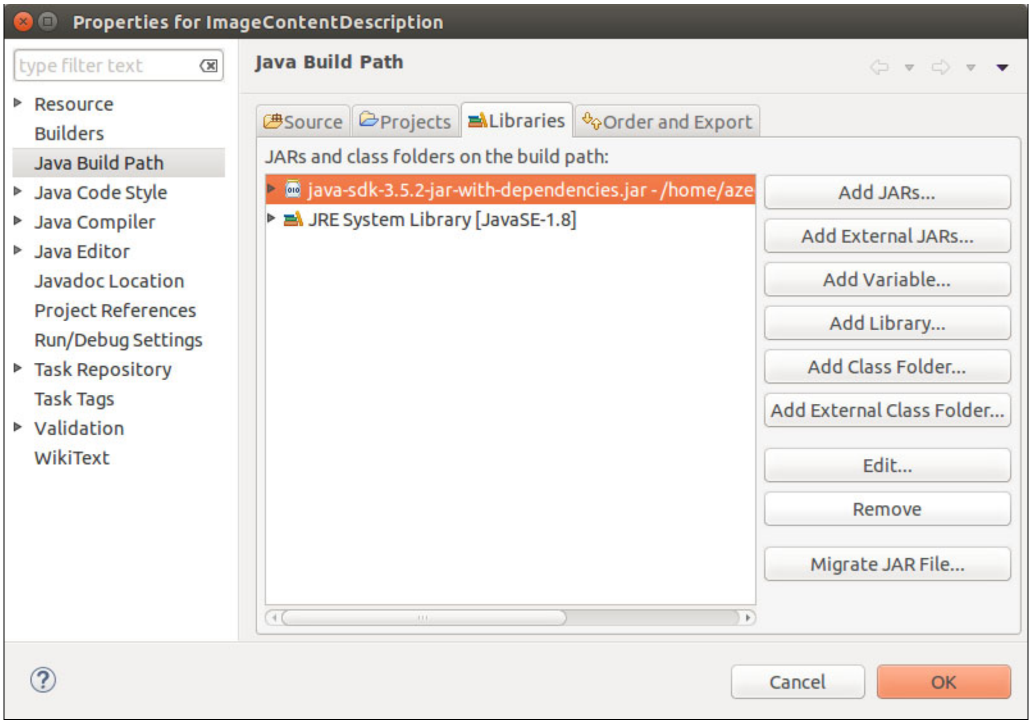


Name	Size	Modified
Classify001		Tuesday
ElhassounyTP		01/21/2017
ImagesContentDescription		Yesterday at 22:43
Classify001.zip	2.8 MB	Tuesday
cognitive-devoxx-videosearch-master.zip	391.1 kB	01/16/2017
ElhassounyV1.zip	264.8 kB	Tuesday
image-resize-servlet-master.zip	280.9 kB	01/21/2017
javaplays-eclipse-master.zip	38.3 MB	01/16/2017
java-sdk-3.5.2-jar-with-dependencies.jar	1.4 MB	Yesterday at 23:37
node-red-labs-master.zip	21.7 MB	01/16/2017
openwhisk-visionapp-master.zip	1.9 MB	01/16/2017

.jar;.zip

Figure 3-27   Select the Java SDK JAR file

6. Check that the JAR file is added to your project and click **OK** (Figure 3-28).



Properties for ImageContentDescription

Java Build Path

JARs and class folders on the build path:

- java-sdk-3.5.2-jar-with-dependencies.jar - /home/aze
- JRE System Library [JavaSE-1.8]

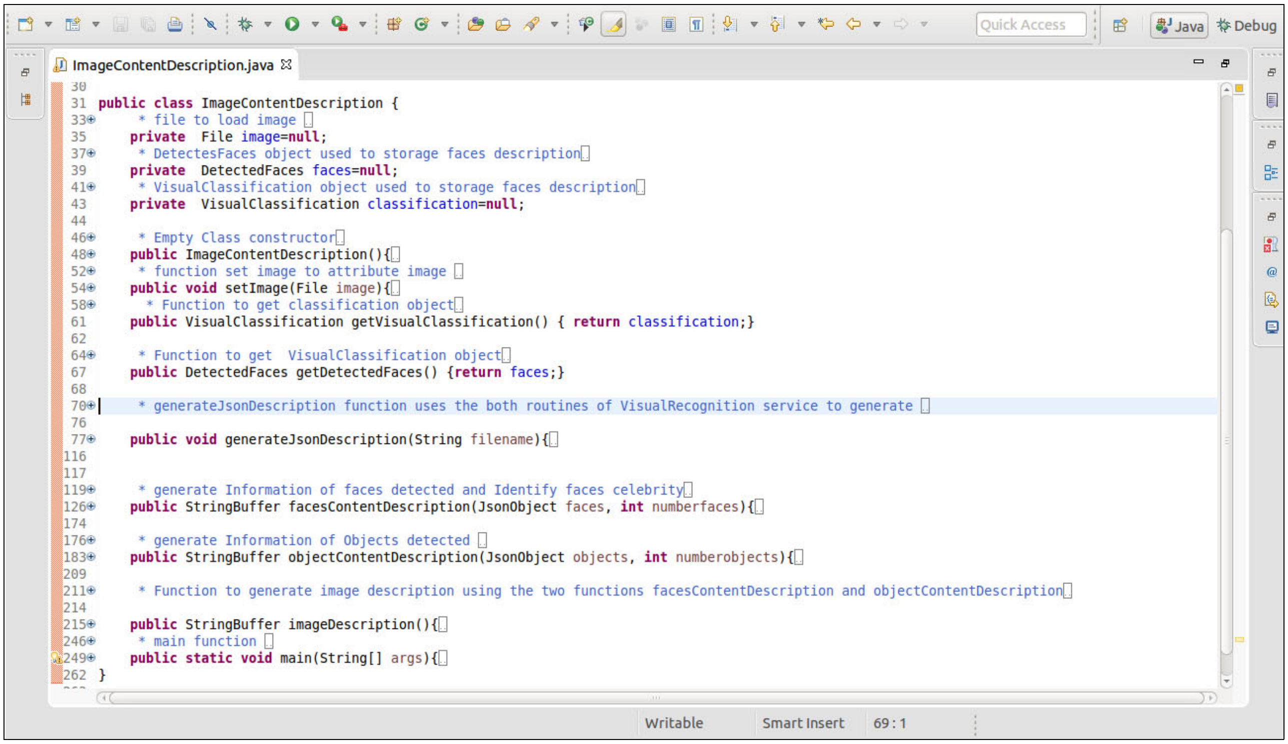
Figure 3-28 Window to check the addition of Java SDK

7. Now that you added the required library, verify that no Java errors exist in the imported project.

3.3.5 Exploring the sample code provided with the use case

Now that you imported the project and resolved the import errors, you can use the Java editor to explore and understand the code.

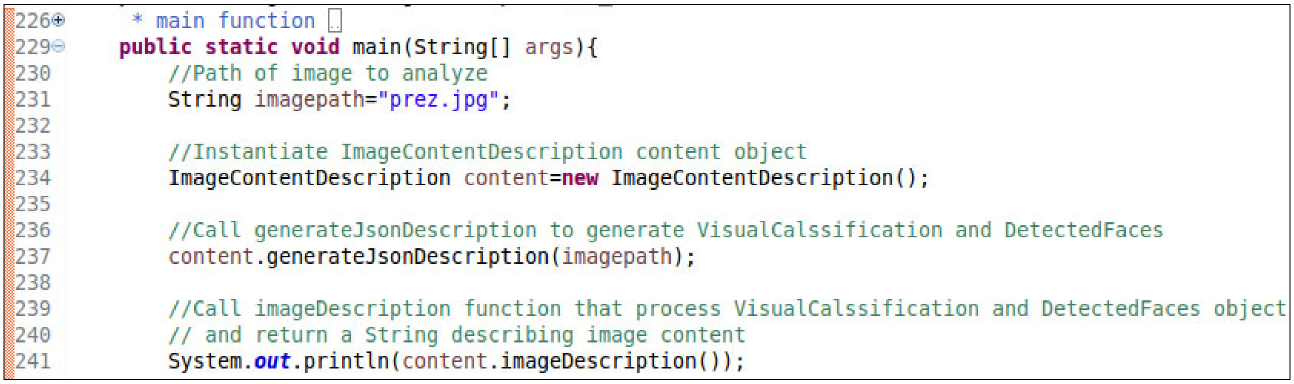
Figure 3-29 shows an overview of the `ImageContentDescription` program.



```
public class ImageContentDescription {
     * file to load image □
    private  File image=null;
     * DetectesFaces object used to storage faces description□
    private  DetectedFaces faces=null;
     * VisualClassification object used to storage faces description□
    private  VisualClassification classification=null;

     * Empty Class constructor□
    public ImageContentDescription(){□
     * function set image to attribute image □
    public void setImage(File image){□
      * Function to get classification object□
    public VisualClassification getVisualClassification() { return classification;}

     * Function to get  VisualClassification object□
    public DetectedFaces getDetectedFaces() {return faces;}

     * generateJsonDescription function uses the both routines of VisualRecognition service to generate □

    public void generateJsonDescription(String filename){□


     * generate Information of faces detected and Identify faces celebrity□
    public StringBuffer facesContentDescription(JsonObject faces, int numberfaces){□

     * generate Information of Objects detected □
    public StringBuffer objectContentDescription(JsonObject objects, int numberobjects){□

     * Function to generate image description using the two functions facesContentDescription and objectContentDescription□

    public StringBuffer imageDescription(){□
     * main function □
    public static void main(String[] args){□
}
```

Figure 3-29 ImageContentDescription sample program snippet overview

As you know, the starting point of execution of a stand-alone Java program is the `main` method (Figure 3-30).

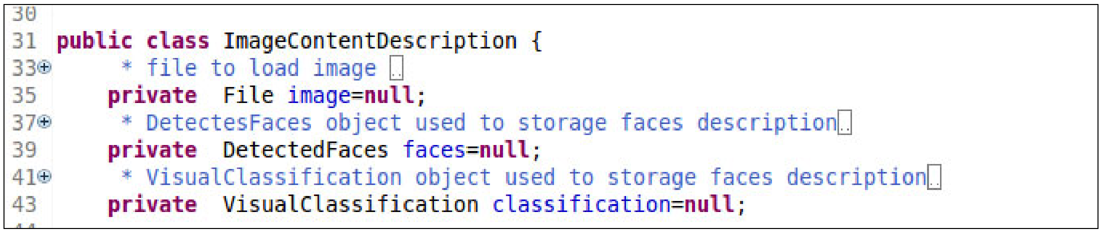


```
     * main function □
    public static void main(String[] args){
        //Path of image to analyze
        String imagepath="prez.jpg";

        //Instantiate ImageContentDescription content object
        ImageContentDescription content=new ImageContentDescription();

        //Call generateJsonDescription to generate VisualCalssification and DetectedFaces
        content.generateJsonDescription(imagepath);

        //Call imageDescription function that process VisualCalssification and DetectedFaces object
        // and return a String describing image content
        System.out.println(content.imageDescription());
```

Figure 3-30 The main method source code snippet

The `main` method shows the instantiation of the `ImageContentDescription` class, which is the only class in this project (Figure 3-31). This class declares three attributes:

- An `image` variable: Holds the image (`file` object) that will be analyzed.
- A `faces` variable: Holds the Watson `DetectedFaces` object.
- A `classification` variable: Holds the Watson `VisualCalssification` object.



```
public class ImageContentDescription {
     * file to load image 
    private  File image=null;
     * DetectesFaces object used to storage faces description
    private  DetectedFaces faces=null;
     * VisualClassification object used to storage faces description
    private  VisualClassification classification=null;
```

Figure 3-31   Class ImageContentDescription

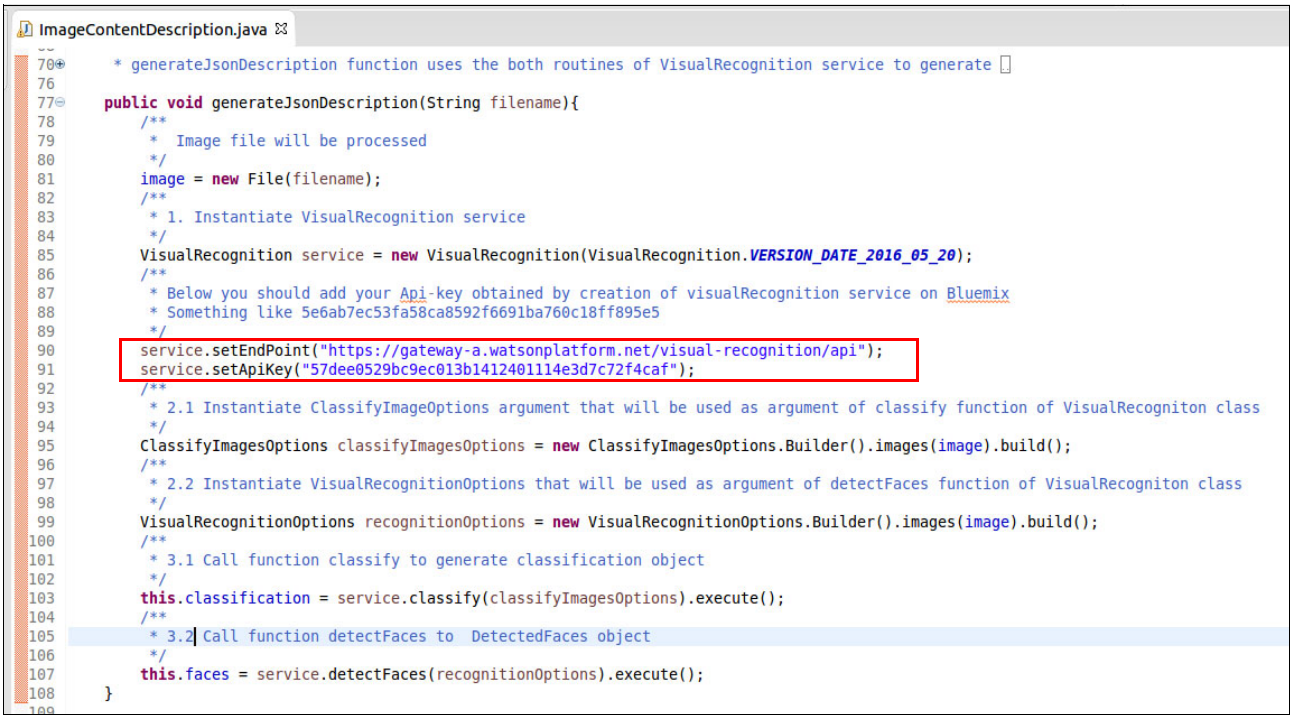
The generateJsonDescription method

After the `content` variable is initialized with the instantiated `ImageContentDescription` object, the `generateJsonDescription(imagepath)` method is called (Figure 3-30 on page 59).

The `generateJsonDescription(imagepath)` method accepts the image path as an argument and it does the following steps:

1. Instantiates a Watson VisualRecognition service with the credentials you obtained in 3.3.1, "Creating a Visual Recognition service instance" on page 41.
2. Creates the `VisualClassification` object.
3. Creates the `DetectedFaces` object.

The source code for `generateJsonDescription` is shown in Figure 3-32. The next sections describe this code. The highlighted code (lines 90 and 91) are described later.



```
70      * generateJsonDescription function uses the both routines of VisualRecognition service to generate
76
77     public void generateJsonDescription(String filename){
78         /**
79          *  Image file will be processed
80          */
81         image = new File(filename);
82         /**
83          * 1. Instantiate VisualRecognition service
84          */
85         VisualRecognition service = new VisualRecognition(VisualRecognition.VERSION_DATE_2016_05_20);
86         /**
87          * Below you should add your Api-key obtained by creation of visualRecognition service on Bluemix
88          * Something like 5e6ab7ec53fa58ca8592f6691ba760c18ff895e5
89          */
90         service.setEndPoint("https://gateway-a.watsonplatform.net/visual-recognition/api");
91         service.setApiKey("57dee0529bc9ec013b1412401114e3d7c72f4caf");
92         /**
93          * 2.1 Instantiate ClassifyImageOptions argument that will be used as argument of classify function of VisualRecogniton class
94          */
95         ClassifyImagesOptions classifyImagesOptions = new ClassifyImagesOptions.Builder().images(image).build();
96         /**
97          * 2.2 Instantiate VisualRecognitionOptions that will be used as argument of detectFaces function of VisualRecogniton class
98          */
99         VisualRecognitionOptions recognitionOptions = new VisualRecognitionOptions.Builder().images(image).build();
100        /**
101         * 3.1 Call function classify to generate classification object
102         */
103        this.classification = service.classify(classifyImagesOptions).execute();
104        /**
105         * 3.2 Call function detectFaces to  DetectedFaces object
106         */
107        this.faces = service.detectFaces(recognitionOptions).execute();
108    }
```

Figure 3-32   The generateJsonDescription source code

Instantiate the VisualRecognition service

As Figure 3-32 on page 60 shows, the first instruction uses `VisualRecognition` to instantiate a new Visual Recognition V3 service with an API key:

```
(VisualRecognition service = New VisualRecognition(String versionDate))
```

It also sets the API key (`setApiKey`) and the endpoint (`setEndPoint`) to the service created.

Now, you provide the values of `EndPoint` and `APIkey` with the information you copied previously; paste them in the selected places, as shown in lines 90 and 91 of the source code in Figure 3-32 on page 60.

Create the VisualClassification object

Consider this information about image classification code (instructions `2.1` and `3.1` in Figure 3-32 on page 60).

- To classify an object, call the classify() method (`service.classify(ClassifyImagesOptions`)) that accepts options (`ClassifyImagesOptions`) as arguments and returns a `VisualClassification` object. The `classify()` method of the `VisualRecognition` class analyzes the image and detects details of the objects within the image.
- To create the new options for the new image, you instantiate a builder (`ClassifyImagesOptions.Builder()`), call the `images`() method to set the new image you want to classify; this method accepts an image file as a parameter and returns the builder.
- By the end, you call the `build` () method which returns the profile options (`ClassifyImagesOptions`).
- The `execute`() method, is used to execute the service which returns the `VisualClasification` object.

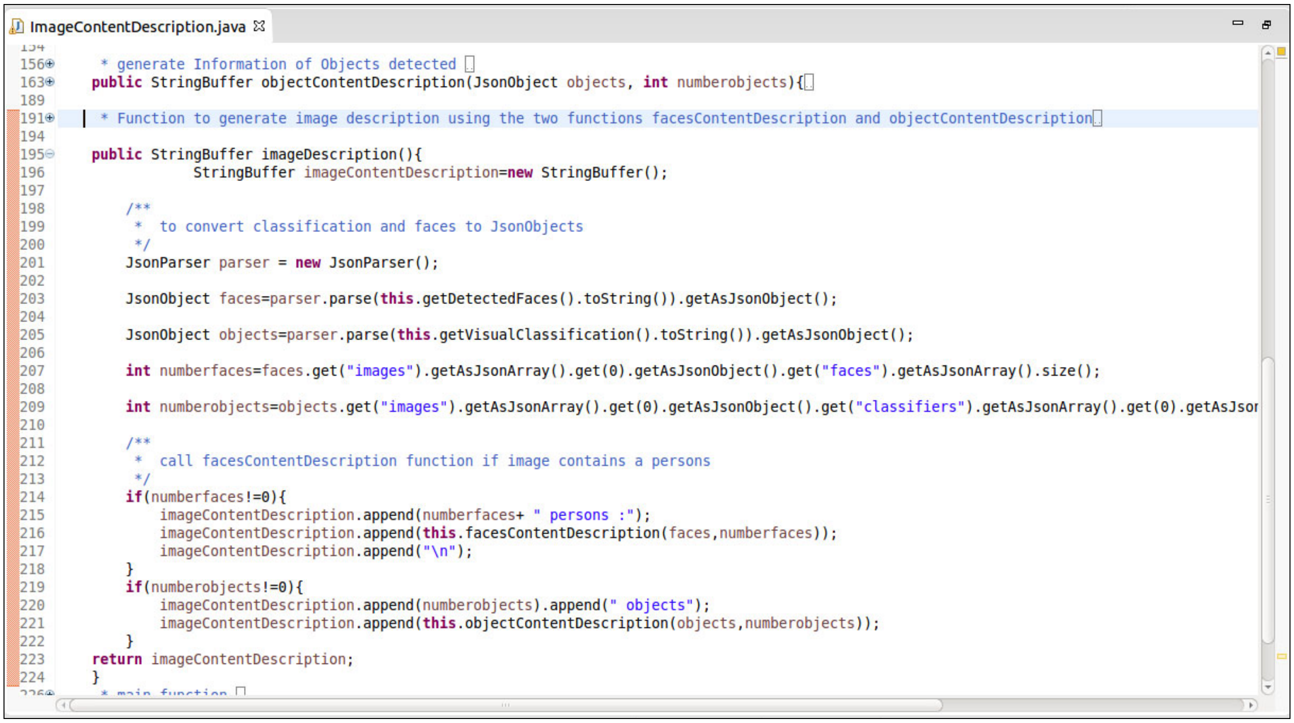
Create the DetectedFaces object

Consider this information about face detection code (instructions `2.2` and `3.2` in Figure 3-32 on page 60):

- To detect faces, call the `detectFaces`() method (`service.detectFaces(VisualRecognitionOptions)`) that accepts options (`VisualRecognitionOptions`) as argument and returns a `DetectedFaces` object. The `detectFaces`() method of the `VisualRecognition` class analyzes faces in images and gets data about them.
- To create the new options for the new image, instantiate a builder (`VisualRecognitionOptions.Builder()`), call the `images`() method to set the new image you want to analyze; this method accepts an image file as a parameter, and returns the builder.
- By the end, you call the `build` () method, which returns the profile options (`VisualRecognitionOptions`).
- The `execute()` method is used to execute the service which returns the `DetectedFaces` object.

The imageDescription method

The `imageDescription()` method processes the classification and faces attributes that were generated as described in "The generateJsonDescription method" on page 60. The `imageDescription()` method returns a string describing image content. Figure 3-33 shows the source code of the `imageDescription()` method.



```java
   * generate Information of Objects detected
  public StringBuffer objectContentDescription(JsonObject objects, int numberobjects){

   * Function to generate image description using the two functions facesContentDescription and objectContentDescription

  public StringBuffer imageDescription(){
              StringBuffer imageContentDescription=new StringBuffer();

      /**
       *  to convert classification and faces to JsonObjects
       */
      JsonParser parser = new JsonParser();

      JsonObject faces=parser.parse(this.getDetectedFaces().toString()).getAsJsonObject();

      JsonObject objects=parser.parse(this.getVisualClassification().toString()).getAsJsonObject();

      int numberfaces=faces.get("images").getAsJsonArray().get(0).getAsJsonObject().get("faces").getAsJsonArray().size();

      int numberobjects=objects.get("images").getAsJsonArray().get(0).getAsJsonObject().get("classifiers").getAsJsonArray().get(0).getAsJsor

      /**
       *  call facesContentDescription function if image contains a persons
       */
      if(numberfaces!=0){
          imageContentDescription.append(numberfaces+ " persons :");
          imageContentDescription.append(this.facesContentDescription(faces,numberfaces));
          imageContentDescription.append("\n");
      }
      if(numberobjects!=0){
          imageContentDescription.append(numberobjects).append(" objects");
          imageContentDescription.append(this.objectContentDescription(objects,numberobjects));
      }
  return imageContentDescription;
  }
   * main function
```

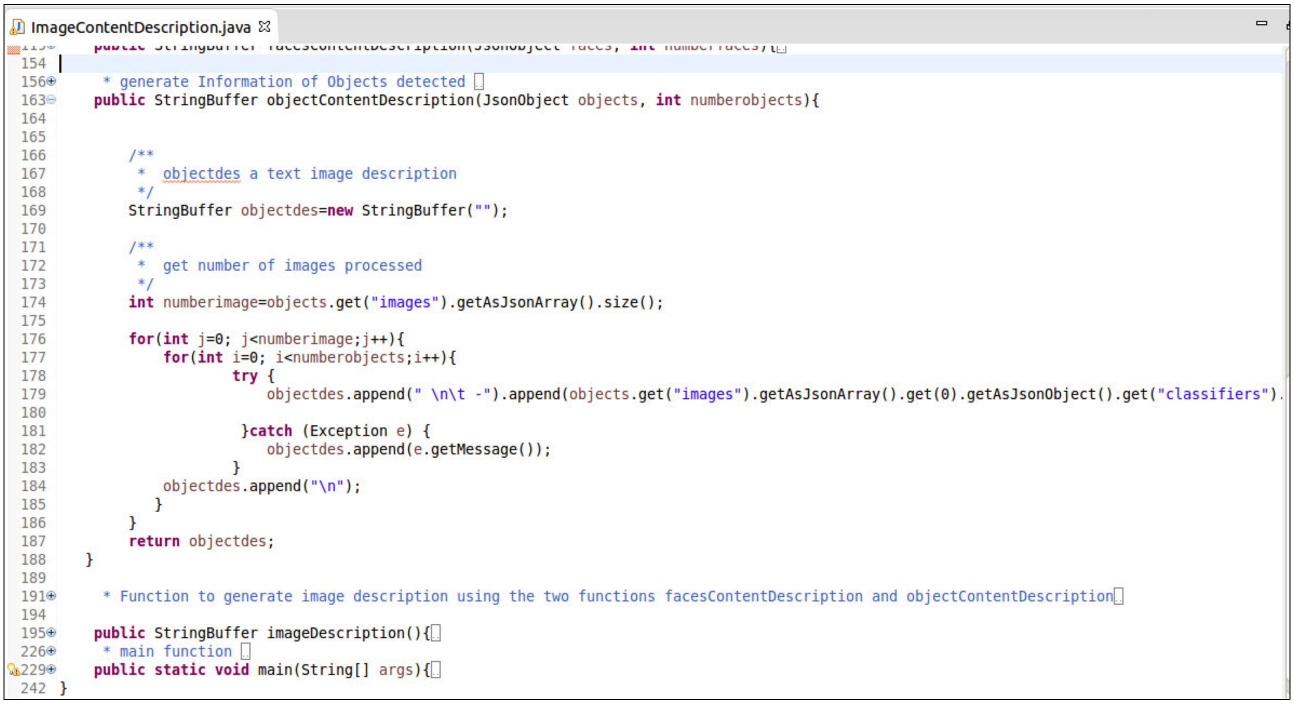
Figure 3-33   The imageDescription method source code

This method converts classification and faces attributes to JSON objects using the JSON Parser, processes its contents and does the following operations:

- Calls `objectContentDescription()` if one or more objects are in the image.
- Calls `facesContentDescription()` if one or more faces are in the image.

The objectContentDescription method

The `objectContentDescription()` method accepts detected objects in JSON format and the number of objects to process as arguments and returns a string describing the objects from the image. Figure 3-34 shows more details of this method source code.

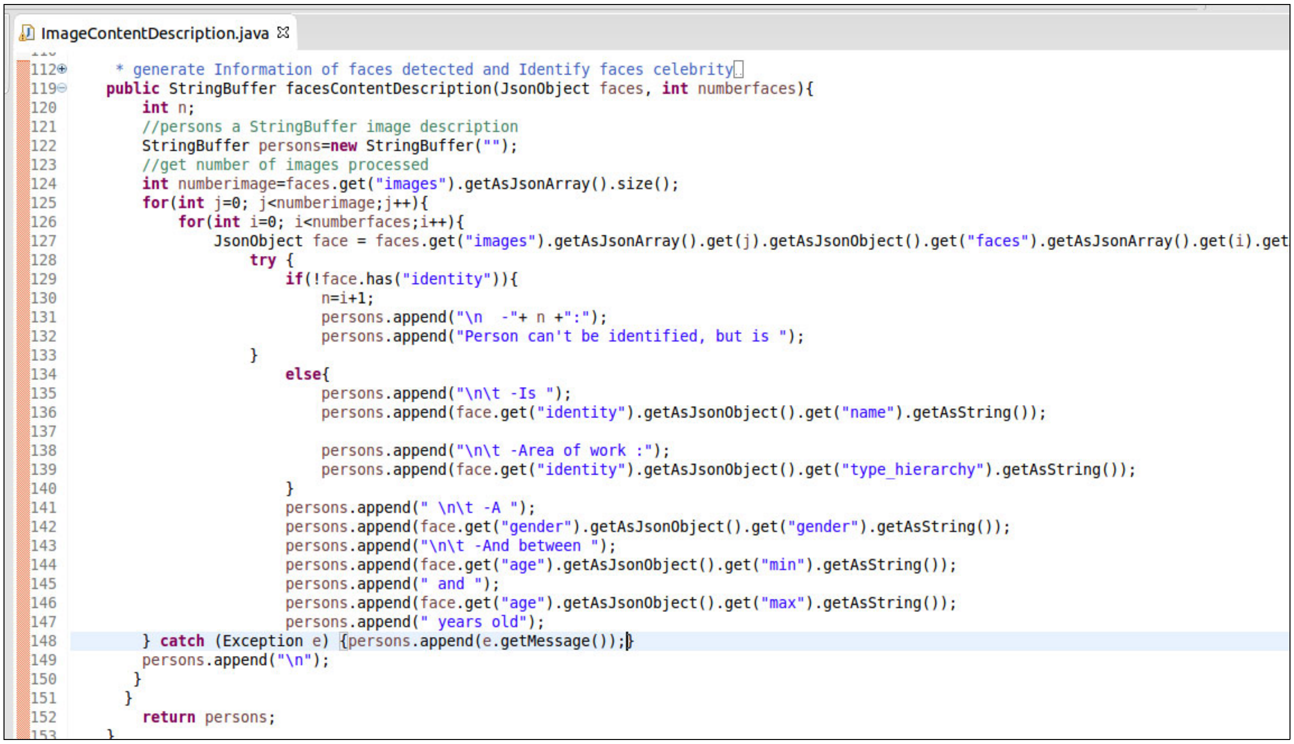


```
    public StringBuffer facesContentDescription(JsonObject faces, int numberfaces){□

     * generate Information of Objects detected □
    public StringBuffer objectContentDescription(JsonObject objects, int numberobjects){


        /**
         *  objectdes a text image description
         */
        StringBuffer objectdes=new StringBuffer("");

        /**
         *  get number of images processed
         */
        int numberimage=objects.get("images").getAsJsonArray().size();

        for(int j=0; j<numberimage;j++){
            for(int i=0; i<numberobjects;i++){
                    try {
                        objectdes.append(" \n\t -").append(objects.get("images").getAsJsonArray().get(0).getAsJsonObject().get("classifiers").

                     }catch (Exception e) {
                        objectdes.append(e.getMessage());
                    }
            objectdes.append("\n");
           }
        }
        return objectdes;
   }

     * Function to generate image description using the two functions facesContentDescription and objectContentDescription□

    public StringBuffer imageDescription(){□
     * main function □
    public static void main(String[] args){□
}
```

Figure 3-34 The objectContentDescription source code

The facesContentDescription method

The `facesContentDescription()` method (Figure 3-35) accepts detected faces as a JSON object and the number of faces to process as arguments and returns a string that describes the faces.



```
     * generate Information of faces detected and Identify faces celebrity
    public StringBuffer facesContentDescription(JsonObject faces, int numberfaces){
        int n;
        //persons a StringBuffer image description
        StringBuffer persons=new StringBuffer("");
        //get number of images processed
        int numberimage=faces.get("images").getAsJsonArray().size();
        for(int j=0; j<numberimage;j++){
            for(int i=0; i<numberfaces;i++){
                JsonObject face = faces.get("images").getAsJsonArray().get(j).getAsJsonObject().get("faces").getAsJsonArray().get(i).get
                    try {
                        if(!face.has("identity")){
                            n=i+1;
                            persons.append("\n  -"+ n +":");
                            persons.append("Person can't be identified, but is ");
                    }
                        else{
                            persons.append("\n\t -Is ");
                            persons.append(face.get("identity").getAsJsonObject().get("name").getAsString());

                            persons.append("\n\t -Area of work :");
                            persons.append(face.get("identity").getAsJsonObject().get("type_hierarchy").getAsString());
                        }
                        persons.append(" \n\t -A ");
                        persons.append(face.get("gender").getAsJsonObject().get("gender").getAsString());
                        persons.append("\n\t -And between ");
                        persons.append(face.get("age").getAsJsonObject().get("min").getAsString());
                        persons.append(" and ");
                        persons.append(face.get("age").getAsJsonObject().get("max").getAsString());
                        persons.append(" years old");
        } catch (Exception e) {persons.append(e.getMessage());}
        persons.append("\n");
       }
      }
        return persons;
    }
```

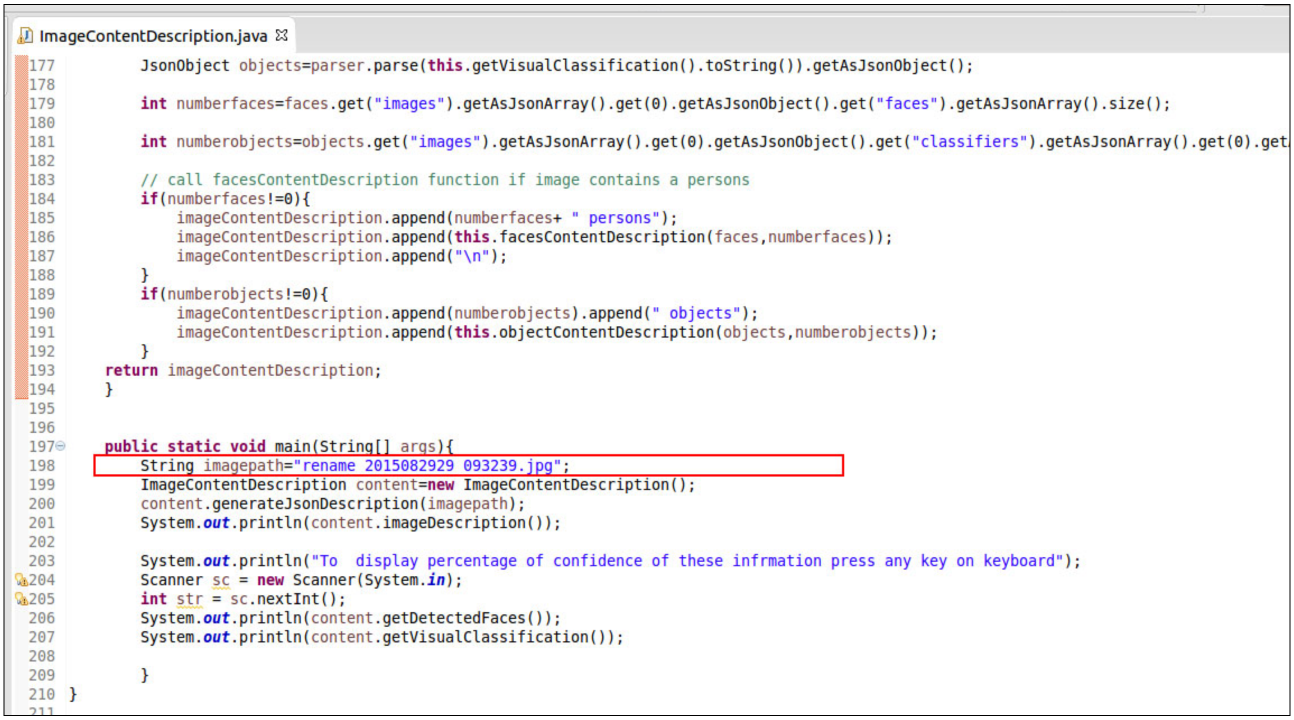
Figure 3-35 The facesContentDescription source code

After exploring the source code, you can run the application (3.3.6, “Running the application” on page 65).

3.3.6 Running the application

To display a description of your image, first set the path of your image, as shown in Figure 3-36. Then, test the program:

1. Copy the path of your image or use the paths of images (loaded with project).



```
        JsonObject objects=parser.parse(this.getVisualClassification().toString()).getAsJsonObject();

        int numberfaces=faces.get("images").getAsJsonArray().get(0).getAsJsonObject().get("faces").getAsJsonArray().size();

        int numberobjects=objects.get("images").getAsJsonArray().get(0).getAsJsonObject().get("classifiers").getAsJsonArray().get(0).get

        // call facesContentDescription function if image contains a persons
        if(numberfaces!=0){
            imageContentDescription.append(numberfaces+ " persons");
            imageContentDescription.append(this.facesContentDescription(faces,numberfaces));
            imageContentDescription.append("\n");
        }
        if(numberobjects!=0){
            imageContentDescription.append(numberobjects).append(" objects");
            imageContentDescription.append(this.objectContentDescription(objects,numberobjects));
        }
    return imageContentDescription;
    }


    public static void main(String[] args){
        String imagepath="rename 2015082929 093239.jpg";
        ImageContentDescription content=new ImageContentDescription();
        content.generateJsonDescription(imagepath);
        System.out.println(content.imageDescription());

        System.out.println("To  display percentage of confidence of these infrmation press any key on keyboard");
        Scanner sc = new Scanner(System.in);
        int str = sc.nextInt();
        System.out.println(content.getDetectedFaces());
        System.out.println(content.getVisualClassification());

        }
}
```

Figure 3-36   Specify the image path

2. Run the project. Right-clicking the project and select **Run As** → **Run Configurations** (Figure 3-37).

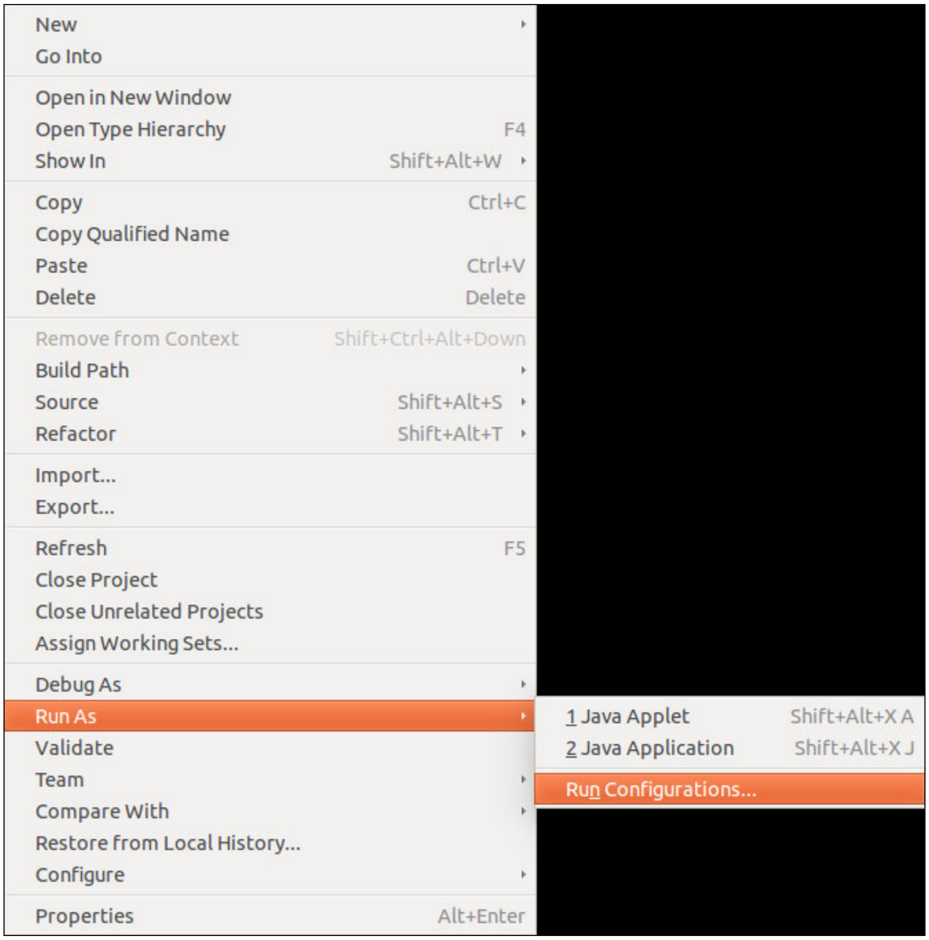
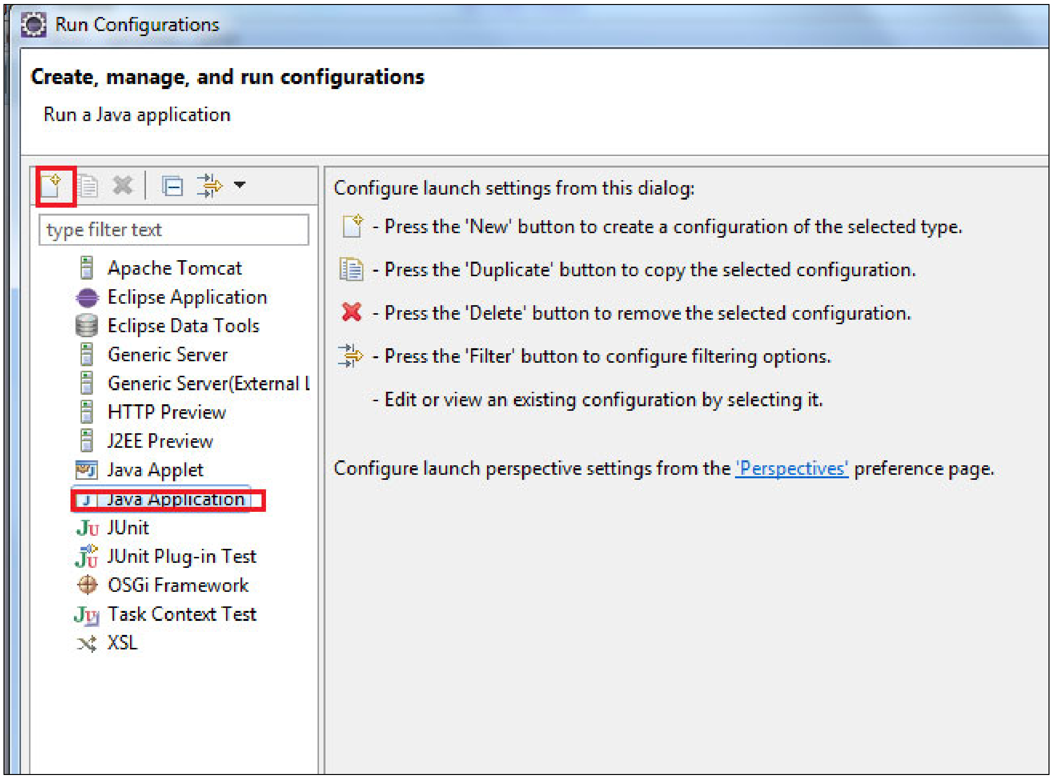


Figure 3-37 Run Configurations

3. Select **Java Application** and click the **New** button to create a configuration (Figure 3-38).



Run Configurations

Create, manage, and run configurations

Run a Java application

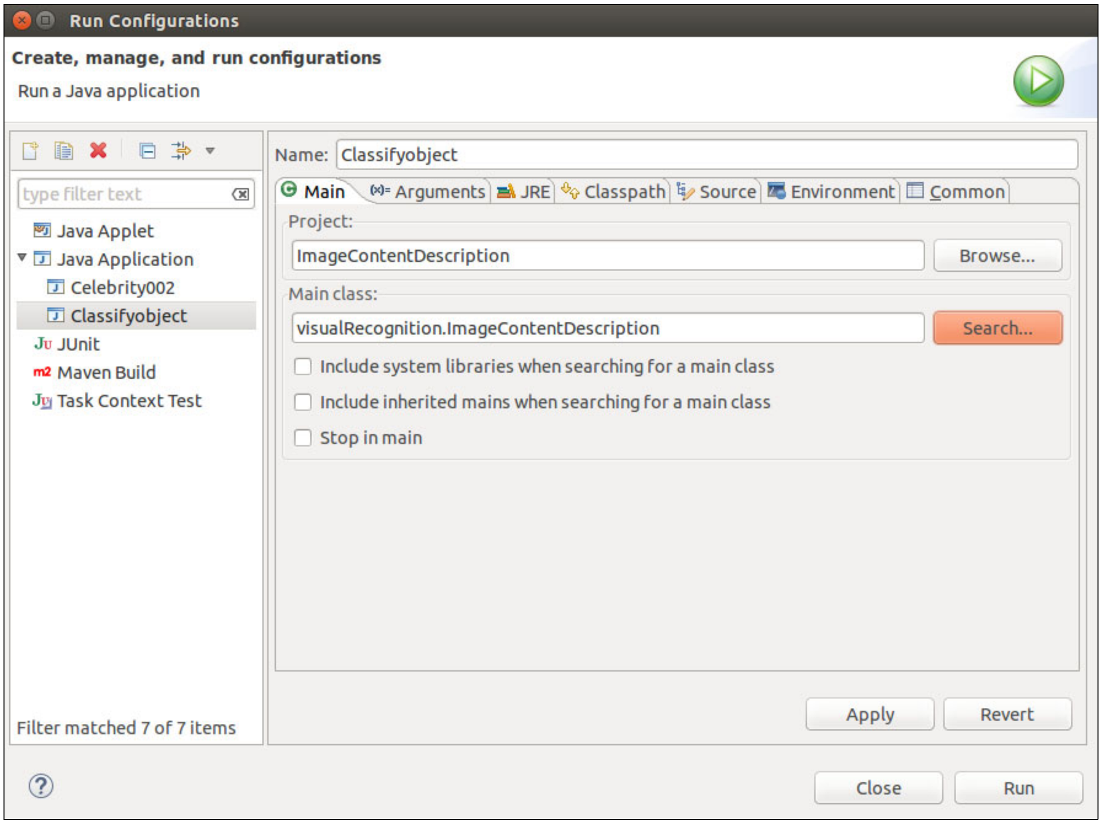
Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

Figure 3-38 The New button

4. On the Main page (Figure 3-39), click **Browse** to find and select the `ImageContentDescription` project, click **Search** to find and select the main class, and then click **Run**.



Run Configurations

Create, manage, and run configurations

Run a Java application

type filter text

- Java Applet
- Java Application
  - Celebrity002
  - Classifyobject
- JUnit
- Maven Build
- Task Context Test

Filter matched 7 of 7 items

Name: Classifyobject

Project: ImageContentDescription

Main class: visualRecognition.ImageContentDescription

□ Include system libraries when searching for a main class

□ Include inherited mains when searching for a main class

□ Stop in main

Figure 3-39 Select the project and main class

The input image is shown in Figure 3-40; the result is shown in Figure 3-41 on page 69 and Figure 3-42 on page 70.

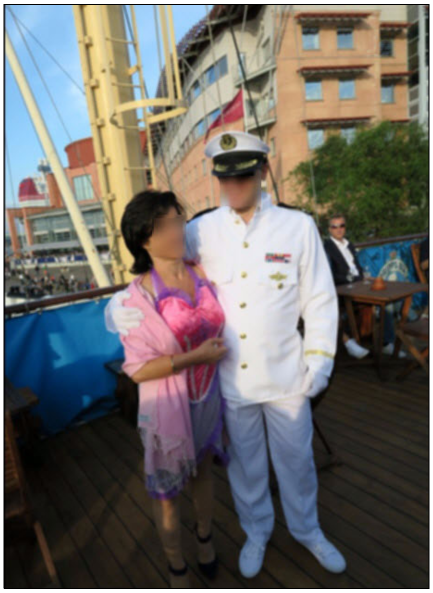
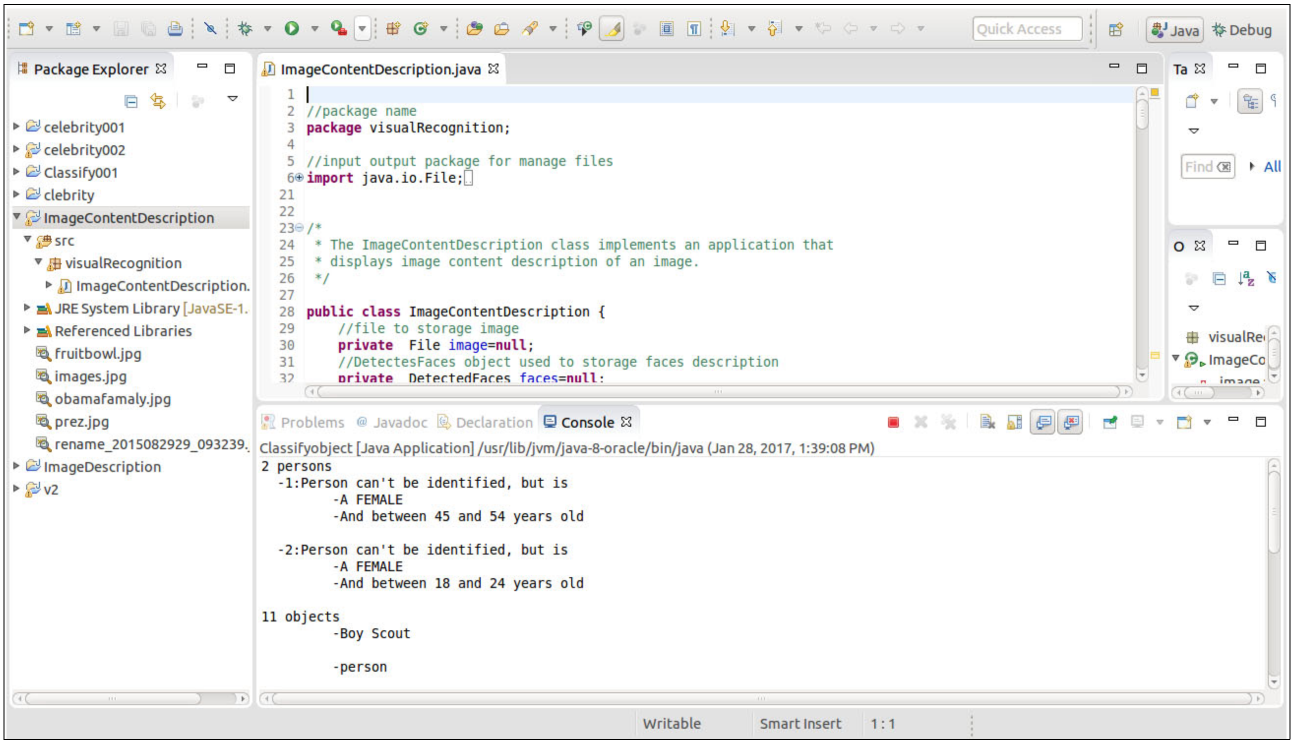


Figure 3-40 Input image for first test (recognize that a person is in the image)

Figure 3-41 shows the result.



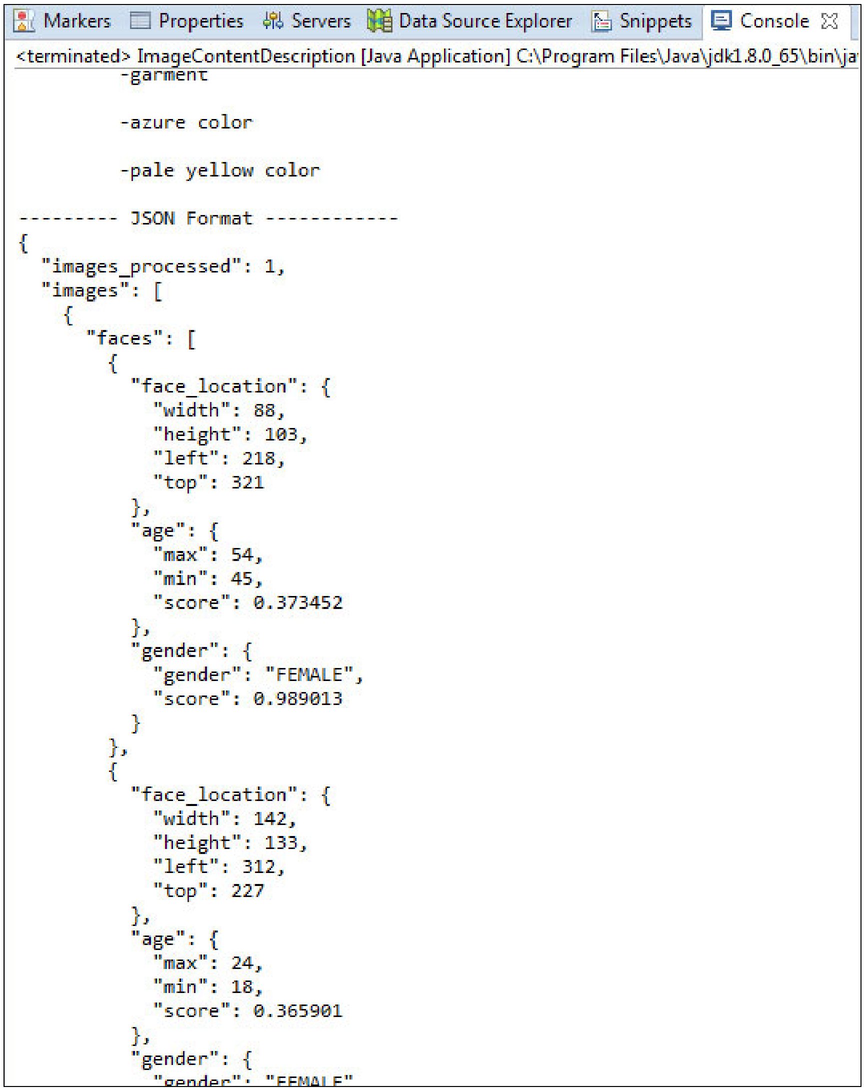
```

//package name
package visualRecognition;

//input output package for manage files
import java.io.File;


/*
 * The ImageContentDescription class implements an application that
 * displays image content description of an image.
 */

public class ImageContentDescription {
    //file to storage image
    private  File image=null;
    //DetectesFaces object used to storage faces description
    private  DetectedFaces faces=null;
```

Classifyobject [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Jan 28, 2017, 1:39:08 PM)

```
2 persons
  -1:Person can't be identified, but is
         -A FEMALE
         -And between 45 and 54 years old

  -2:Person can't be identified, but is
         -A FEMALE
         -And between 18 and 24 years old

11 objects
         -Boy Scout

         -person
```

Figure 3-41 Results

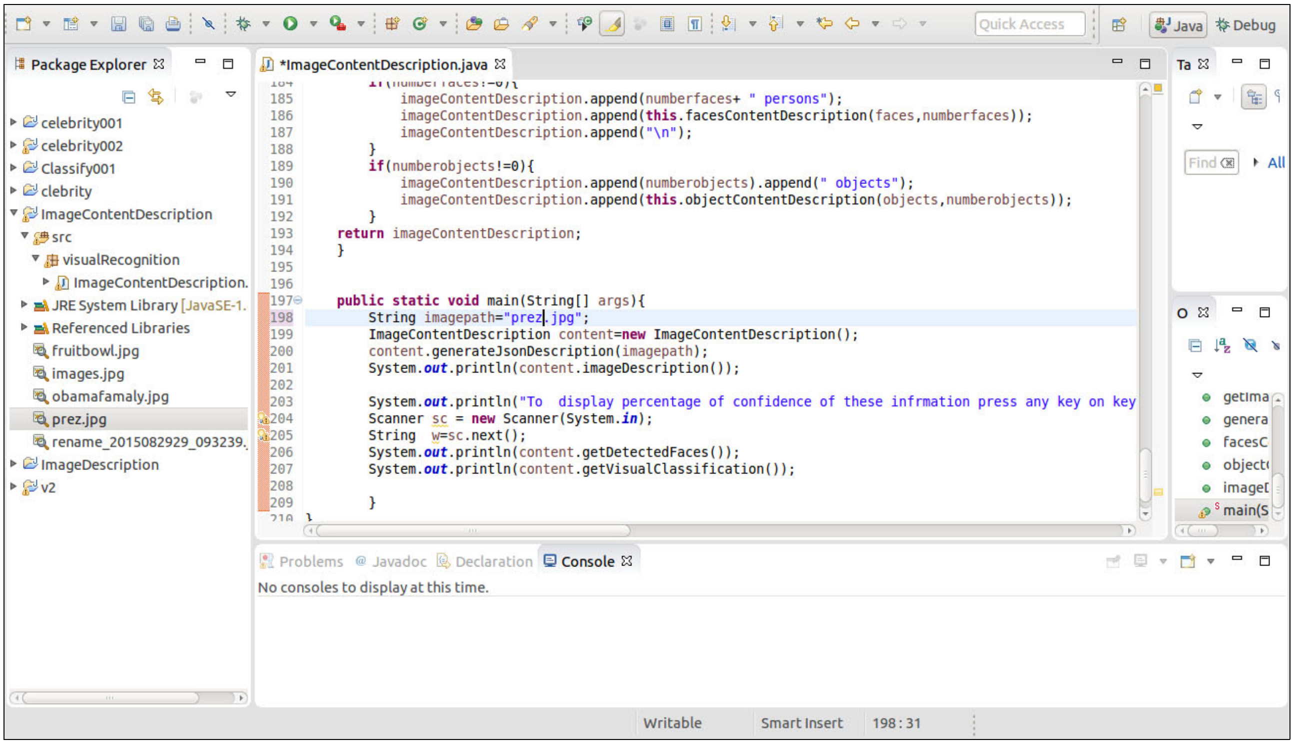
Maximize the console window to show all results (Figure 3-42).



```
<terminated> ImageContentDescription [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\ja
         -garment

         -azure color

         -pale yellow color

--------- JSON Format ------------
{
  "images_processed": 1,
  "images": [
    {
      "faces": [
        {
          "face_location": {
            "width": 88,
            "height": 103,
            "left": 218,
            "top": 321
          },
          "age": {
            "max": 54,
            "min": 45,
            "score": 0.373452
          },
          "gender": {
            "gender": "FEMALE",
            "score": 0.989013
          }
        },
        {
          "face_location": {
            "width": 142,
            "height": 133,
            "left": 312,
            "top": 227
          },
          "age": {
            "max": 24,
            "min": 18,
            "score": 0.365901
          },
          "gender": {
            "gender": "FEMALE"
```

Figure 3-42 JSON object results

Another test of the program uses the image of former President Obama to show how the program can recognize a celebrity face. Change the path to the image path (Figure 3-43).



```
        if(numberfaces!=0){
            imageContentDescription.append(numberfaces+ " persons");
            imageContentDescription.append(this.facesContentDescription(faces,numberfaces));
            imageContentDescription.append("\n");
        }
        if(numberobjects!=0){
            imageContentDescription.append(numberobjects).append(" objects");
            imageContentDescription.append(this.objectContentDescription(objects,numberobjects));
        }
    return imageContentDescription;
    }


    public static void main(String[] args){
        String imagepath="prez.jpg";
        ImageContentDescription content=new ImageContentDescription();
        content.generateJsonDescription(imagepath);
        System.out.println(content.imageDescription());

        System.out.println("To  display percentage of confidence of these infrmation press any key on key
        Scanner sc = new Scanner(System.in);
        String  w=sc.next();
        System.out.println(content.getDetectedFaces());
        System.out.println(content.getVisualClassification());

        }
}
```

No consoles to display at this time.

Figure 3-43 Another image test

The input image is shown in Figure 3-44; the result is shown in Figure 3-45 and Figure 3-46 on page 73.

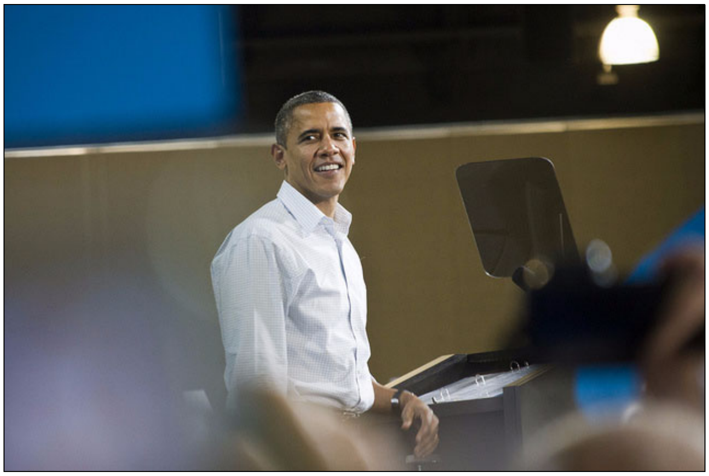
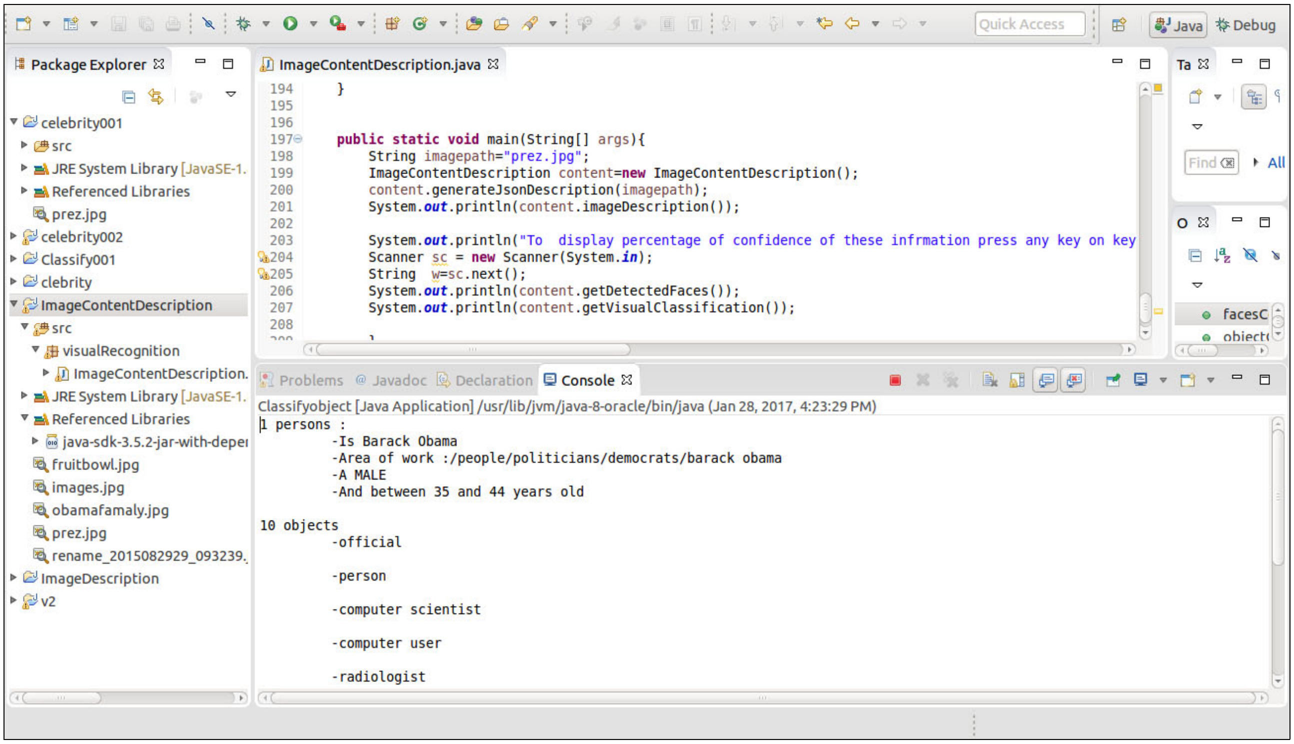


Figure 3-44   Input image for second test (recognize that the image is of a celebrity person)

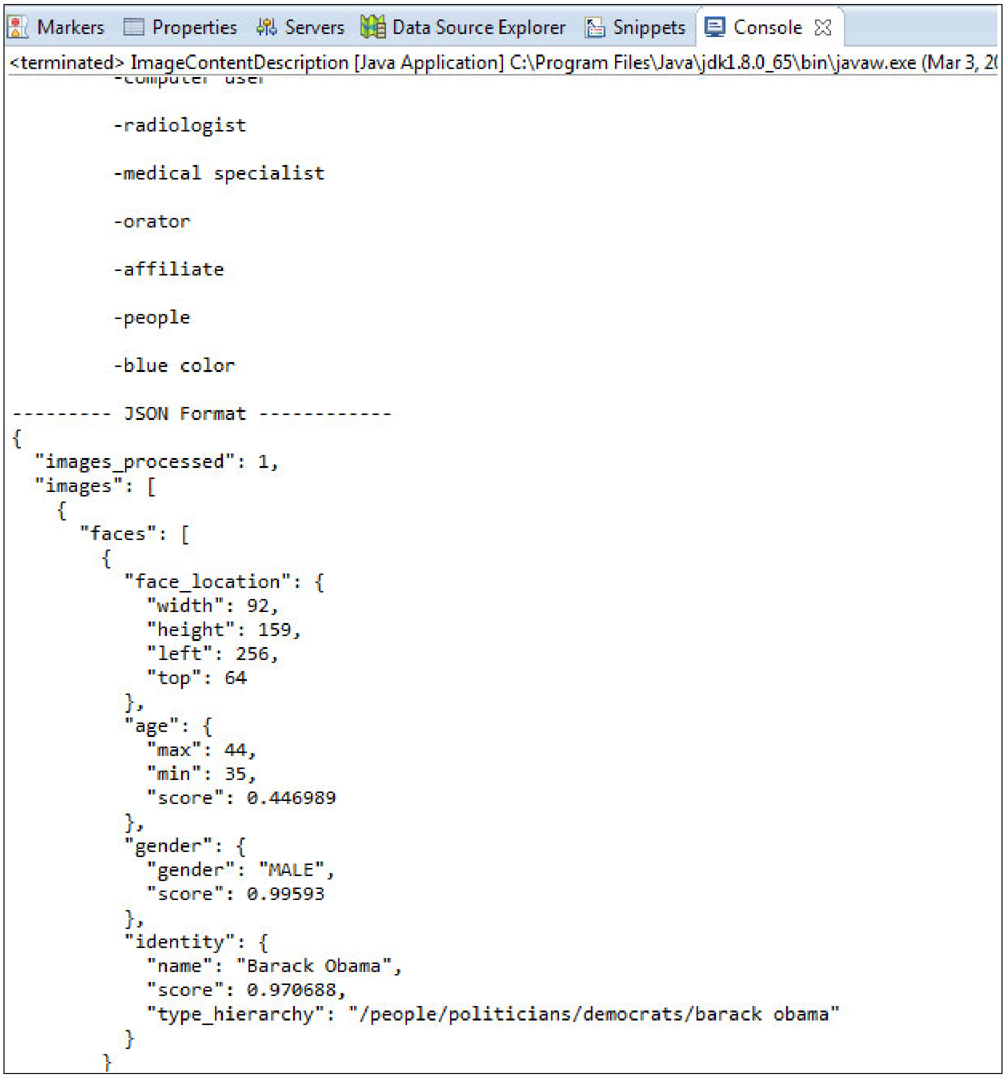


```
    }


    public static void main(String[] args){
        String imagepath="prez.jpg";
        ImageContentDescription content=new ImageContentDescription();
        content.generateJsonDescription(imagepath);
        System.out.println(content.imageDescription());

        System.out.println("To  display percentage of confidence of these infrmation press any key on key
        Scanner sc = new Scanner(System.in);
        String  w=sc.next();
        System.out.println(content.getDetectedFaces());
        System.out.println(content.getVisualClassification());
```

```
Classifyobject [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Jan 28, 2017, 4:23:29 PM)
1 persons :
         -Is Barack Obama
         -Area of work :/people/politicians/democrats/barack obama
         -A MALE
         -And between 35 and 44 years old

10 objects
         -official

         -person

         -computer scientist

         -computer user

         -radiologist
```

Figure 3-45   Image description for Barack Obama image

Maximize the console window to show all results (Figure 3-46).



<terminated> ImageContentDescription [Java Application] C:\Program Files\Java\jdk1.8.0_65\bin\javaw.exe (Mar 3, 2(

```
         -computer user

         -radiologist

         -medical specialist

         -orator

         -affiliate

         -people

         -blue color

--------- JSON Format ------------
{
  "images_processed": 1,
  "images": [
    {
      "faces": [
        {
          "face_location": {
            "width": 92,
            "height": 159,
            "left": 256,
            "top": 64
          },
          "age": {
            "max": 44,
            "min": 35,
            "score": 0.446989
          },
          "gender": {
            "gender": "MALE",
            "score": 0.99593
          },
          "identity": {
            "name": "Barack Obama",
            "score": 0.970688,
            "type_hierarchy": "/people/politicians/democrats/barack obama"
          }
        }
```

Figure 3-46 Image description

3.4 Deploy a Java application to Bluemix

To deploy the project to Bluemix, first create a runnable JAR file and then use the Cloud Foundry command-line interface (CLI) to deploy the application.

3.4.1 Create a runnable JAR file to deploy the application to Bluemix

Complete these steps to create a runnable JAR file:

1. Select **File** → **Export**. In the Export window, make sure that you export it as a `Runnable JAR file`, *not* as a standard JAR file, and then click **Next** (Figure 3-47).

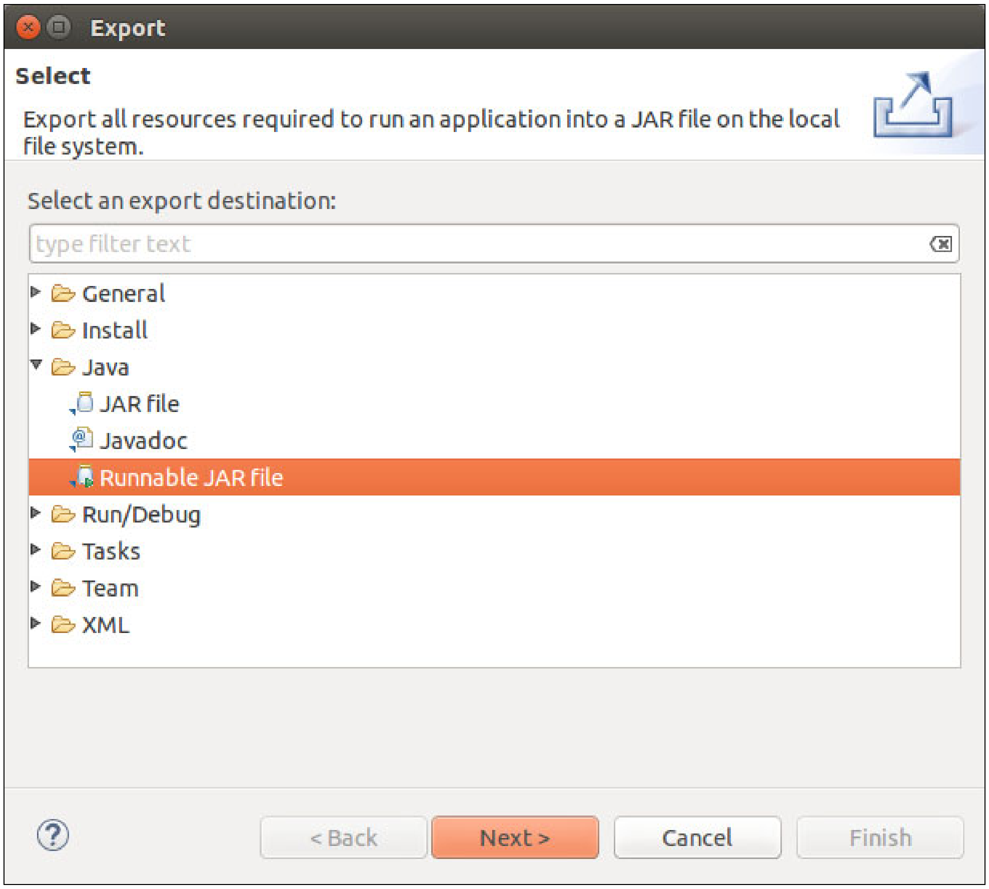
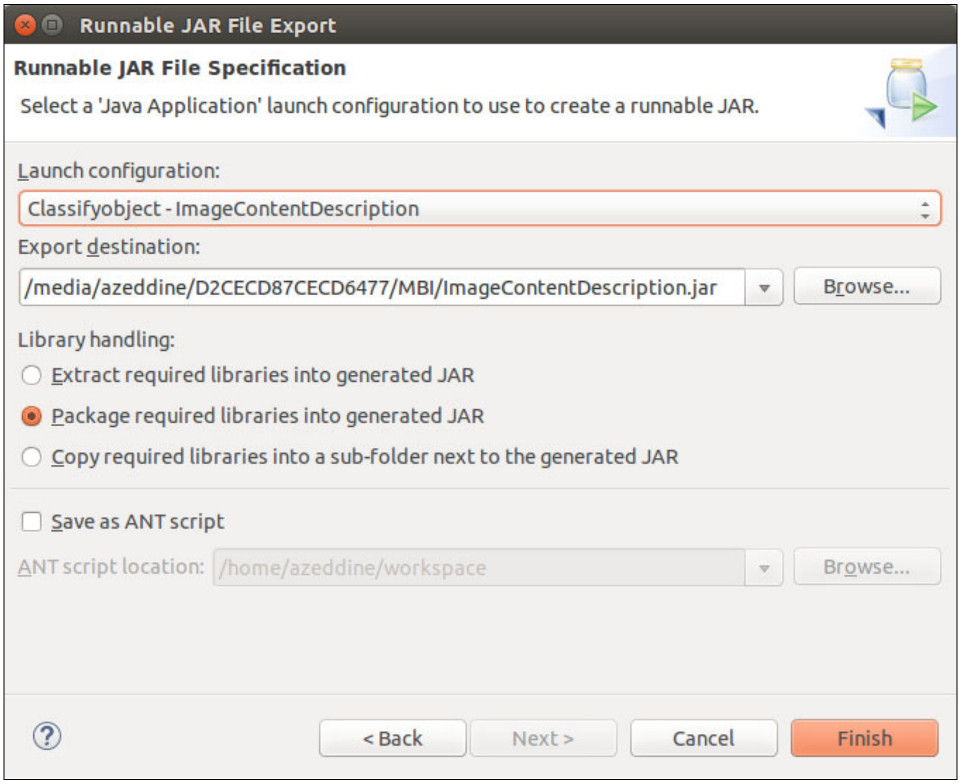


Figure 3-47 Select type of export file

2. In the Launch configuration field, select **ImageContentDescription**. In the Export destination field, click **Browse** (Figure 3-48).



Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration: Classifyobject - ImageContentDescription

Export destination: /media/azeddine/D2CECD87CECD6477/MBI/ImageContentDescription.jar

Library handling:

○ Extract required libraries into generated JAR

● Package required libraries into generated JAR

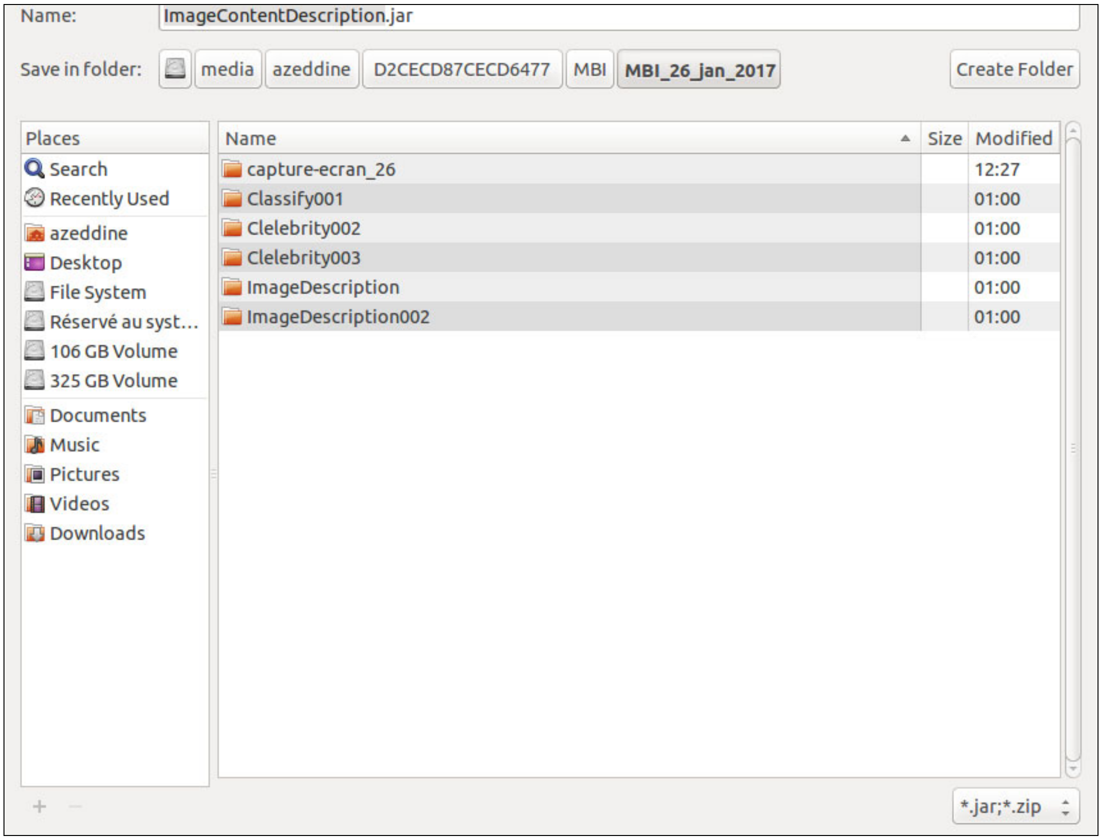
○ Copy required libraries into a sub-folder next to the generated JAR

□ Save as ANT script

ANT script location: /home/azeddine/workspace

Figure 3-48 ImageContentDescription runnable JAR specification

3. Browse to the folder where you will export your launch configuration, enter the name of your JAR file, and click **OK** (Figure 3-49).



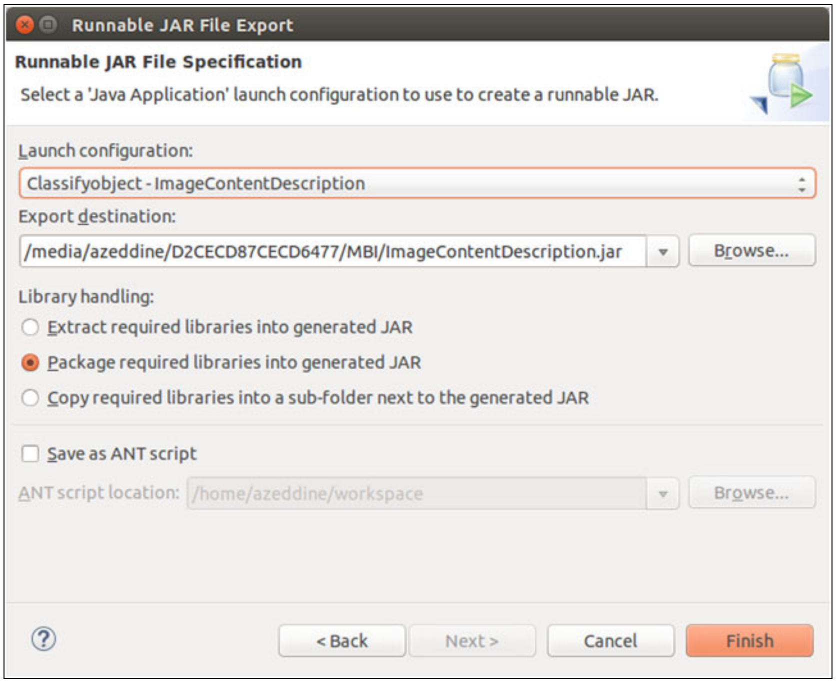
Name: ImageContentDescription.jar

Save in folder: media | azeddine | D2CECD87CECD6477 | MBI | MBI_26_jan_2017

Name	Size	Modified
capture-ecran_26		12:27
Classify001		01:00
Clelebrity002		01:00
Clelebrity003		01:00
ImageDescription		01:00
ImageDescription002		01:00

Figure 3-49 Specify name of JAR file

4. You are returned to the previous window (Figure 3-50). Select **Package required libraries into generated JAR**, and click **Finish**. This creates the runnable JAR file.



Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration: Classifyobject - ImageContentDescription

Export destination: /media/azeddine/D2CECD87CECD6477/MBI/ImageContentDescription.jar

Library handling:

○ Extract required libraries into generated JAR

● Package required libraries into generated JAR

○ Copy required libraries into a sub-folder next to the generated JAR

□ Save as ANT script

ANT script location: /home/azeddine/workspace

Figure 3-50 Runnable JAR File Export

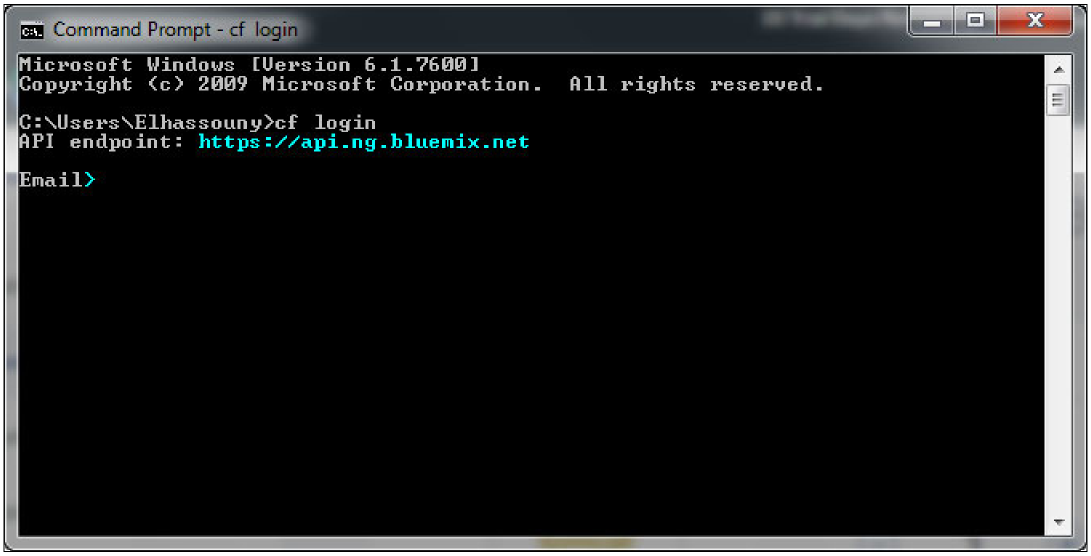
3.4.2 Deploy the Java application to Bluemix

This section explains how to make a stand-alone Java program, with a `main()` method, run in Bluemix.

For more information, see Move your Java application into a hybrid cloud using Bluemix, which is in IBM developerWorks.

Complete these steps:

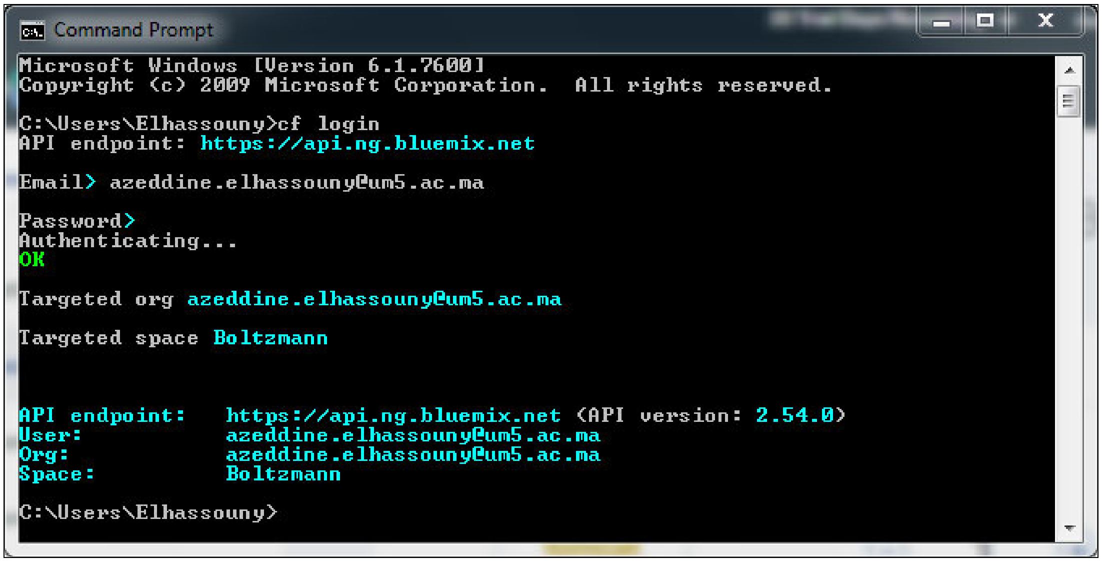
1. Download and install the Cloud Foundry command-line interface.
2. Open a Command Prompt session and run the **`cf login`** command (Figure 3-51).



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Elhassouny>cf login
API endpoint: https://api.ng.bluemix.net

Email>
```

Figure 3-51 Authentication to Cloud Foundry

3. Enter your IBMid (the email address that you use to sign in to Bluemix) and your password (Figure 3-52).



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Elhassouny>cf login
API endpoint: https://api.ng.bluemix.net

Email> azeddine.elhassouny@um5.ac.ma

Password>
Authenticating...
OK

Targeted org azeddine.elhassouny@um5.ac.ma

Targeted space Boltzmann


API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
User:           azeddine.elhassouny@um5.ac.ma
Org:            azeddine.elhassouny@um5.ac.ma
Space:          Boltzmann

C:\Users\Elhassouny>
```

Figure 3-52 Authentication result

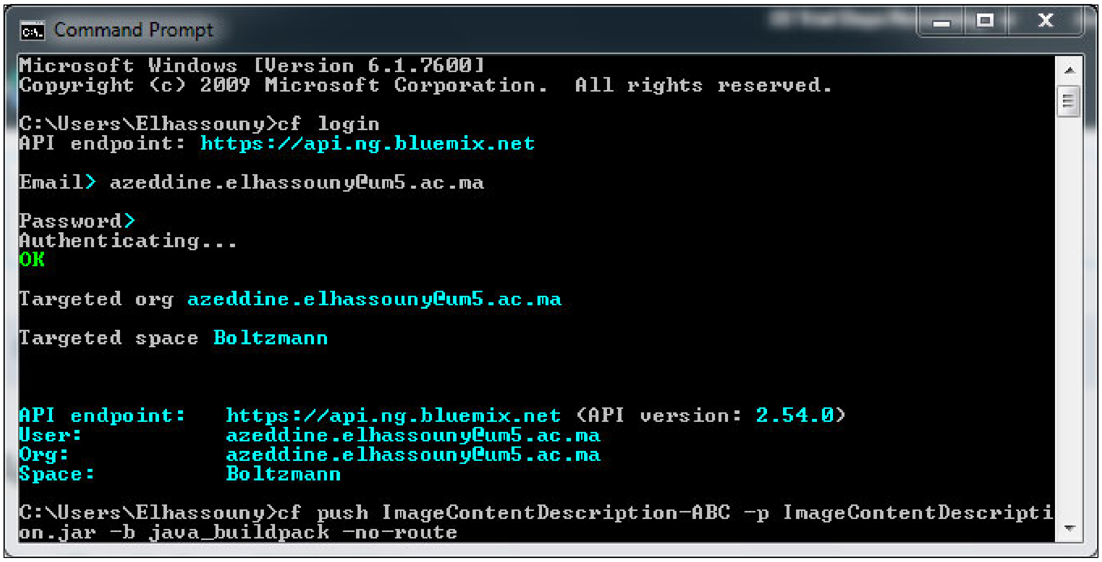
4. Use one of the following commands to deploy your Java stand-alone application to Bluemix (Figure 3-53). The **cf** command is in this format:

   - `cf push <ANY_APP_NAME> -p <JAR_NAME>.jar -b java_buildpack -no-route`

     For example:

     ```
     cf push ImageContenteDescription-ABC -p ImageContentDescription.jar -b
     java_buildpack -no-route
     ```

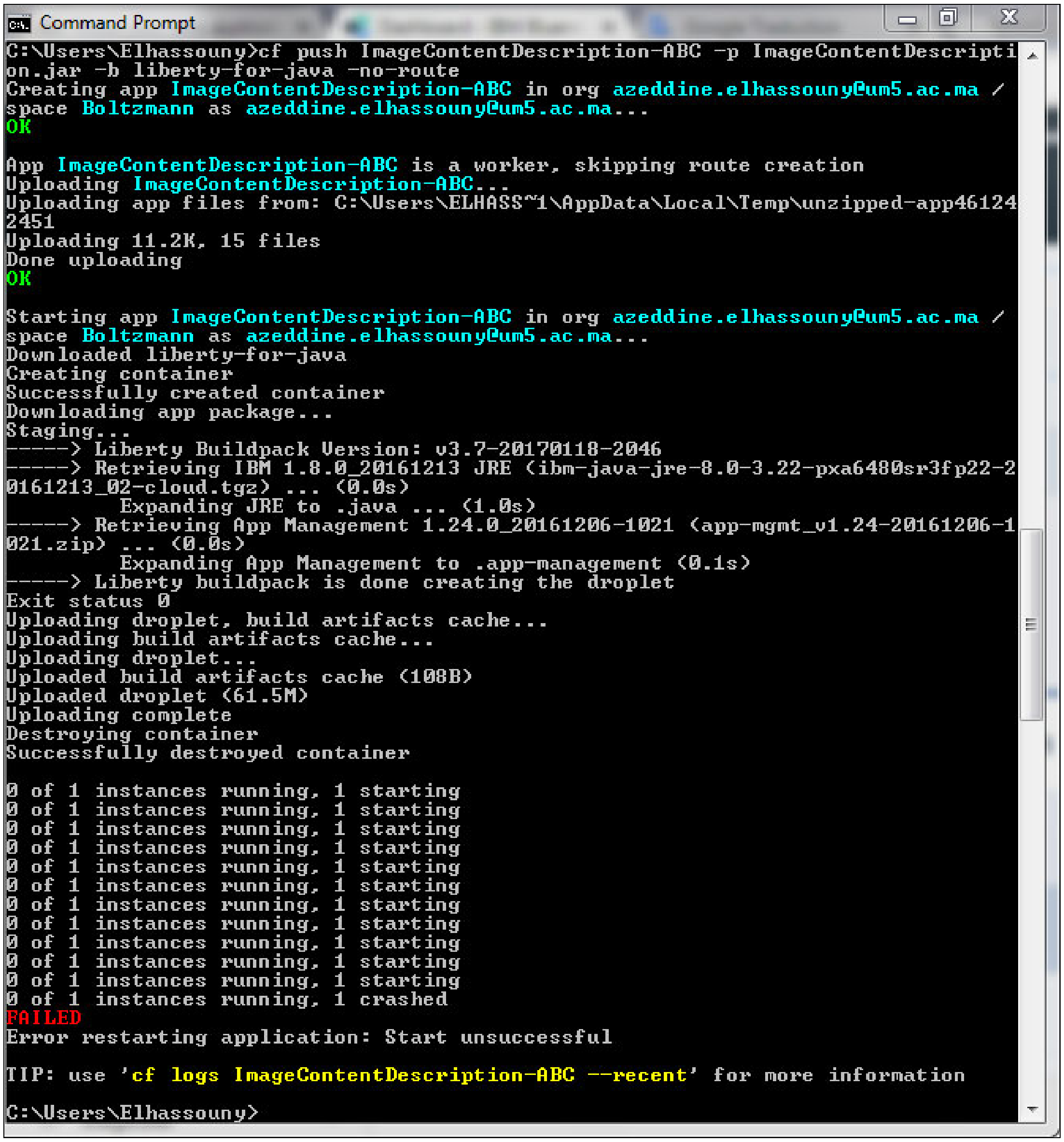
   -
     ```
     cf push ImageContentDescription-ABC -p ImageContentDescription.jar -b
     liberty-for-java -no-route
     ```



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Elhassouny>cf login
API endpoint: https://api.ng.bluemix.net

Email> azeddine.elhassouny@um5.ac.ma

Password>
Authenticating...
OK

Targeted org azeddine.elhassouny@um5.ac.ma

Targeted space Boltzmann


API endpoint:   https://api.ng.bluemix.net (API version: 2.54.0)
User:           azeddine.elhassouny@um5.ac.ma
Org:            azeddine.elhassouny@um5.ac.ma
Space:          Boltzmann

C:\Users\Elhassouny>cf push ImageContentDescription-ABC -p ImageContentDescripti
on.jar -b java_buildpack -no-route
```

Figure 3-53   Commands to deploy your Java stand-alone application

The result is shown in Figure 3-54.



```
C:\Users\Elhassouny>cf push ImageContentDescription-ABC -p ImageContentDescripti
on.jar -b liberty-for-java -no-route
Creating app ImageContentDescription-ABC in org azeddine.elhassouny@um5.ac.ma /
space Boltzmann as azeddine.elhassouny@um5.ac.ma...
OK

App ImageContentDescription-ABC is a worker, skipping route creation
Uploading ImageContentDescription-ABC...
Uploading app files from: C:\Users\ELHASS~1\AppData\Local\Temp\unzipped-app46124
2451
Uploading 11.2K, 15 files
Done uploading
OK

Starting app ImageContentDescription-ABC in org azeddine.elhassouny@um5.ac.ma /
space Boltzmann as azeddine.elhassouny@um5.ac.ma...
Downloaded liberty-for-java
Creating container
Successfully created container
Downloading app package...
Staging...
-----> Liberty Buildpack Version: v3.7-20170118-2046
-----> Retrieving IBM 1.8.0_20161213 JRE (ibm-java-jre-8.0-3.22-pxa6480sr3fp22-2
0161213_02-cloud.tgz) ... (0.0s)
         Expanding JRE to .java ... (1.0s)
-----> Retrieving App Management 1.24.0_20161206-1021 (app-mgmt_v1.24-20161206-1
021.zip) ... (0.0s)
         Expanding App Management to .app-management (0.1s)
-----> Liberty buildpack is done creating the droplet
Exit status 0
Uploading droplet, build artifacts cache...
Uploading build artifacts cache...
Uploading droplet...
Uploaded build artifacts cache (108B)
Uploaded droplet (61.5M)
Uploading complete
Destroying container
Successfully destroyed container

0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 starting
0 of 1 instances running, 1 crashed
FAILED
Error restarting application: Start unsuccessful

TIP: use 'cf logs ImageContentDescription-ABC --recent' for more information

C:\Users\Elhassouny>
```

Figure 3-54 Deployment process result

5. If the deployment is successful, switch to your Bluemix space to check the deployment of your application to Bluemix. Click the **Logs** tab to see the execution of your application.

3.5 References

See the following resources:

- OpenCV 3.0.0-dev documentation (Using OpenCV Java with Eclipse):

  http://docs.opencv.org/3.0-beta/doc/tutorials/introduction/java_eclipse/java_eclipse.html

- Watson Developer Cloud: Java SDK Downloads:

  https://github.com/watson-developer-cloud/java-sdk/releases

- Move your Java application into a hybrid cloud using Bluemix, Part 3 web page in IBM developerWorks:

  http://www.ibm.com/developerworks/cloud/library/cl-move-java-app-hybrid-cloud3-bluemix-trs/

- Watson Developer Cloud:

  https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/

- For source code comments, explore documentation with Javadoc in the following file (download the `javadoc.rar` file and extract the contents):

  https://github.com/snippet-java/redbooks-vis-301-ImageContentDescription/blob/master/javadoc.rar

4

Intelligent Video Content Analytics

This chapter focuses on the development of Java programs using the Watson Visual Recognition service to analyze video files and generate video content description.

The Intelligent Video Content Analytics sample application in this chapter performs object classification and face detection on videos instead of images.

In addition, VideoCapture and some other classes of the OpenCV library permit reading a video file and getting frames from it. See the OpenCV website.

This chapter focuses on the development of Java programs using the Watson Visual Recognition service and OpenCV classes to analyze video content.

The program can be run in Eclipse on Linux or Windows.

The following topics are covered in this chapter:

- Getting started
- Architecture
- Implementation
- Changing your application to detect faces
- References

4.1 Getting started

To start, read through the objectives, prerequisites, and expected results of this use case.

4.1.1 Objectives

After completing this chapter, you should be able to accomplish these objectives:

- Investigate the set of built-in classes of Watson Visual Recognition and OpenCV to perform object classification and face detection on video files instead of a photographic image.
- Use Watson Visual Recognition service and OpenCV for your own projects using video captured from any source (file, camera, or others).

4.1.2 Prerequisites

You must have the following accounts, resources, knowledge, and experiences:

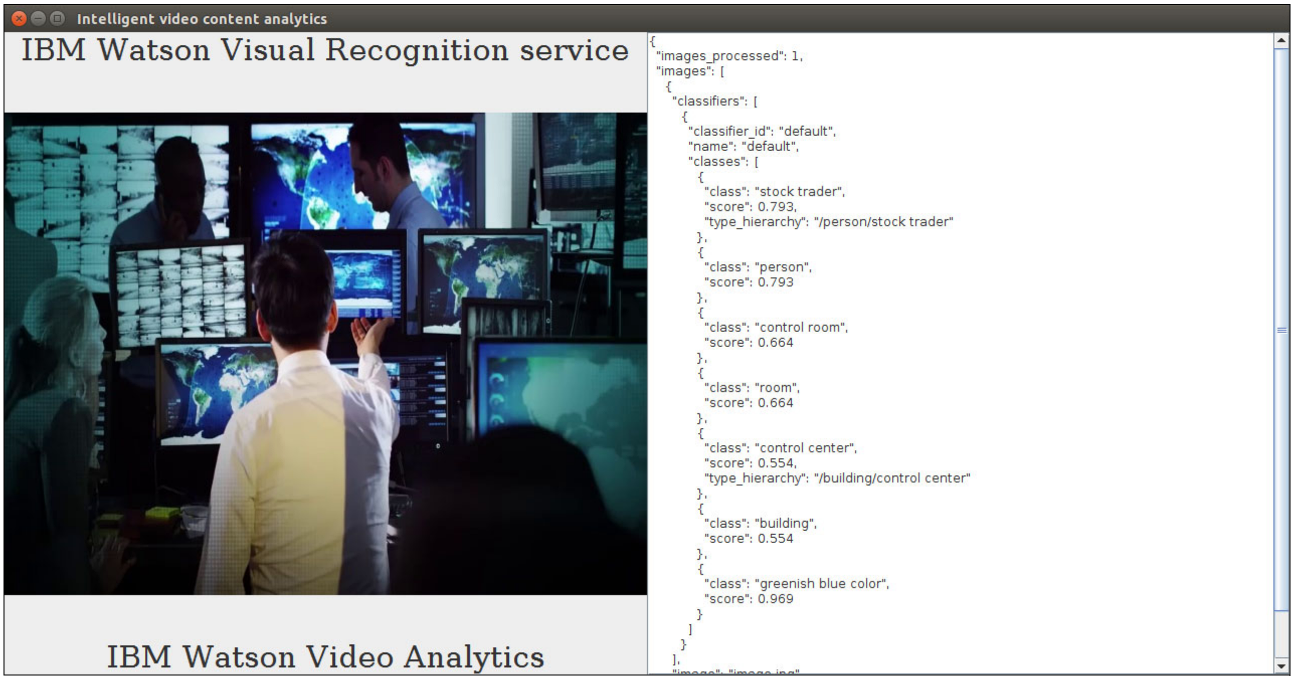
- An IBM Bluemix account (register for a new account or log in to Bluemix if you already have an account)
- Eclipse IDE Luna
- Java 8
- OpenCV 3.x.x for Java, installed

4.1.3 Expected results

The video file you analyze in this chapter contains various scenes that IBM created. It summarizes a diversity of objects and people in different but real daily situations and will serve as a real test of the program.

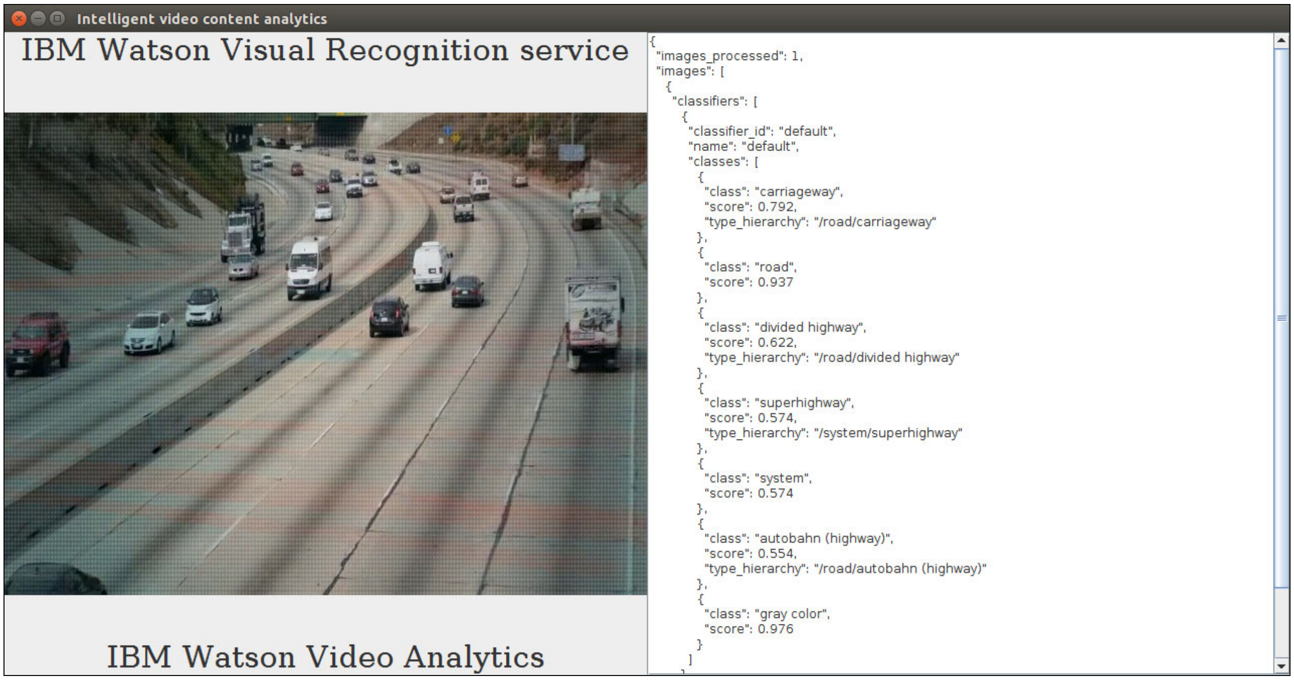
The following images illustrate a subset of sample output results that are displayed when running the sample program:

- Figure 4-1 on page 85: Result obtained for a control center scene in video input
- Figure 4-2 on page 85: Result obtained for road scene in input video
- Figure 4-3 on page 86: Result obtained for surveillance system scene in input video
- Figure 4-4 on page 86: Result obtained for person in scene in input video


Intelligent video content analytics
IBM Watson Visual Recognition service
IBM Watson Video Analytics

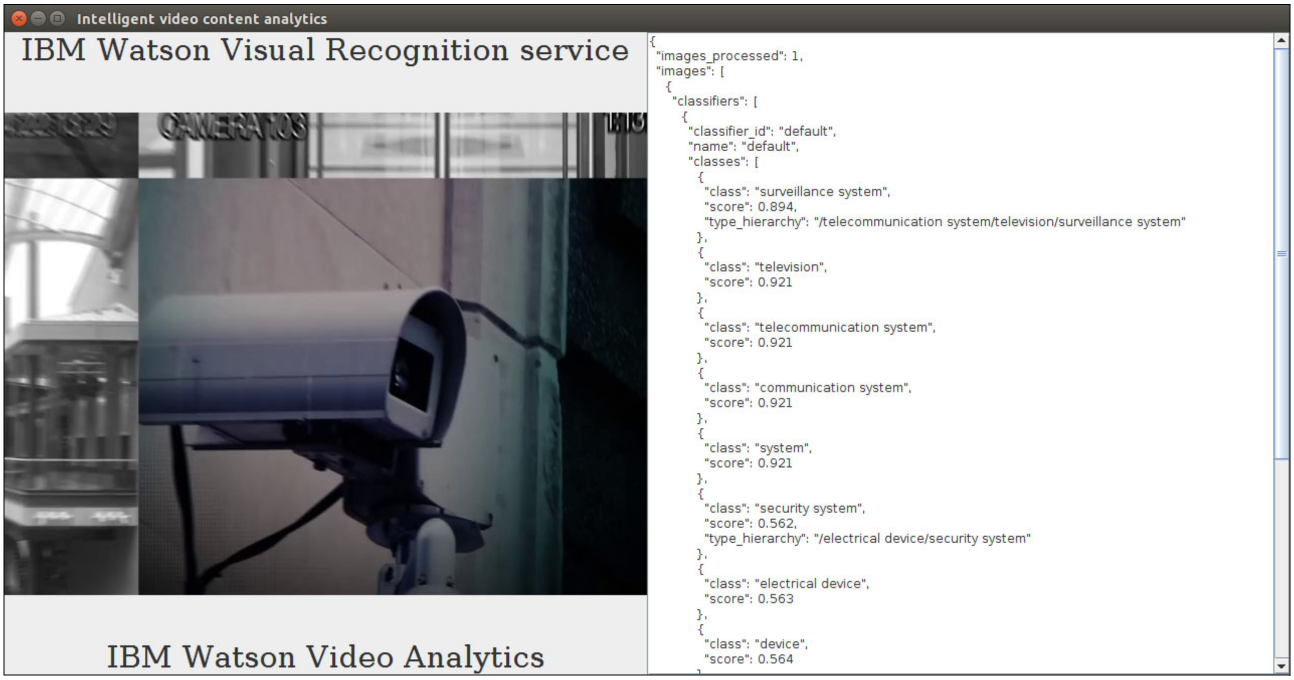
```
{
 "images_processed": 1,
 "images": [
  {
   "classifiers": [
    {
     "classifier_id": "default",
     "name": "default",
     "classes": [
      {
       "class": "stock trader",
       "score": 0.793,
       "type_hierarchy": "/person/stock trader"
      },
      {
       "class": "person",
       "score": 0.793
      },
      {
       "class": "control room",
       "score": 0.664
      },
      {
       "class": "room",
       "score": 0.664
      },
      {
       "class": "control center",
       "score": 0.554,
       "type_hierarchy": "/building/control center"
      },
      {
       "class": "building",
       "score": 0.554
      },
      {
       "class": "greenish blue color",
       "score": 0.969
      }
     ]
    }
   ],
   "image": "image.jpg"
```

Figure 4-1   Result obtained for a control center scene in video input


Intelligent video content analytics
IBM Watson Visual Recognition service
IBM Watson Video Analytics

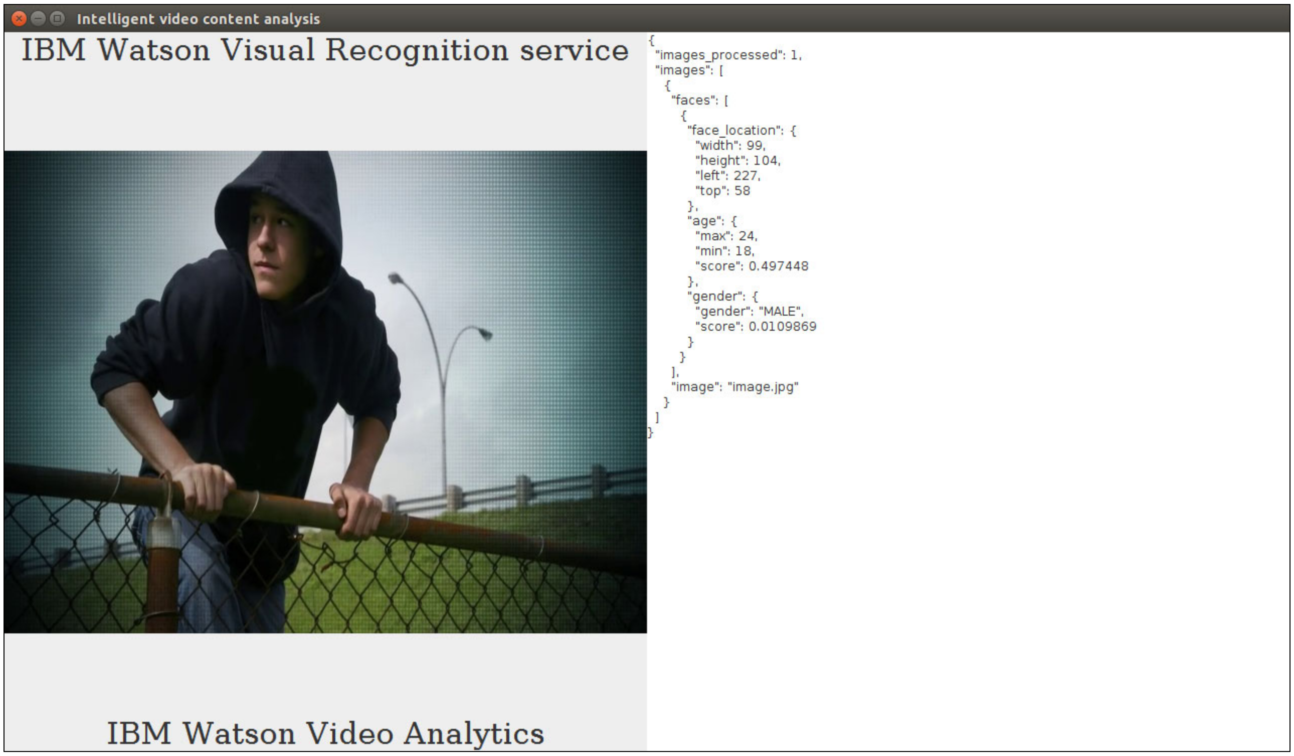
```
{
 "images_processed": 1,
 "images": [
  {
   "classifiers": [
    {
     "classifier_id": "default",
     "name": "default",
     "classes": [
      {
       "class": "carriageway",
       "score": 0.792,
       "type_hierarchy": "/road/carriageway"
      },
      {
       "class": "road",
       "score": 0.937
      },
      {
       "class": "divided highway",
       "score": 0.622,
       "type_hierarchy": "/road/divided highway"
      },
      {
       "class": "superhighway",
       "score": 0.574,
       "type_hierarchy": "/system/superhighway"
      },
      {
       "class": "system",
       "score": 0.574
      },
      {
       "class": "autobahn (highway)",
       "score": 0.554,
       "type_hierarchy": "/road/autobahn (highway)"
      },
      {
       "class": "gray color",
       "score": 0.976
      }
     ]
```

Figure 4-2   Result obtained for road scene in input video


Intelligent video content analytics
IBM Watson Visual Recognition service
IBM Watson Video Analytics

```
{
 "images_processed": 1,
 "images": [
  {
   "classifiers": [
    {
     "classifier_id": "default",
     "name": "default",
     "classes": [
      {
       "class": "surveillance system",
       "score": 0.894,
       "type_hierarchy": "/telecommunication system/television/surveillance system"
      },
      {
       "class": "television",
       "score": 0.921
      },
      {
       "class": "telecommunication system",
       "score": 0.921
      },
      {
       "class": "communication system",
       "score": 0.921
      },
      {
       "class": "system",
       "score": 0.921
      },
      {
       "class": "security system",
       "score": 0.562,
       "type_hierarchy": "/electrical device/security system"
      },
      {
       "class": "electrical device",
       "score": 0.563
      },
      {
       "class": "device",
       "score": 0.564
```

Figure 4-3   Result obtained for surveillance system scene in input video


Intelligent video content analysis
IBM Watson Visual Recognition service
IBM Watson Video Analytics

```
{
 "images_processed": 1,
 "images": [
  {
   "faces": [
    {
     "face_location": {
      "width": 99,
      "height": 104,
      "left": 227,
      "top": 58
     },
     "age": {
      "max": 24,
      "min": 18,
      "score": 0.497448
     },
     "gender": {
      "gender": "MALE",
      "score": 0.0109869
     }
    }
   ],
   "image": "image.jpg"
  }
 ]
}
```

Figure 4-4   Result obtained for person in scene in input video

4.2 Architecture

Figure 4-5 summarizes the main steps of the program:

1. First, the video is loaded using VideoCapture (an OpenCV class).
2. The video is divided into individual frames that are processed sequentially.
3. Each frame is passed to the Watson Visual Recognition service, which detects faces and classifies objects contained in the frame.
4. The results are sent to the `display` method which displays the video frame, the detected objects (or faces), and additional descriptive information.

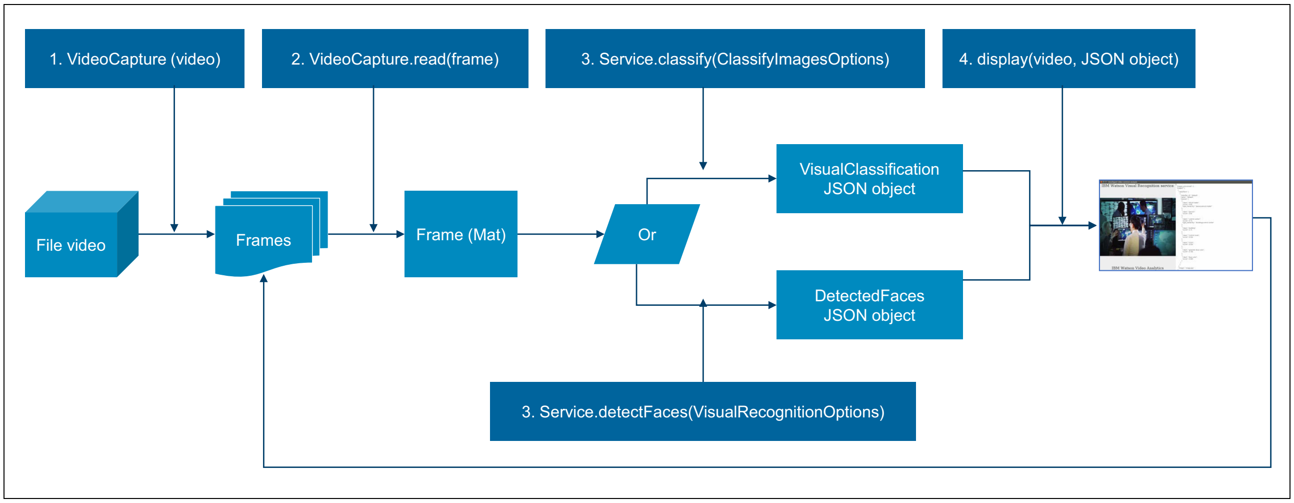

1. VideoCapture (video)
2. VideoCapture.read(frame)
3. Service.classify(ClassifyImagesOptions)
4. display(video, JSON object)
File video
Frames
Frame (Mat)
Or
VisualClassification JSON object
DetectedFaces JSON object
3. Service.detectFaces(VisualRecognitionOptions)

Figure 4-5 Flow chart of the Intelligent Video Content Analytics program

Before starting, you will need an input video file and credentials of a Watson Visual Recognition service instance. The program reads the input video file and displays a JSON object describing its content:

1. VideoCapture (an OpenCV class) captures video from the input video file.

   Steps 2, 3, and 4 are repeated until the video ends.

2. The video is read frame by frame.
3. The current frame is used to create an *options* object (either the `ClassifyImagesOptions` class or the `VisualRecognitionOptions` class).

   This *options* object is used as an argument when accessing the Watson Visual Recognition service (either the `classify` or `detectFaces` method on the `VisualRecognition` class) depending if you want to classify objects or detect faces.

4. The result of both methods is a JSON object that describes the frame content. An internal `display` method is called to display the current frame and the description.

4.3 Implementation

Implementing this use case involves the following steps:

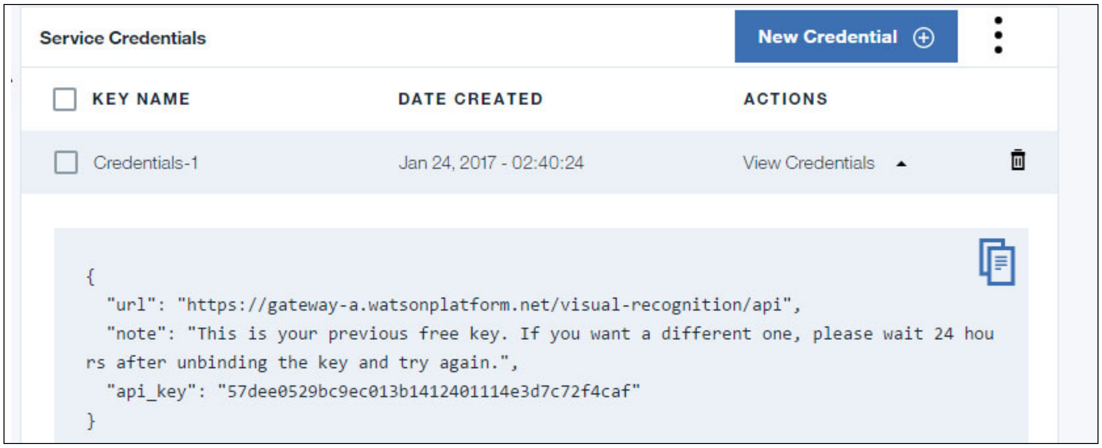
- Creating a Visual Recognition service instance.
- Downloading the project from Git.
- Importing the project to Eclipse.
- Importing Watson Java SDK and additional OpenCV libraries.
- Exploring and completing the sample code provided with the use case.
- Running the application.

4.3.1 Creating a Visual Recognition service instance

Before you can use the Watson services, you must create an instance of the service in Bluemix. For this use case, create a Visual Recognition service instance as described in 1.2, “Creating a Watson Visual Recognition service instance and getting the API key” on page 2.

After creating the service instance, view the credentials (Figure 4-6). Copy and save the following values for later use:

- `url`, which is the API endpoint
- `api-key`, which is the API key



□ KEY NAME	DATE CREATED	ACTIONS
□ Credentials-1	Jan 24, 2017 - 02:40:24	View Credentials ▲

```
{
  "url": "https://gateway-a.watsonplatform.net/visual-recognition/api",
  "note": "This is your previous free key. If you want a different one, please wait 24 hou
rs after unbinding the key and try again.",
  "api_key": "57dee0529bc9ec013b1412401114e3d7c72f4caf"
}
```

Figure 4-6 Credentials of Visual Recognition service instance

4.3.2 Downloading the project from Git

A Git repository is provided for this use case which includes the code to implement the IntelligentVideoContentAnalytics application with comments to make it easier to understand.

1. Download the repository from the following GitHub location:

   https://github.com/snippet-java/redbooks-vis-301-IntelligentVideoContentAnalytics

2. Download `IntelligentVideoContentAnalytics_student.zip` file.
3. Extract the file, which then creates a Java Eclipse Project folder.

4.3.3 Importing the project to Eclipse

In this section you will import the `IntelligentContentVideoAnalytics` project into the Eclipse workspace as an existing project.

After you extract the project, complete these steps:

1. Launch the Eclipse IDE. When prompted for a workspace, keep the existing workspace or change the workspace as desired, and click **OK**.
2. In the Eclipse environment, select **File** → **Import** (Figure 4-7).

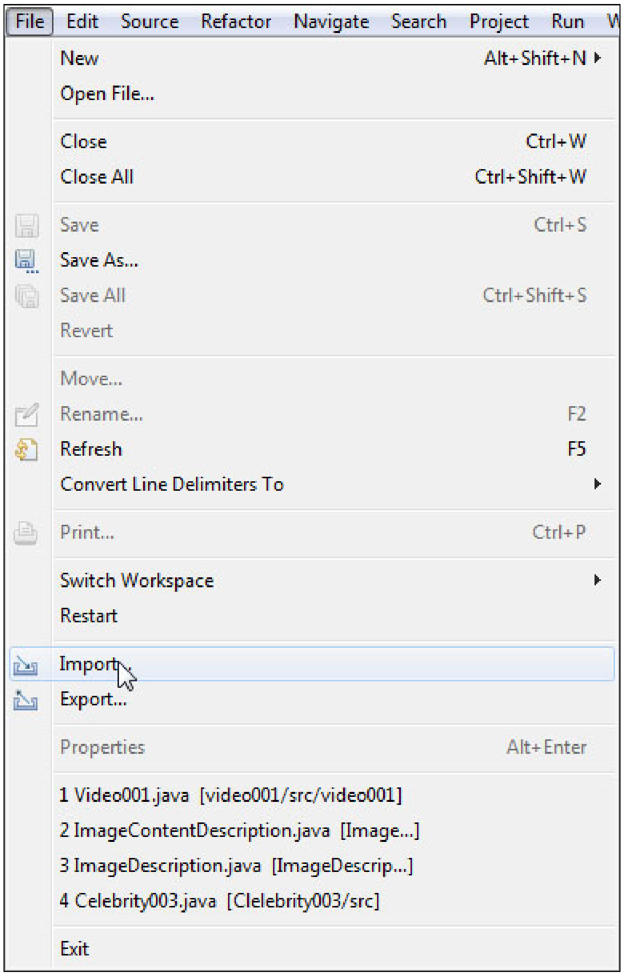
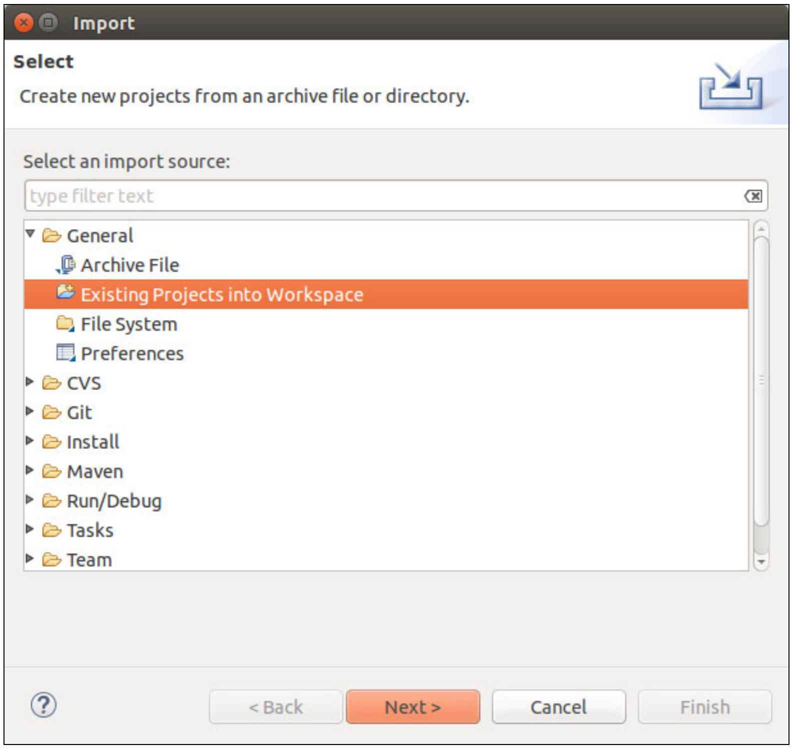

File Edit Source Refactor Navigate Search Project Run W
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
1 Video001.java [video001/src/video001]
2 ImageContentDescription.java [Image...]
3 ImageDescription.java [ImageDescrip...]
4 Celebrity003.java [Clelebrity003/src]
Exit

Figure 4-7 Import project menu

3. Select **General** → **Existing Projects into Workspac**e (Figure 4-8) and click **Next**. The import process has three pages.



Import

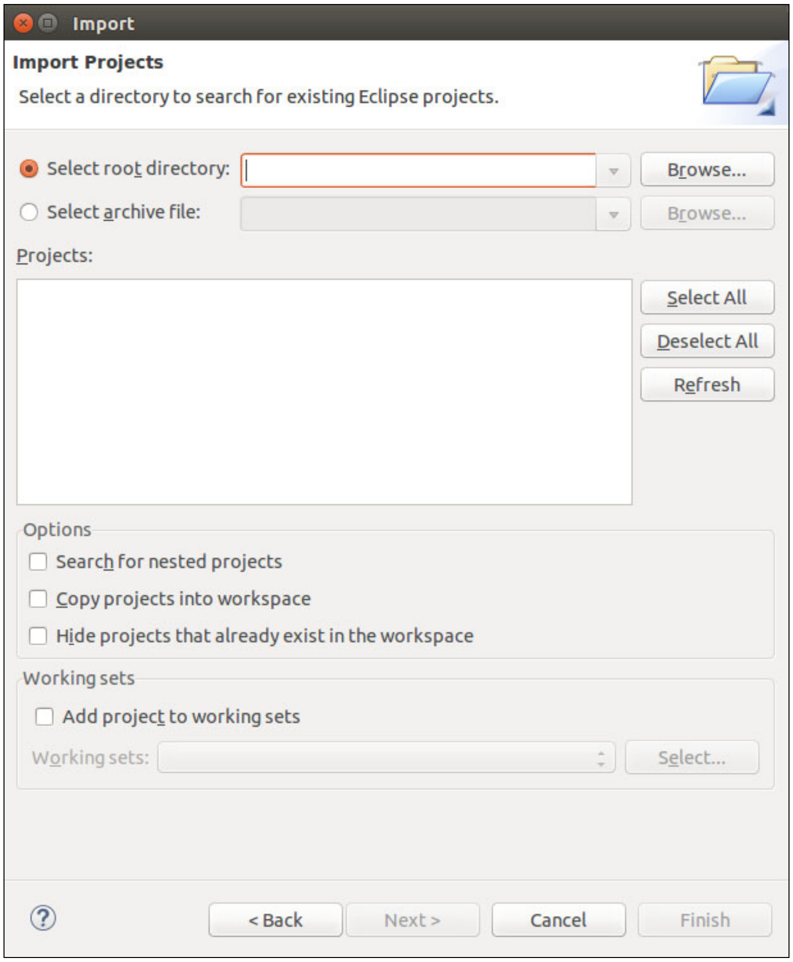
Select

Create new projects from an archive file or directory.

Select an import source:

Figure 4-8 Type imported project dialog

4. Select a root directory. Click **Browse** to navigate to your project's directory (Figure 4-9).



Import

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

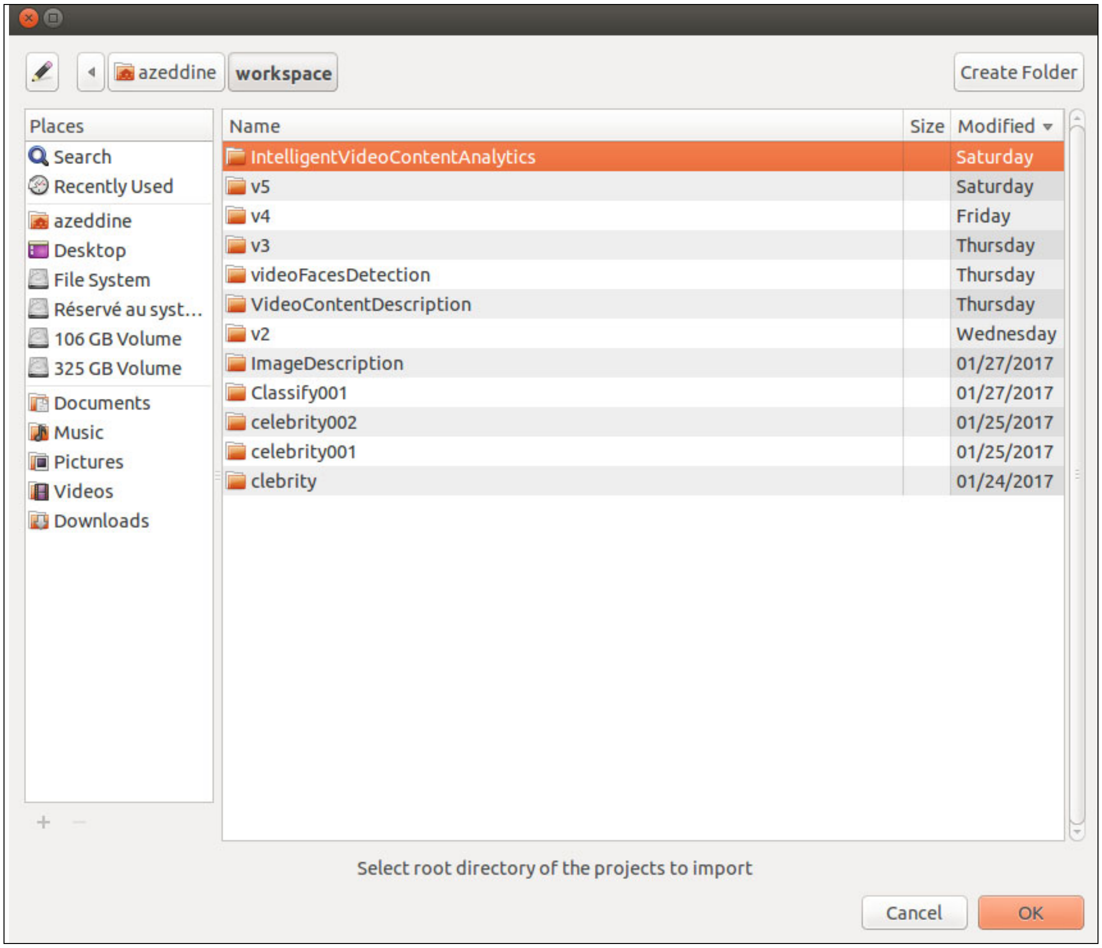
Working sets

□ Add project to working sets

Working sets:

Figure 4-9 Select root directory

5. Find the `IntelligentVideoContentAnalytics` folder (Figure 4-10), and then click **OK**.

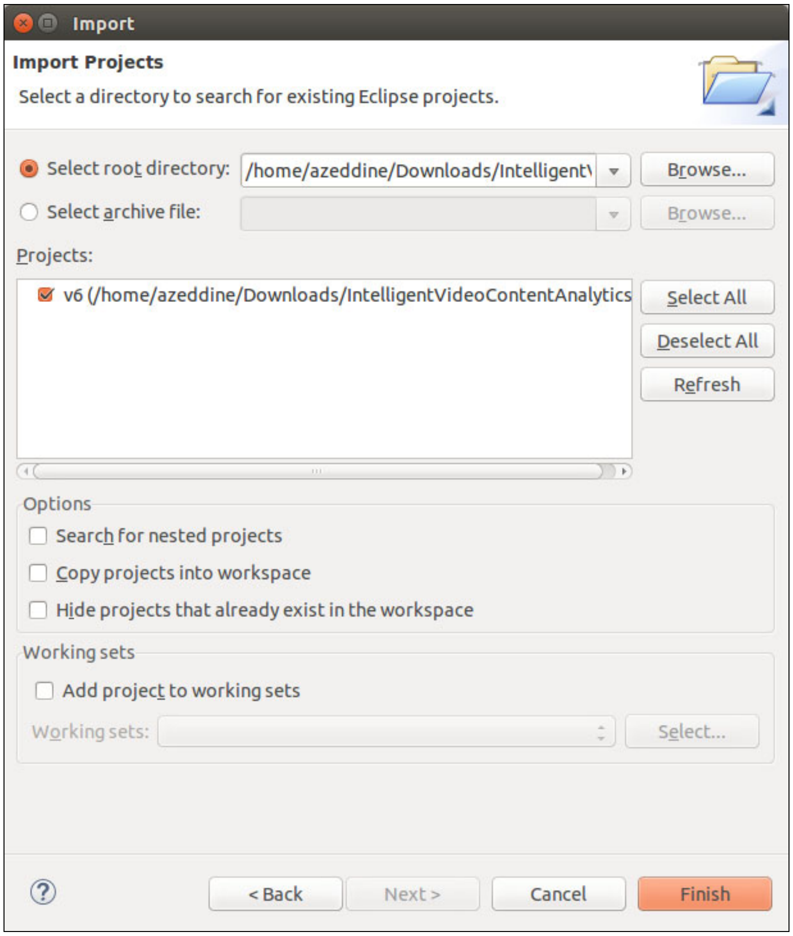


Name	Size	Modified
IntelligentVideoContentAnalytics		Saturday
v5		Saturday
v4		Friday
v3		Thursday
videoFacesDetection		Thursday
VideoContentDescription		Thursday
v2		Wednesday
ImageDescription		01/27/2017
Classify001		01/27/2017
celebrity002		01/25/2017
celebrity001		01/25/2017
clebrity		01/24/2017

Select root directory of the projects to import

Figure 4-10 Navigation window to import project

6. Under Projects, select the **IntelligentVideoContentAnalytics** check box and click **Finish** (Figure 4-11).



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: /home/azeddine/Downloads/Intelligent\

○ Select archive file:

Projects:

☑ v6 (/home/azeddine/Downloads/IntelligentVideoContentAnalytics

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

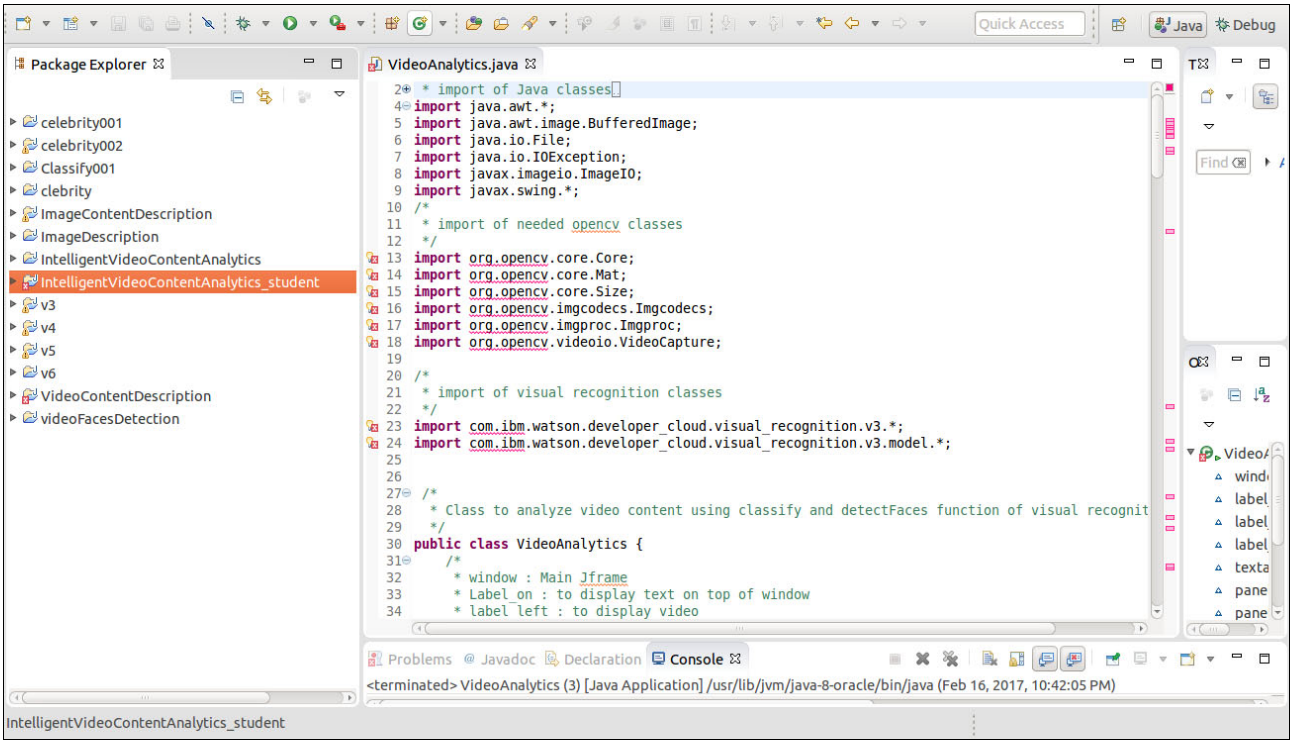
Working sets

□ Add project to working sets

Working sets:

Figure 4-11 Last import project dialog

7. Verify that the IntelligentVideoContentAnalytics project folder is imported to Eclipse Package Explorer (Figure 4-12) and explore its structure (for more details, see the `README.txt` file).



```
 * import of Java classes
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.swing.*;
/*
 * import of needed opencv classes
 */
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.videoio.VideoCapture;

/*
 * import of visual recognition classes
 */
import com.ibm.watson.developer_cloud.visual_recognition.v3.*;
import com.ibm.watson.developer_cloud.visual_recognition.v3.model.*;


 /*
  * Class to analyze video content using classify and detectFaces function of visual recognit
  */
public class VideoAnalytics {
    /*
     * window : Main Jframe
     * Label_on : to display text on top of window
     * label left : to display video
```

```
<terminated> VideoAnalytics (3) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Feb 16, 2017, 10:42:05 PM)
```

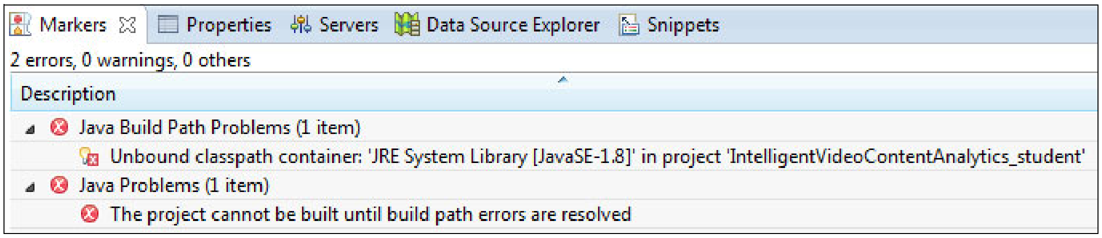
Figure 4-12 Eclipse Package Explorer dialog

4.3.4 Importing Watson Java SDK and additional OpenCV libraries

You might notice some errors when you import the source code. Correcting those errors requires adding an extra dependency and libraries.

Fix Java problems

Figure 4-13 shows Java problems that you might see.



2 errors, 0 warnings, 0 others

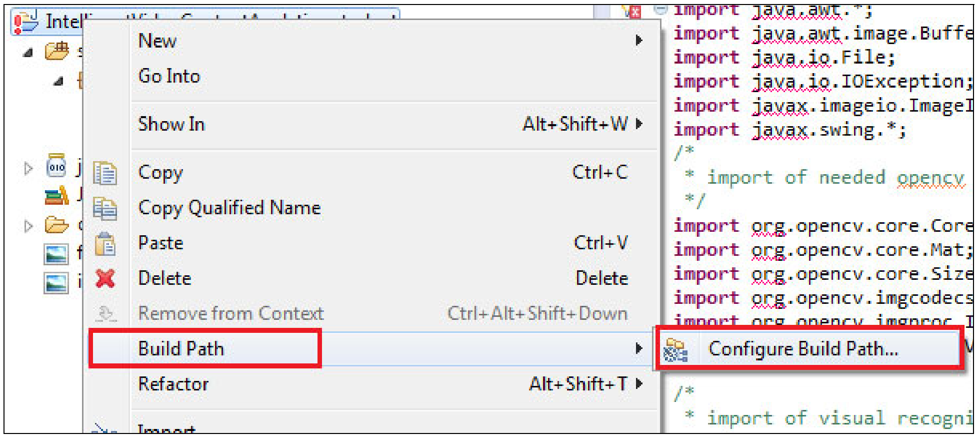
Description

- Java Build Path Problems (1 item)
  - Unbound classpath container: 'JRE System Library [JavaSE-1.8]' in project 'IntelligentVideoContentAnalytics_student'
- Java Problems (1 item)
  - The project cannot be built until build path errors are resolved

Figure 4-13 Java problems

To correct the problem, complete these steps:

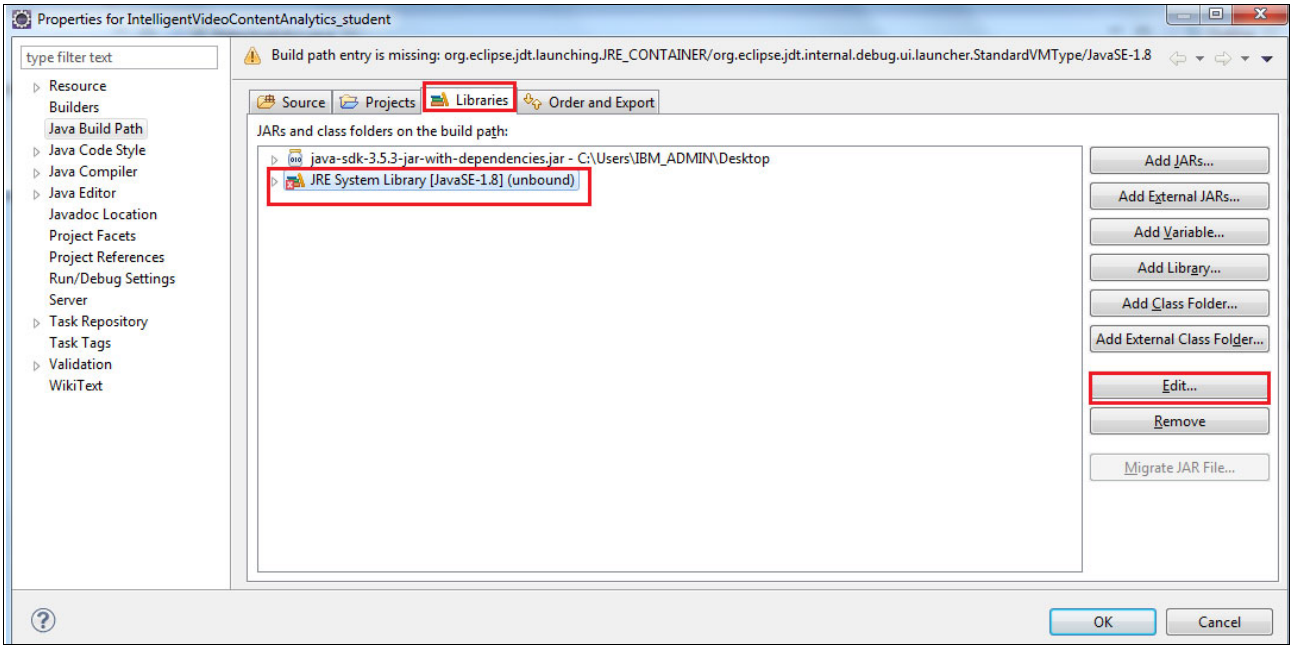
1. Right-click the **IntelligentVideoContentAnalytics** project, and select **Build Path** → **Configure Build Path** (Figure 4-14).



```
import java.awt.*;
import java.awt.image.Buffe
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageI
import javax.swing.*;
/*
 * import of needed opencv
 */
import org.opencv.core.Core
import org.opencv.core.Mat;
import org.opencv.core.Size
import org.opencv.imgcodecs
import org.opencv.imgproc.I

/*
 * import of visual recogni
```

Figure 4-14 Configure Build Path

2. Select the **Libraries** tab, click the library that shows errors, and click **Edit** (Figure 4-15).



Properties for IntelligentVideoContentAnalytics_student

Build path entry is missing: org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/JavaSE-1.8

JARs and class folders on the build path:

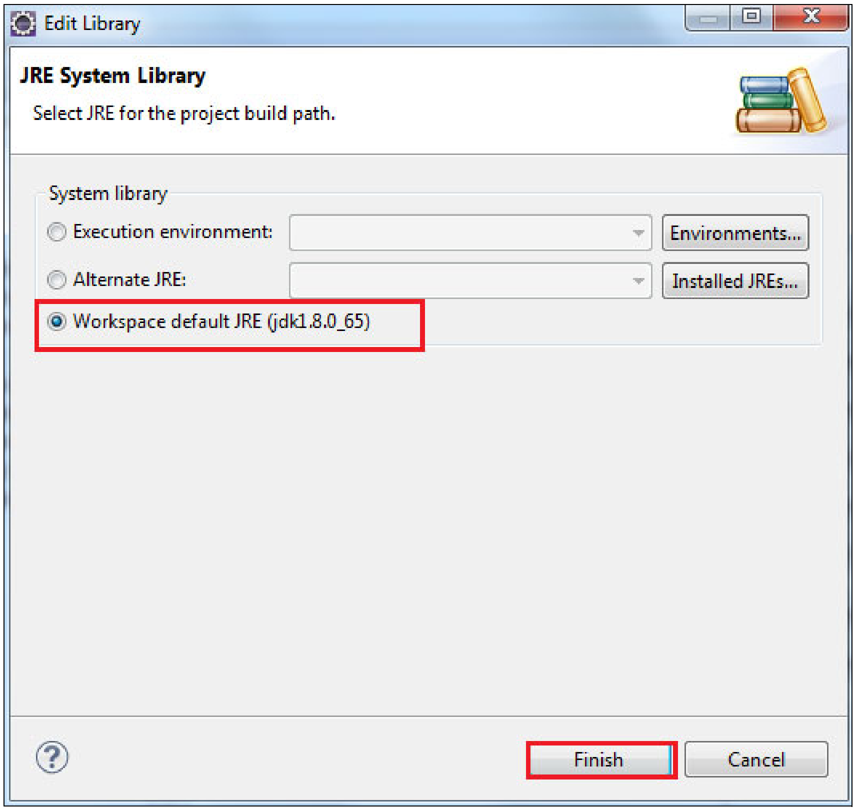
- java-sdk-3.5.3-jar-with-dependencies.jar - C:\Users\IBM_ADMIN\Desktop
- JRE System Library [JavaSE-1.8] (unbound)

Figure 4-15 Select the library in error

3. Do one of the following steps:

   – If *no* default JRE was previously defined: Skip to step 4 on page 97.

   – If a default JRE was previously defined: Select **Workspace default JRE**, and click **Finish** (Figure 4-16). You can now skip to “Add Watson Java SDK with dependencies to your project” on page 101.



Edit Library

JRE System Library

Select JRE for the project build path.

System library

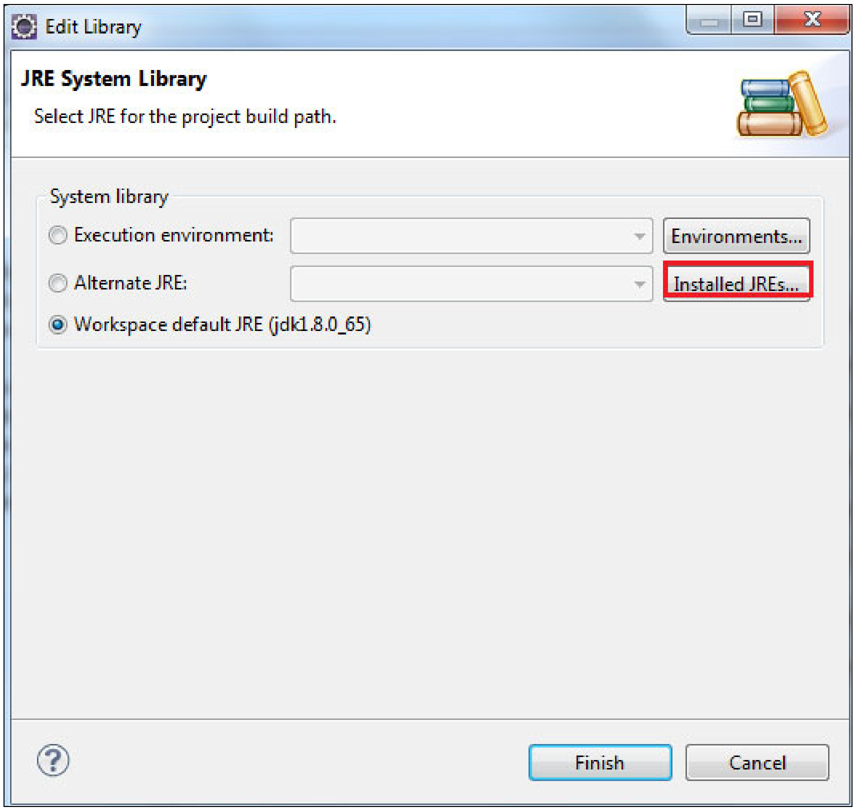
○ Execution environment:

○ Alternate JRE:

● Workspace default JRE (jdk1.8.0_65)

Figure 4-16 Select Workspace default JRE, if one was previously defined

4. This step through step 9 on page 101 are needed *only* if no default JRE was installed previously. Click **Installed JREs** (Figure 4-17).



Edit Library

JRE System Library

Select JRE for the project build path.

System library

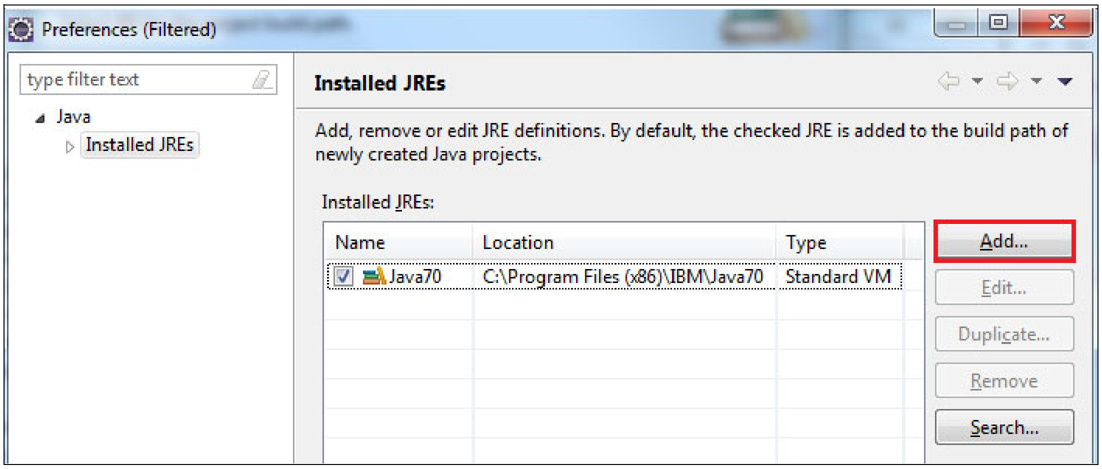
○ Execution environment:

○ Alternate JRE:

● Workspace default JRE (jdk1.8.0_65)

Figure 4-17 Installed JREs

5. Click **Add** (Figure 4-18).



Preferences (Filtered)

Installed JREs

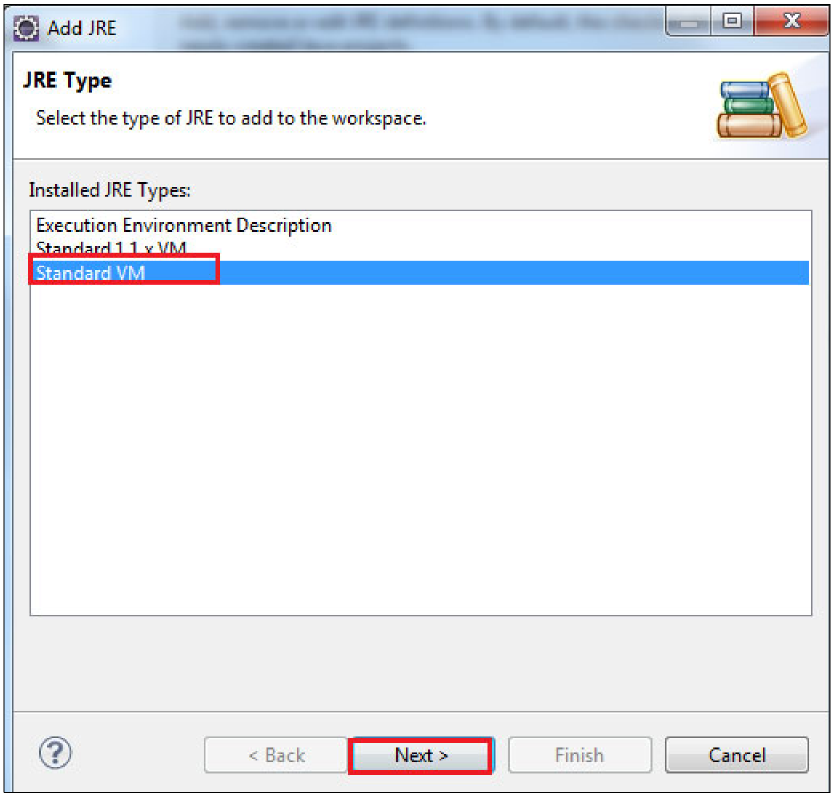
Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

Name	Location	Type
☑ Java70	C:\Program Files (x86)\IBM\Java70	Standard VM

Figure 4-18 Add a JRE definition

6. Select **Standard VM** and click **Next** (Figure 4-19).



Add JRE

JRE Type

Select the type of JRE to add to the workspace.

Installed JRE Types:

Execution Environment Description
Standard 1.1.x VM
Standard VM

Figure 4-19 Standard VM installed JRE type

7. Click **Directory**, select a JDK installation path, and click **OK** (Figure 4-20).

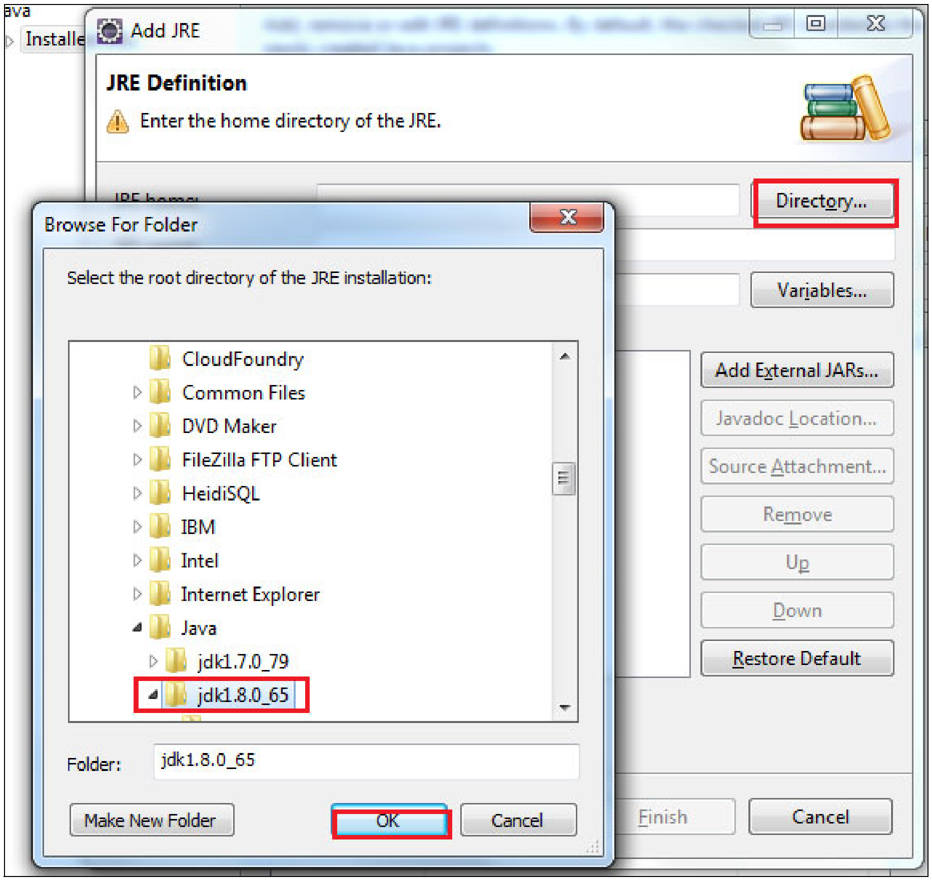
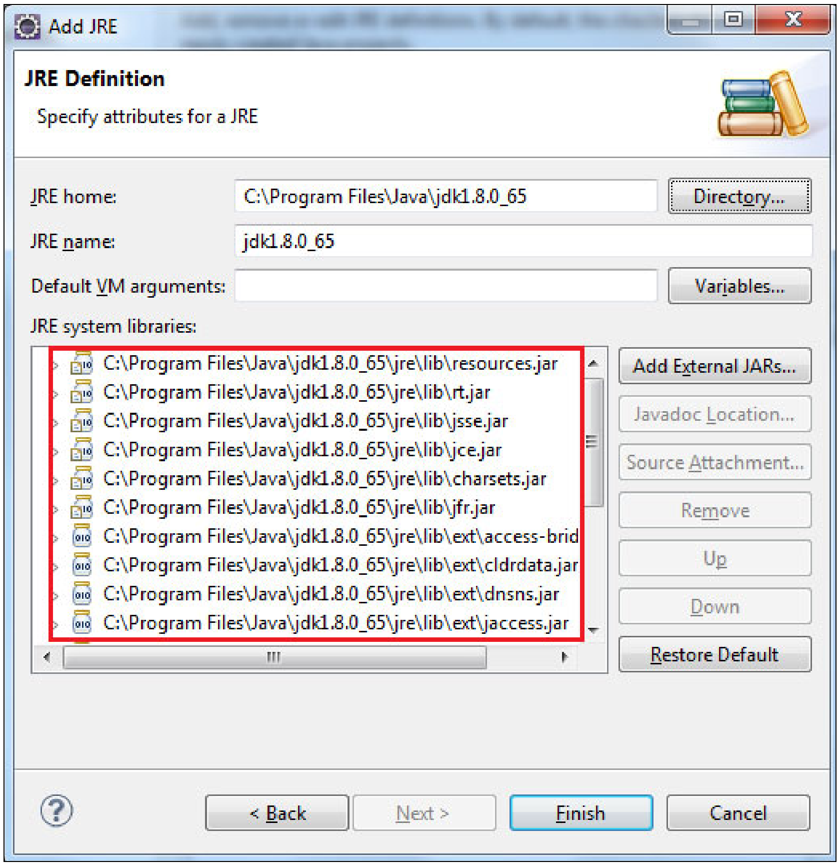


Figure 4-20 Select root directory of JRE installation

8. Your panel should look similar to the one shown in Figure 4-21. Click **Finish**.



Add JRE

JRE Definition

Specify attributes for a JRE

JRE home: C:\Program Files\Java\jdk1.8.0_65

JRE name: jdk1.8.0_65

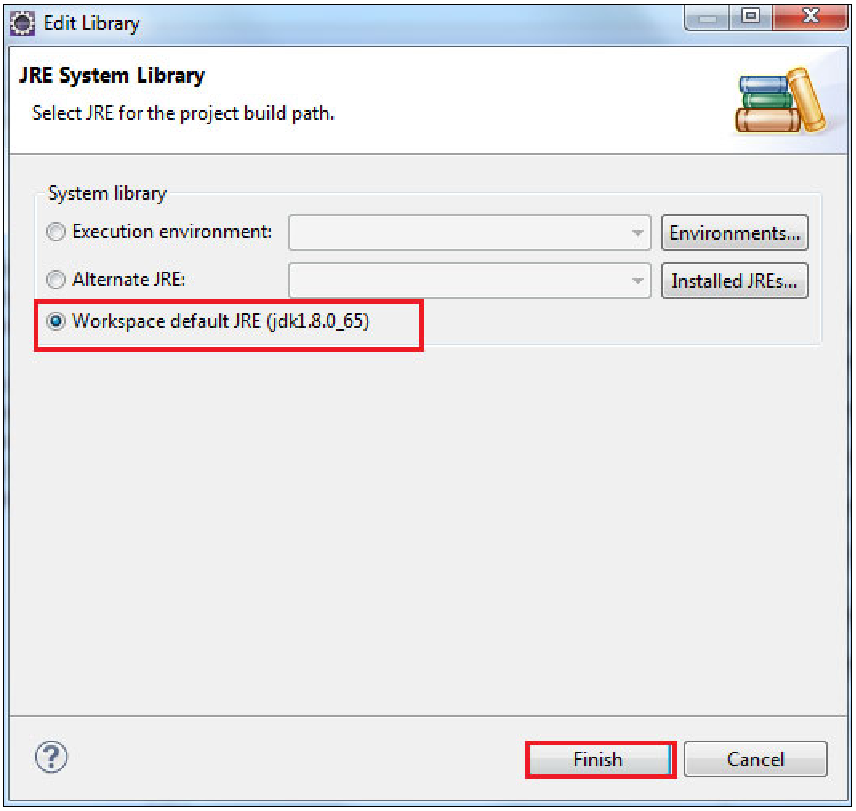
Default VM arguments:

JRE system libraries:

C:\Program Files\Java\jdk1.8.0_65\jre\lib\resources.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\rt.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\jsse.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\jce.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\charsets.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\jfr.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\ext\access-brid
C:\Program Files\Java\jdk1.8.0_65\jre\lib\ext\cldrdata.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\ext\dnsns.jar
C:\Program Files\Java\jdk1.8.0_65\jre\lib\ext\jaccess.jar

Figure 4-21 Sample valid Java library

9. Now you can select **Workspace default JRE**, and click **Finish** (Figure 4-22).



Edit Library

JRE System Library

Select JRE for the project build path.

System library

○ Execution environment:

○ Alternate JRE:

● Workspace default JRE (jdk1.8.0_65)

Figure 4-22 Select Workspace default JRE

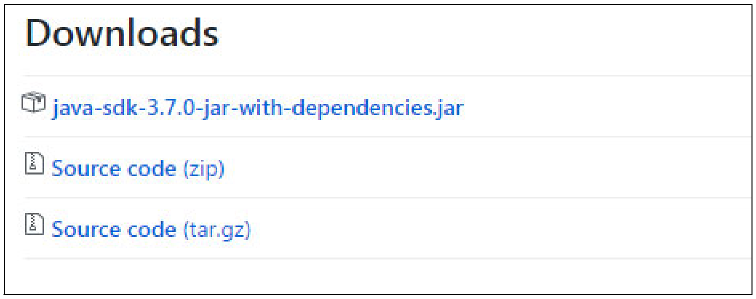
Add Watson Java SDK with dependencies to your project

Complete the following steps:

1. Download the Watson Java SDK dependencies JAR (with dependencies) files:

   `https://github.com/watson-developer-cloud/java-sdk/releases`

2. Scroll to the **Downloads** section and click **java-sdk-3.7.0-jar-with-dependencies.jar** (Figure 4-23).



Downloads

- java-sdk-3.7.0-jar-with-dependencies.jar
- Source code (zip)
- Source code (tar.gz)

Figure 4-23 Download Watson Java SDK

3. After the JAR file is downloaded, open Eclipse, right-click the project name, and then select **Build Path** → **Configure Build Path** (Figure 4-24).

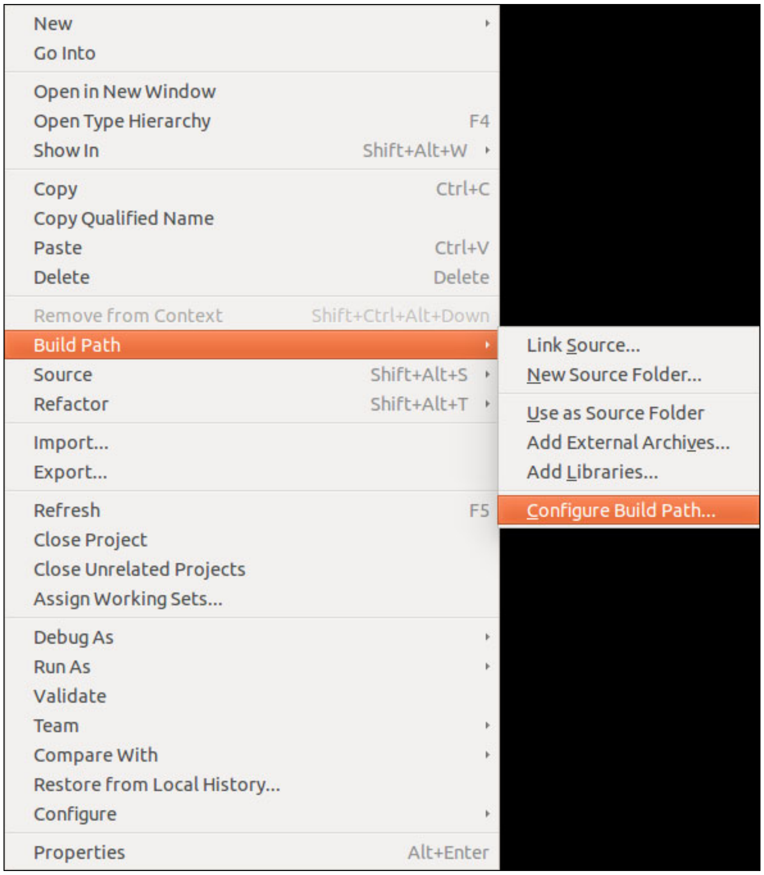
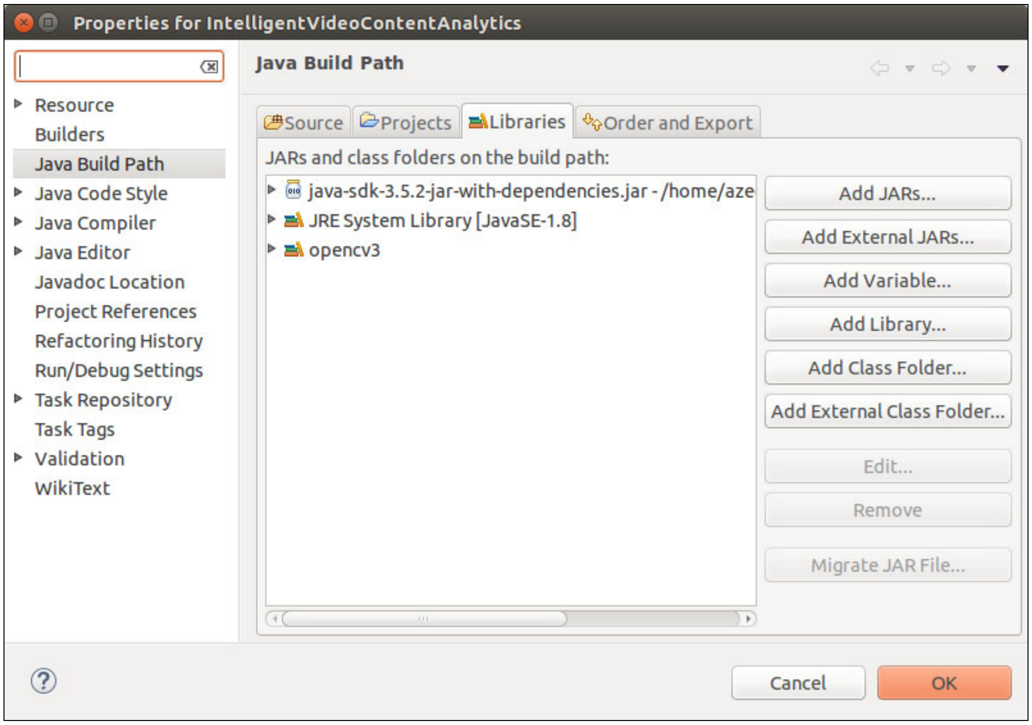


Figure 4-24 Configure Build Path

4. Open the **Libraries** tab, and then click **Add External JARs** (Figure 4-25).



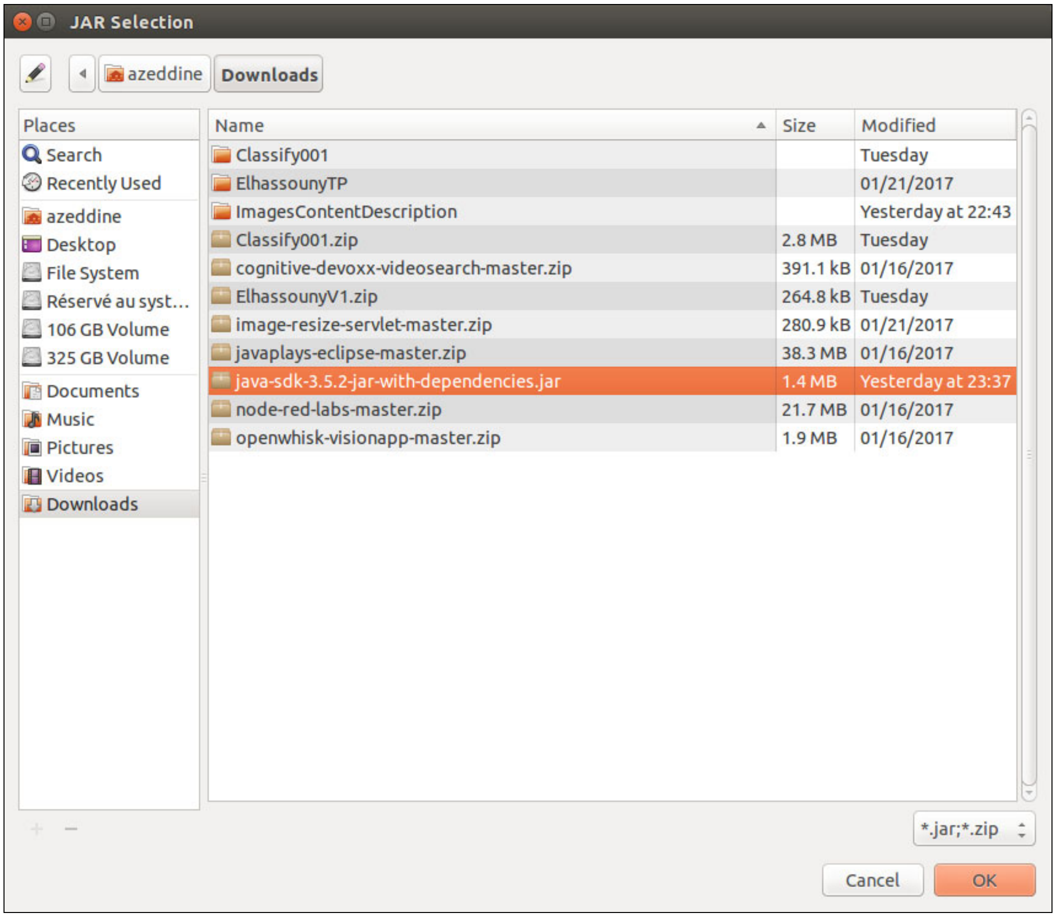
JARs and class folders on the build path:

- java-sdk-3.5.2-jar-with-dependencies.jar - /home/aze
- JRE System Library [JavaSE-1.8]
- opencv3

Figure 4-25 Configure Java Build Path

5. Navigate to the JAR file (`java-sdk-3.5.2-jar-with-dependencies.jar`), select it, and then click **OK** (Figure 4-26).

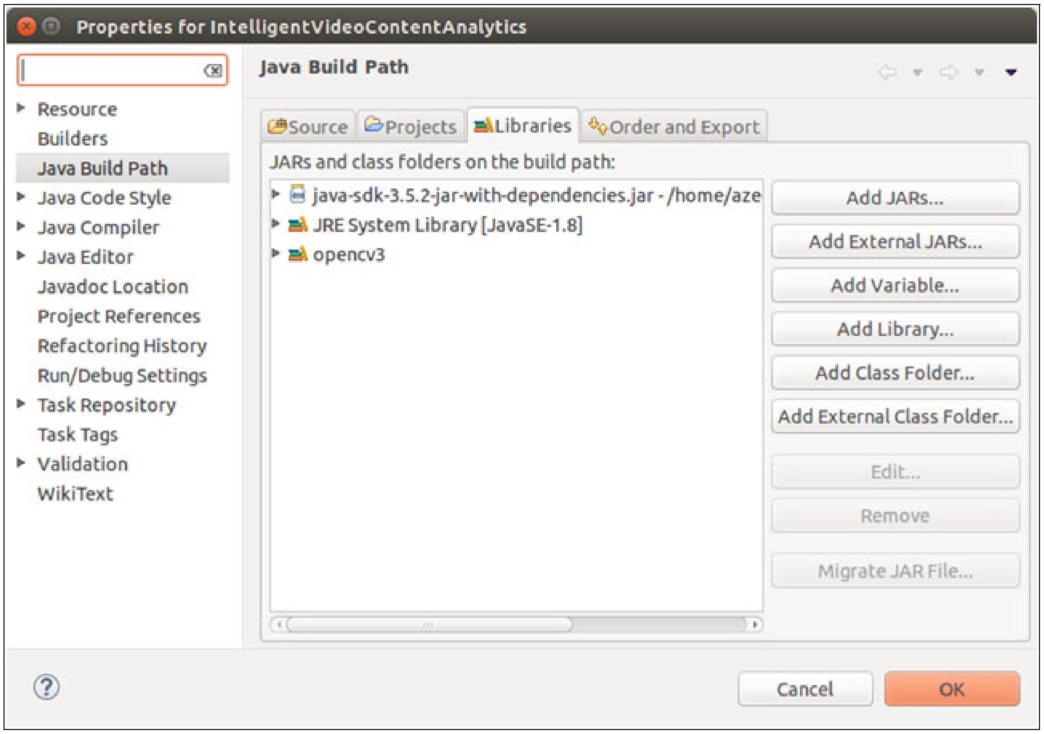
Note: The JAR file name (`java-sdk-x.x.x-jar-with-dependencies.jar`) will vary depending on the version available when you download it.



Name	Size	Modified
Classify001		Tuesday
ElhassounyTP		01/21/2017
ImagesContentDescription		Yesterday at 22:43
Classify001.zip	2.8 MB	Tuesday
cognitive-devoxx-videosearch-master.zip	391.1 kB	01/16/2017
ElhassounyV1.zip	264.8 kB	Tuesday
image-resize-servlet-master.zip	280.9 kB	01/21/2017
javaplays-eclipse-master.zip	38.3 MB	01/16/2017
java-sdk-3.5.2-jar-with-dependencies.jar	1.4 MB	Yesterday at 23:37
node-red-labs-master.zip	21.7 MB	01/16/2017
openwhisk-visionapp-master.zip	1.9 MB	01/16/2017

Figure 4-26 Select the Java SDK JAR file

6. Check that the JAR file is added to your project and click **OK** (Figure 4-27).

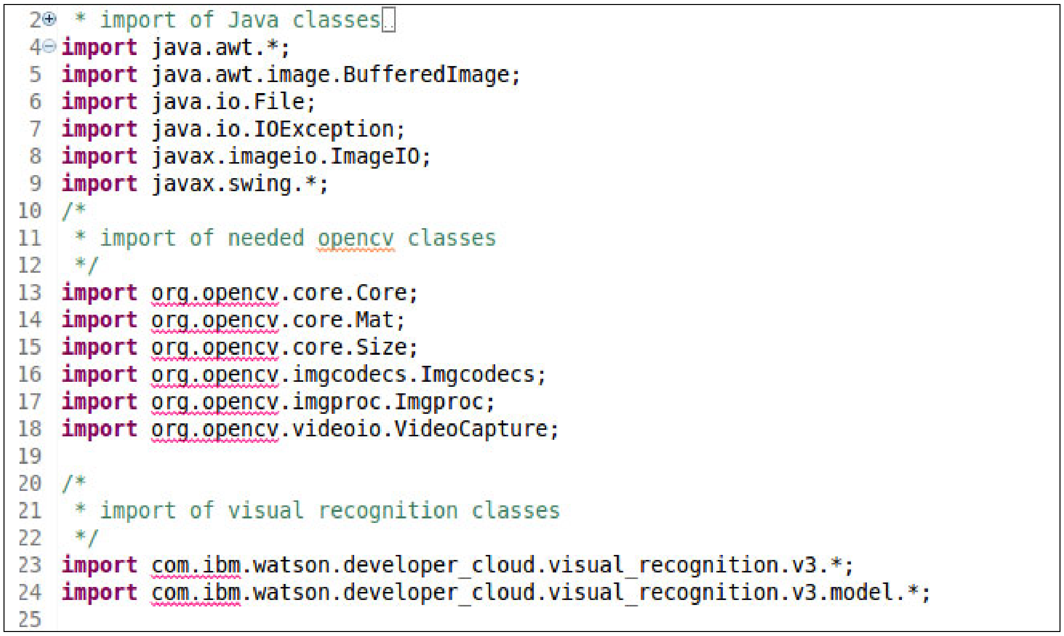


JARs and class folders on the build path:

- java-sdk-3.5.2-jar-with-dependencies.jar - /home/aze
- JRE System Library [JavaSE-1.8]
- opencv3

Figure 4-27 Window to check the addition of Java SDK

7. After the Watson Java SDK is imported to the project, verify that the Java errors concerning Visual Recognition are resolved (as shown in lines 23 and 24 of Figure 4-28).



```
 2⊕ * import of Java classes
 4⊖import java.awt.*;
 5 import java.awt.image.BufferedImage;
 6 import java.io.File;
 7 import java.io.IOException;
 8 import javax.imageio.ImageIO;
 9 import javax.swing.*;
10 /*
11  * import of needed opencv classes
12  */
13 import org.opencv.core.Core;
14 import org.opencv.core.Mat;
15 import org.opencv.core.Size;
16 import org.opencv.imgcodecs.Imgcodecs;
17 import org.opencv.imgproc.Imgproc;
18 import org.opencv.videoio.VideoCapture;
19
20 /*
21  * import of visual recognition classes
22  */
23 import com.ibm.watson.developer_cloud.visual_recognition.v3.*;
24 import com.ibm.watson.developer_cloud.visual_recognition.v3.model.*;
25
```

Figure 4-28 Import of Visual Recognition classes

Create OpenCV3.x.x Java as a user library to Eclipse

To resolve import errors of OpenCV, define OpenCV as a user library in Eclipse. Complete the following steps:

> **Note:** These steps are from the Using OpenCV Java with Eclipse web page.

1. After the OpenCV3.x.x Java library is installed, return to Eclipse and select **Window** → **Preferences** (Figure 4-29).

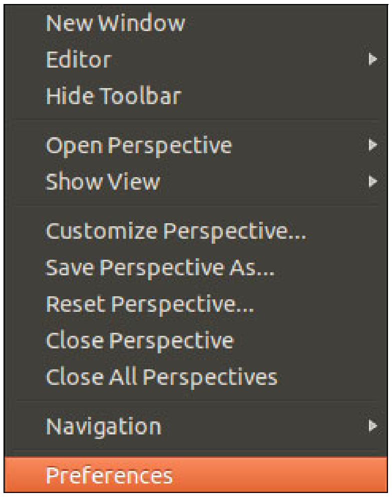
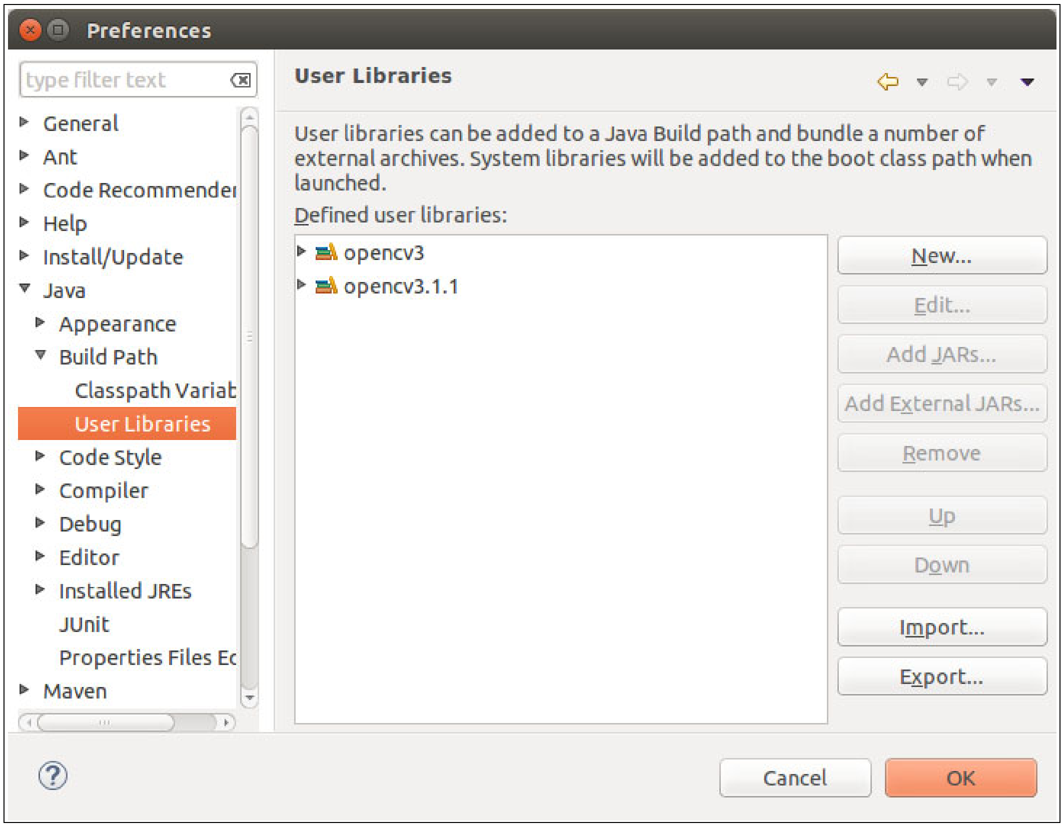


Figure 4-29 Select Preferences

2. Expand **Java** → **Build Path** → **User Libraries** and click **New** (Figure 4-30 on page 107).



User Libraries

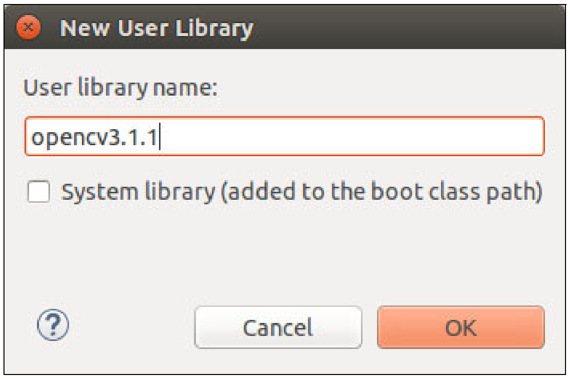
User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

Defined user libraries:

- opencv3
- opencv3.1.1

Figure 4-30 Add new user library

3. Provide a name for your new user library, for example `opencv3.x.x` (Figure 4-31), and then click **OK**.



User library name: opencv3.1.1

☐ System library (added to the boot class path)

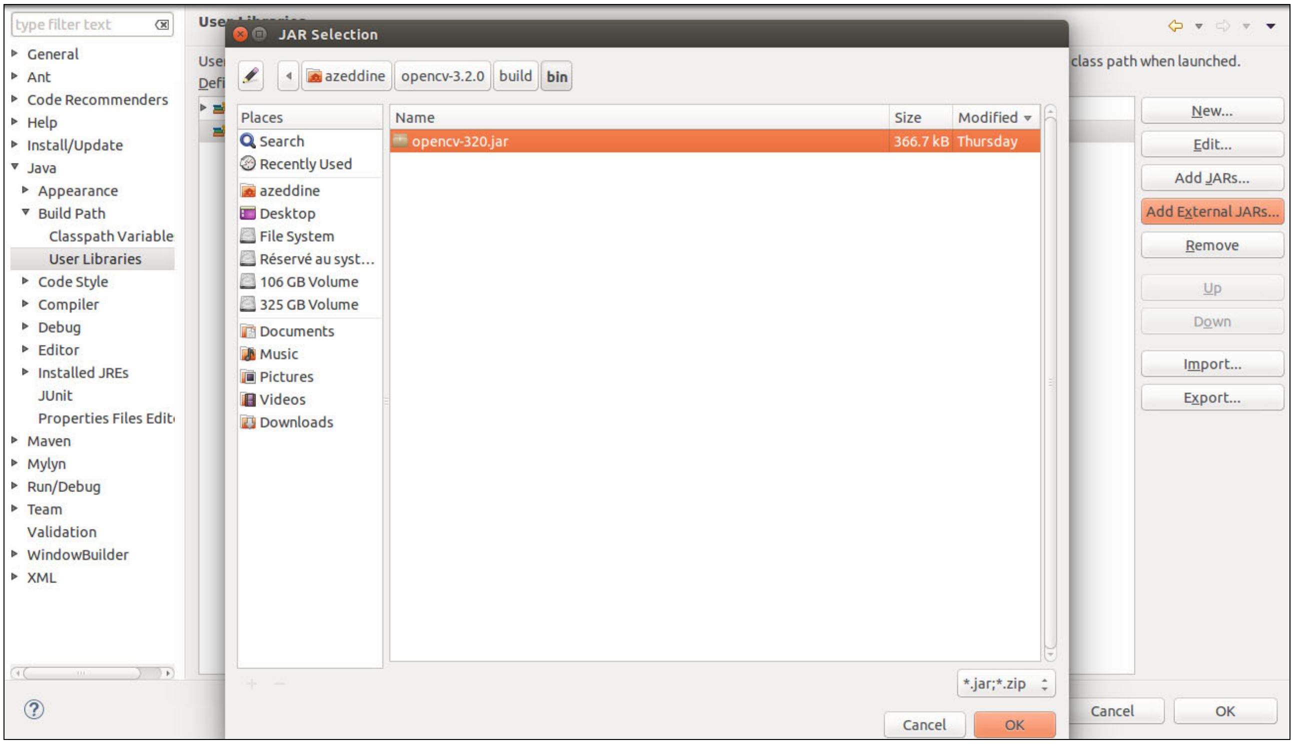
Figure 4-31 Fill user library name dialog

4. Select your new user library (`opencv3.x.x`) and click **Add External JARs**. A dialog opens where you can navigate folders (Figure 4-32 on page 108) to find the `opencv-3xx.jar` file.

   Select the `opencv-3xx.jar` file that is in the installation folder of OpenCV library.The location of the JAR file depends on the operating system you use:

   – For Linux: `/opencv3.x.x/build/bin/`
   – For Windows: `C:\OpenCV-3.x.x\build\java\x64` (or x86 if you have a 32-bit OS)

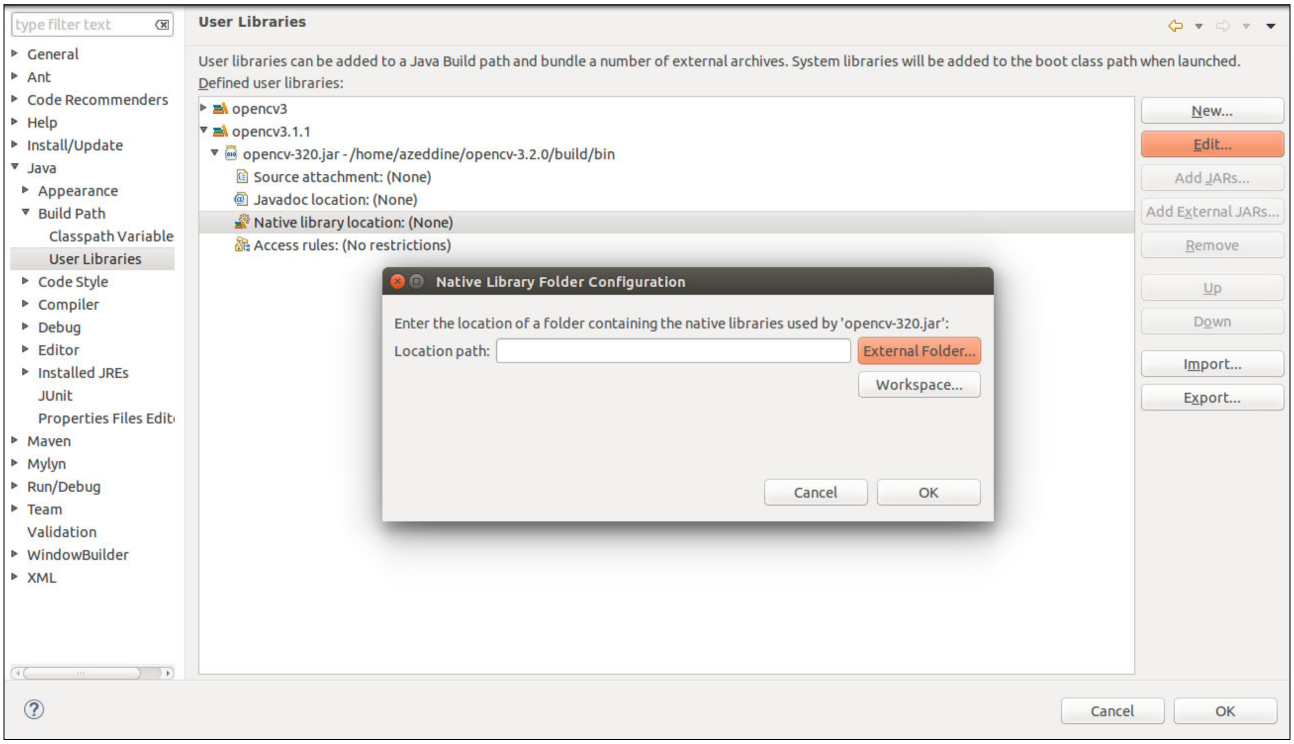
   After you select the `opencv-3xx.jar`, click **OK**.



Name	Size	Modified
opencv-320.jar	366.7 kB	Thursday

Figure 4-32 Navigate folders dialog

5. Select **Native library location** and click **Edit**. The Native Library Folder Configuration dialog opens (Figure 4-33).



User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

Defined user libraries:

- opencv3
- opencv3.1.1
  - opencv-320.jar - /home/azeddine/opencv-3.2.0/build/bin
    - Source attachment: (None)
    - Javadoc location: (None)
    - Native library location: (None)
    - Access rules: (No restrictions)

Enter the location of a folder containing the native libraries used by 'opencv-320.jar':

Location path:

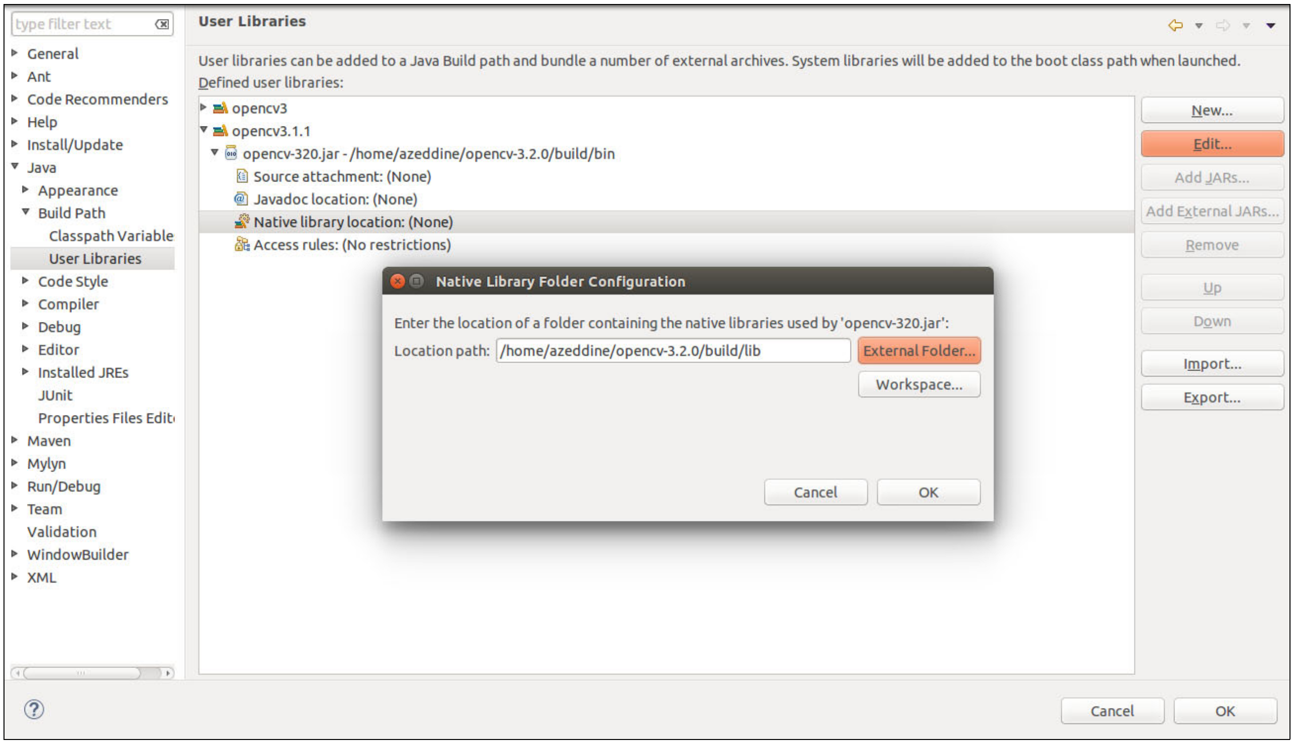
Figure 4-33 Native Library Folder Configuration dialog

6. Click **External Folder** and browse to select the folder of the Native Library Location:

   – For Linux: `/opencv3.x.x/build/lib`

   – For Windows: `C:\OpenCV-3.x.x\build\java\x64` (if you have a 32-bit OS, select the **x86** folder instead **x64**).

   After the OpenCV Native Library Location is determined, click **OK** on the Native Library Folder Configuration dialog and then click **OK** on the User Libraries page (Figure 4-34).



User Libraries

User libraries can be added to a Java Build path and bundle a number of external archives. System libraries will be added to the boot class path when launched.

Defined user libraries:

```
▸ opencv3
▾ opencv3.1.1
  ▾ opencv-320.jar - /home/azeddine/opencv-3.2.0/build/bin
      Source attachment: (None)
      Javadoc location: (None)
      Native library location: (None)
      Access rules: (No restrictions)
```

Native Library Folder Configuration

Enter the location of a folder containing the native libraries used by 'opencv-320.jar':

Location path: /home/azeddine/opencv-3.2.0/build/lib

Figure 4-34 Native library folder configuration dialog

7. After you add the OpenCV library, right-click the project name and select **Build Path** → **Configure Build Path** (Figure 4-35 on page 110).

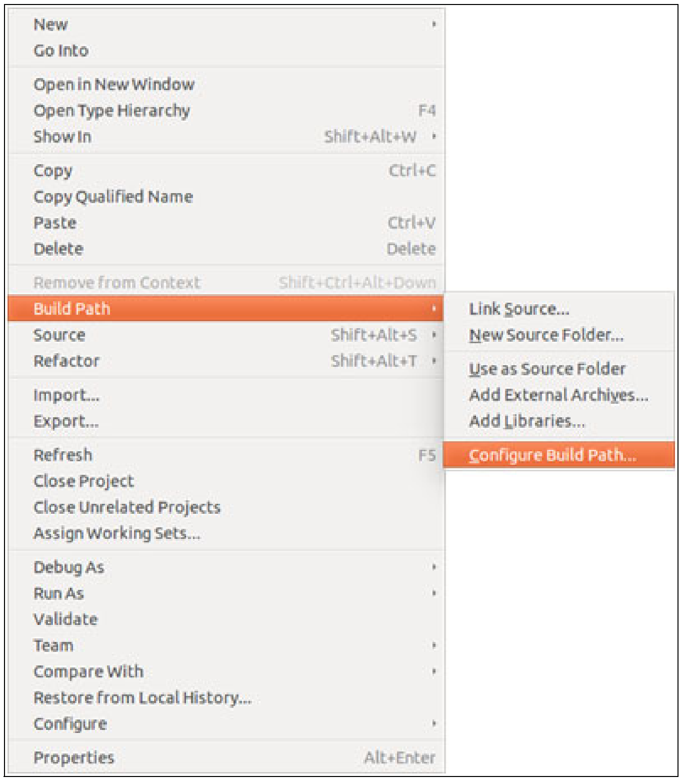


Figure 4-35 Configure Build Path

8. Click the **Libraries** tab and click **Add Library** to open the Add Library wizard (Figure 4-36).

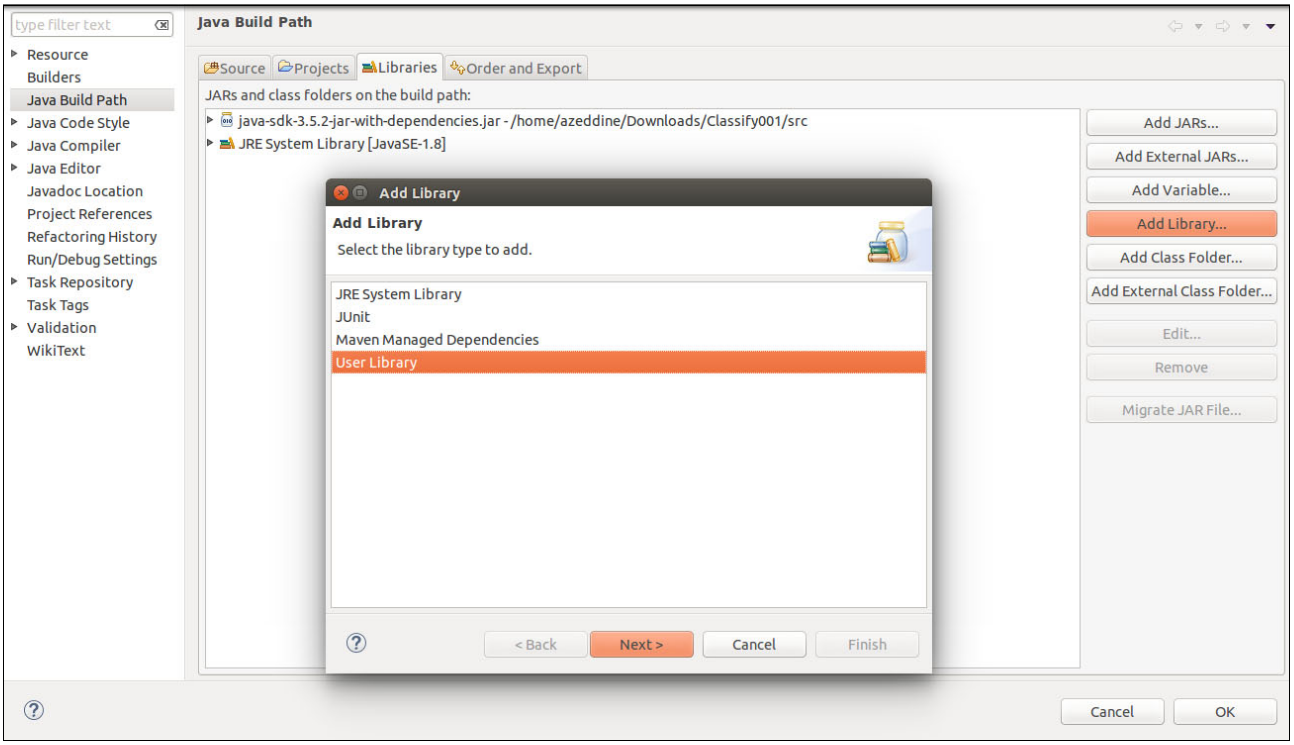
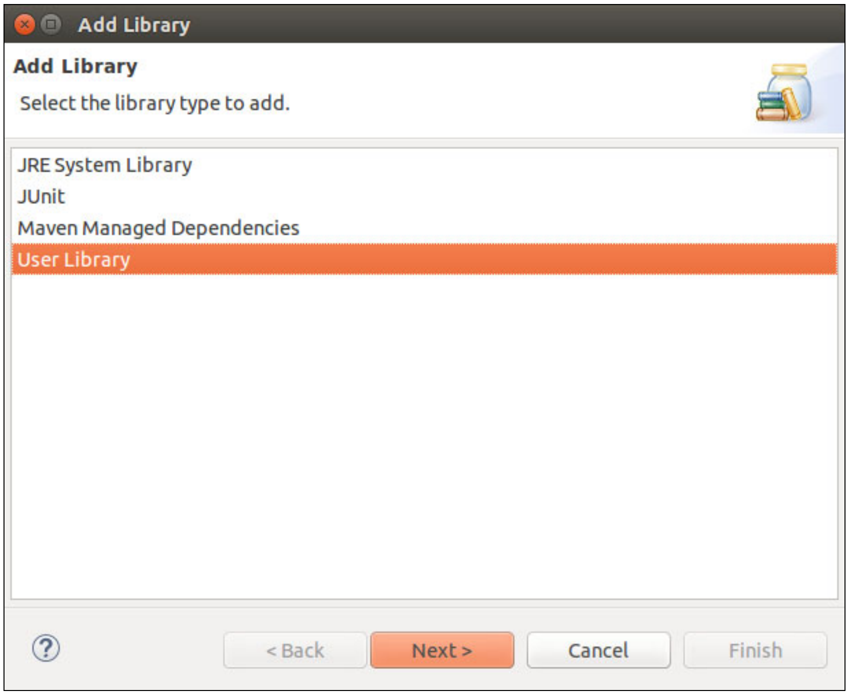


Figure 4-36 Add Library window

9. Select **User Library** and click **Next** (Figure 4-37).

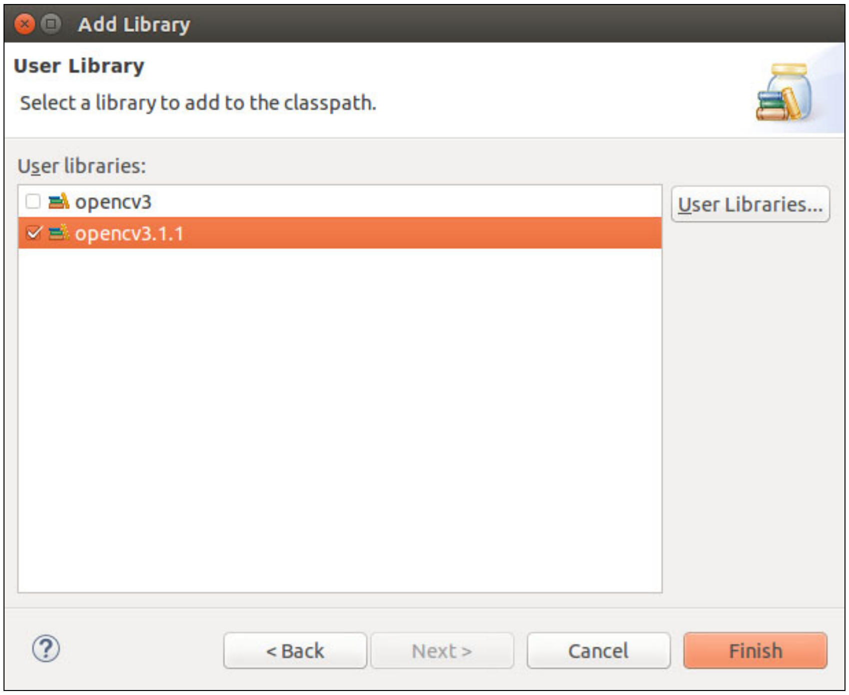

Add Library

Add Library

Select the library type to add.

Figure 4-37 Add Library dialog

10. Select the **opencv3.x.x** check box and click **Finish** (Figure 4-38).


Add Library

User Library

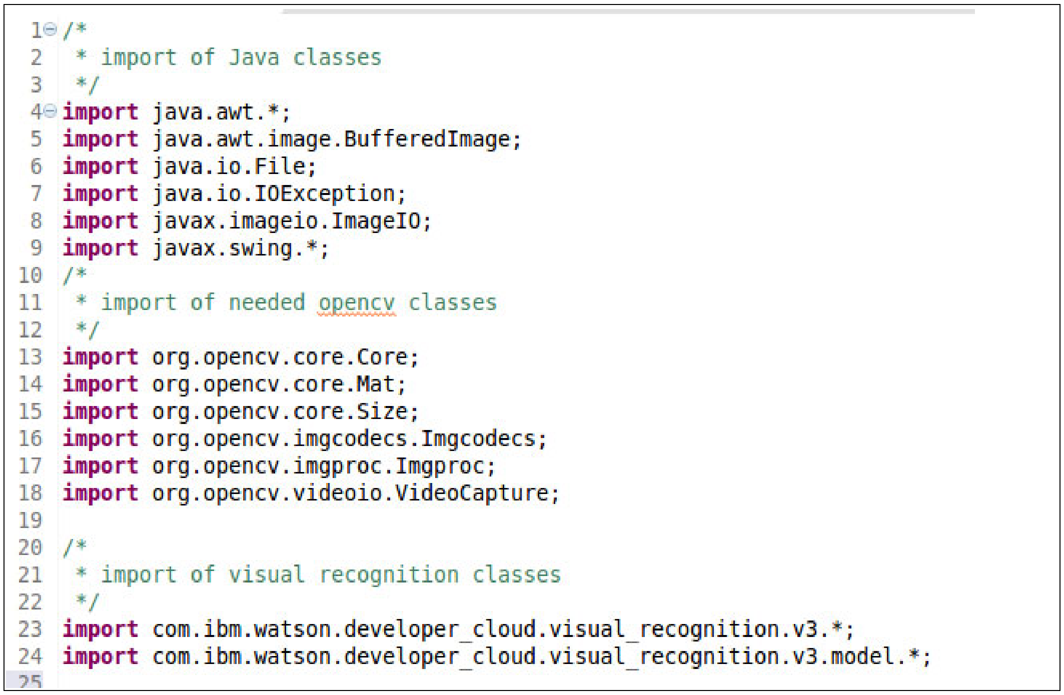
Select a library to add to the classpath.

User libraries:

- □ opencv3
- ☑ opencv3.1.1

Figure 4-38 Add Library dialog

11. Now that all required libraries are added, verify that no import errors exist (Figure 4-39).



```
/*
 * import of Java classes
 */
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import javax.imageio.ImageIO;
import javax.swing.*;
/*
 * import of needed opencv classes
 */
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.videoio.VideoCapture;

/*
 * import of visual recognition classes
 */
import com.ibm.watson.developer_cloud.visual_recognition.v3.*;
import com.ibm.watson.developer_cloud.visual_recognition.v3.model.*;
```

Figure 4-39   No errors

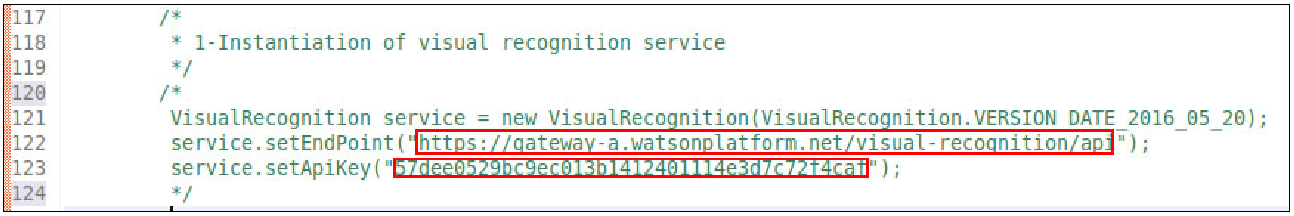
4.3.5 Exploring and completing the sample code provided with the use case

You imported the project and resolved the import errors. Now you can use the Java editor in Eclipse to explore and understand the code and make a few changes to the source code in order to complete it. These steps focus mainly on removing comments around several key instructions and customizing the program with your Watson Visual Recognition service credentials.

1. The starting point of the execution of a stand-alone Java program is the `main` method. Figure 4-40 shows a snippet of the main method.

   Update the code: Remove the block comment around the three first instructions (lines 121, 122, and 123 in Figure 4-40).

   On line 121 the VisualRecognition class is instantiated. This Java class is used to access the Watson Visual Recognition service.



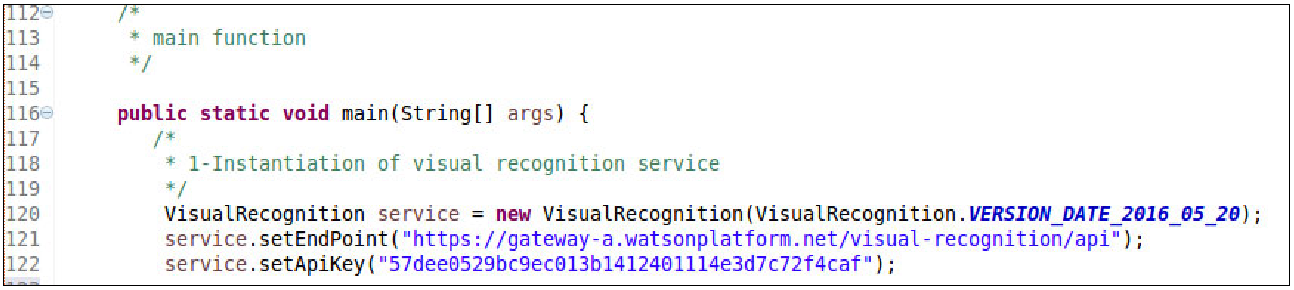
```
117          /*
118           * 1-Instantiation of visual recognition service
119           */
120          /*
121           VisualRecognition service = new VisualRecognition(VisualRecognition.VERSION_DATE_2016_05_20);
122           service.setEndPoint("https://gateway-a.watsonplatform.net/visual-recognition/api");
123           service.setApiKey("57dee0529bc9ec013b1412401114e3d7c72f4caf");
124           */
```

Figure 4-40   Instantiation of Visual Recognition service code

2. The first instruction in Figure 4-41 on page 113 instantiates a new `VisualRecognition` object to access the Watson Visual Recognition V3 service.

Update the code: In the next two lines of code (121 and 122), replace the values of the `ApiKey` and `EndPoint` with your values that you copied previously in 4.3.1, “Creating a Visual Recognition service instance” on page 88.

Your code should now appear similar to Figure 4-41.

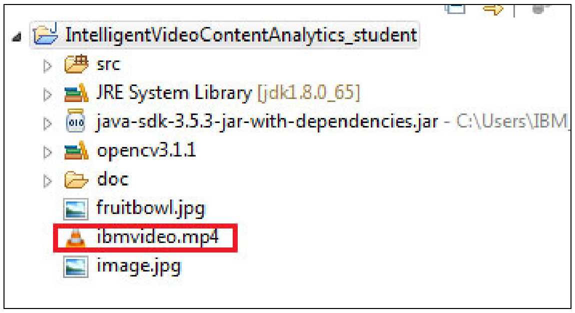


```
112     /*
113      * main function
114      */
115
116     public static void main(String[] args) {
117         /*
118          * 1-Instantiation of visual recognition service
119          */
120         VisualRecognition service = new VisualRecognition(VisualRecognition.VERSION_DATE_2016_05_20);
121         service.setEndPoint("https://gateway-a.watsonplatform.net/visual-recognition/api");
122         service.setApiKey("57dee0529bc9ec013b1412401114e3d7c72f4caf");
```

Figure 4-41   Code overview after removing comments and setting ApiKey and EndPoint URL

3. Copy a video file, for example `ibmvideo.mp4`, to the project file directory (Figure 4-42). You can download the video file from this location:

   https://www.youtube.com/watch?v=fUKpGLk9Ml8&cm_mc_uid=11487496984514811404484&cm_mc_sid_50200000=1487265686

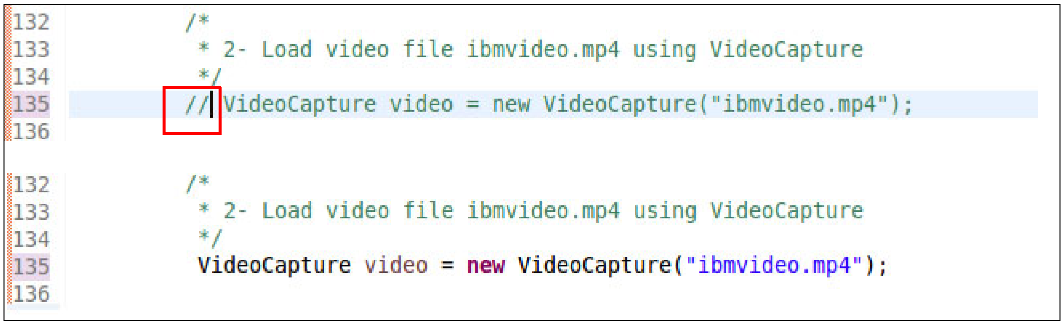


IntelligentVideoContentAnalytics_student
src
JRE System Library [jdk1.8.0_65]
java-sdk-3.5.3-jar-with-dependencies.jar - C:\Users\IBM
opencv3.1.1
doc
fruitbowl.jpg
ibmvideo.mp4
image.jpg

Figure 4-42   A video file in the project directory

4. Now, load the input video file using VideoCapture (an OpenCV class).

Update the code: This simple step involves removing the comment characters in line 135. Figure 4-43 shows how the code looks before and after the change.

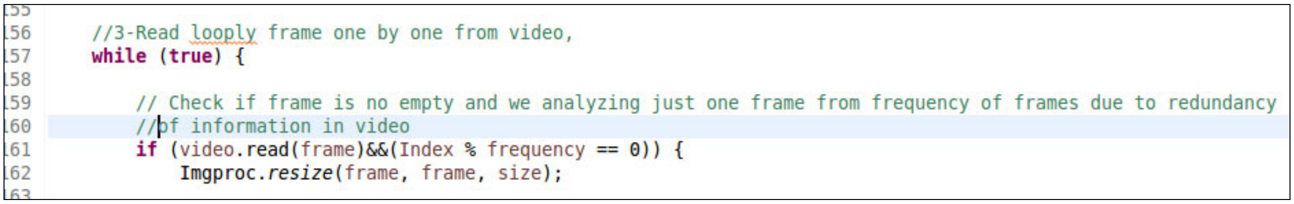


```
132         /*
133          * 2- Load video file ibmvideo.mp4 using VideoCapture
134          */
135         //VideoCapture video = new VideoCapture("ibmvideo.mp4");
136
```

```
132         /*
133          * 2- Load video file ibmvideo.mp4 using VideoCapture
134          */
135         VideoCapture video = new VideoCapture("ibmvideo.mp4");
136
```

Figure 4-43   Load video line code

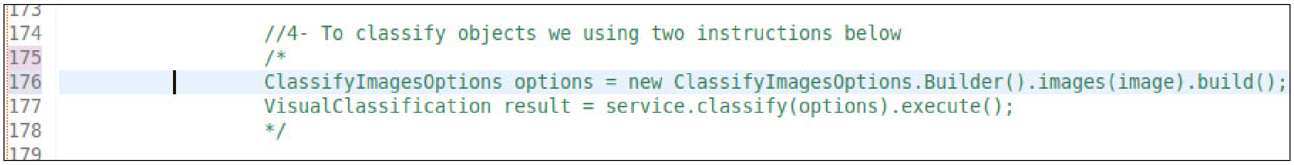
5. A `while` loop reads the video frame by frame (Figure 4-44) and analyzes the video content. Note that the program does not analyze every frame, the main reason being that there is a lot of redundancy in consecutive frames. This sample program analyzes one out of every 40 frames. You can change this by simply updating the `frequency` variable.



```
    //3-Read looply frame one by one from video,
    while (true) {

        // Check if frame is no empty and we analyzing just one frame from frequency of frames due to redundancy
        //of information in video
        if (video.read(frame)&&(Index % frequency == 0)) {
            Imgproc.resize(frame, frame, size);
```

Figure 4-44 while loop to read frames from video

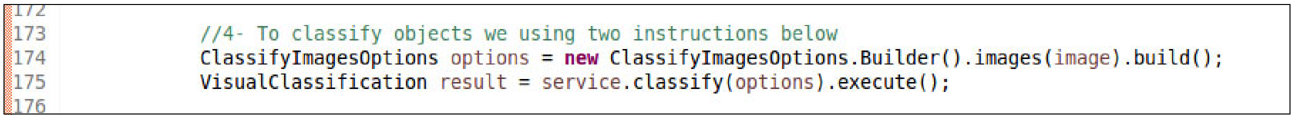
6. Figure 4-45 shows the code that classifies the objects in the video frame.



```
                //4- To classify objects we using two instructions below
                /*
                ClassifyImagesOptions options = new ClassifyImagesOptions.Builder().images(image).build();
                VisualClassification result = service.classify(options).execute();
                */
```

Figure 4-45 Classify objects code with comments

Update the code: Remove the block comments so your code looks identical to Figure 4-46.



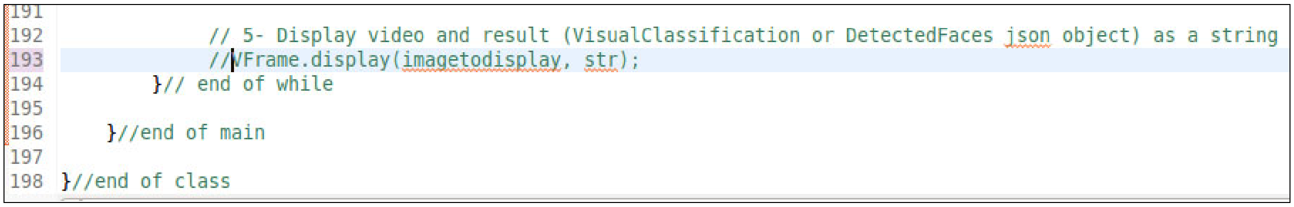
```
            //4- To classify objects we using two instructions below
            ClassifyImagesOptions options = new ClassifyImagesOptions.Builder().images(image).build();
            VisualClassification result = service.classify(options).execute();
```

Figure 4-46 Classify objects code

About the code:

- The first line of code shows how to create a `ClassifyImagesOptions` object based on the current video frame image. Consider this information:
  - To create the new options for the new image, instantiate a new builder (`ClassifyImagesOptions.Builder()`), call the `images()` method to set the new image to classify. This function accepts an image file as the parameter and returns the builder.
  - At the end, call the `build()` method with any argument that builds and returns the profile options (`ClassifyImagesOptions`).
- The second line of code shows how to call the Watson Visual Recognition service that performs the actual classification of objects within the current video frame image. The result of the classification is saved in the `result` variable. Consider this information:
  - To classify an object, call the `classify()` method, `service.classify(ClassifyImagesOptions)`.
  - It accepts options (`ClassifyImagesOptions`) as argument and returns a `VisualClassification` JSON object. The `classify()` method of the Visual Recognition service analyzes images and detects details of objects.
  - The `execute()` function is used to run the service.

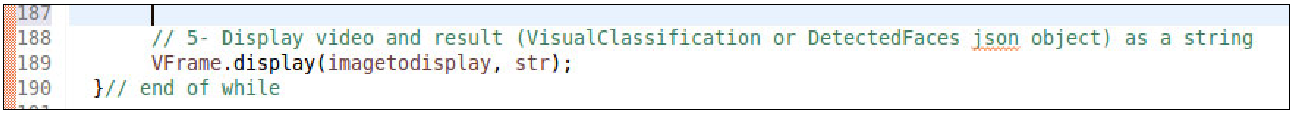
7. Figure 4-47 shows how the sample program calls the `display()` method to display the video frame and the result of the classification, before moving on to the next frame. This method receives two arguments:
   - Frame
   - Frame description (`str = result.toString()`)



```
191
192             // 5- Display video and result (VisualClassification or DetectedFaces json object) as a string
193             //VFrame.display(imagetodisplay, str);
194         }// end of while
195
196     }//end of main
197
198 }//end of class
```

Figure 4-47 Display frame and description objects code

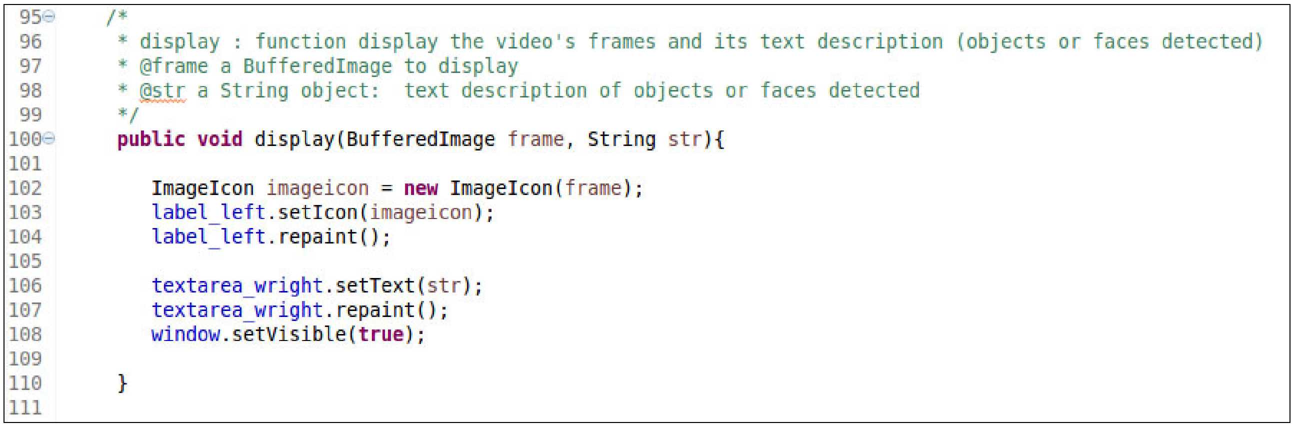
Update the code: As before, remove the comment from line 193. Your code should now look like the code in Figure 4-48.



```

        // 5- Display video and result (VisualClassification or DetectedFaces json object) as a string
        VFrame.display(imagetodisplay, str);
    }// end of while
```

Figure 4-48 Result

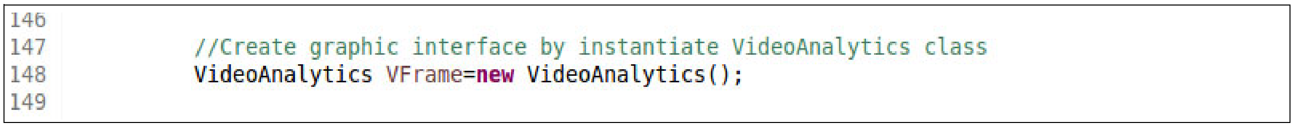
The code of the `display()` method is shown in Figure 4-49.



```
/*
 * display : function display the video's frames and its text description (objects or faces detected)
 * @frame a BufferedImage to display
 * @str a String object:  text description of objects or faces detected
 */
 public void display(BufferedImage frame, String str){

    ImageIcon imageicon = new ImageIcon(frame);
    label_left.setIcon(imageicon);
    label_left.repaint();

    textarea_wright.setText(str);
    textarea_wright.repaint();
    window.setVisible(true);

 }

```

Figure 4-49 The display method code

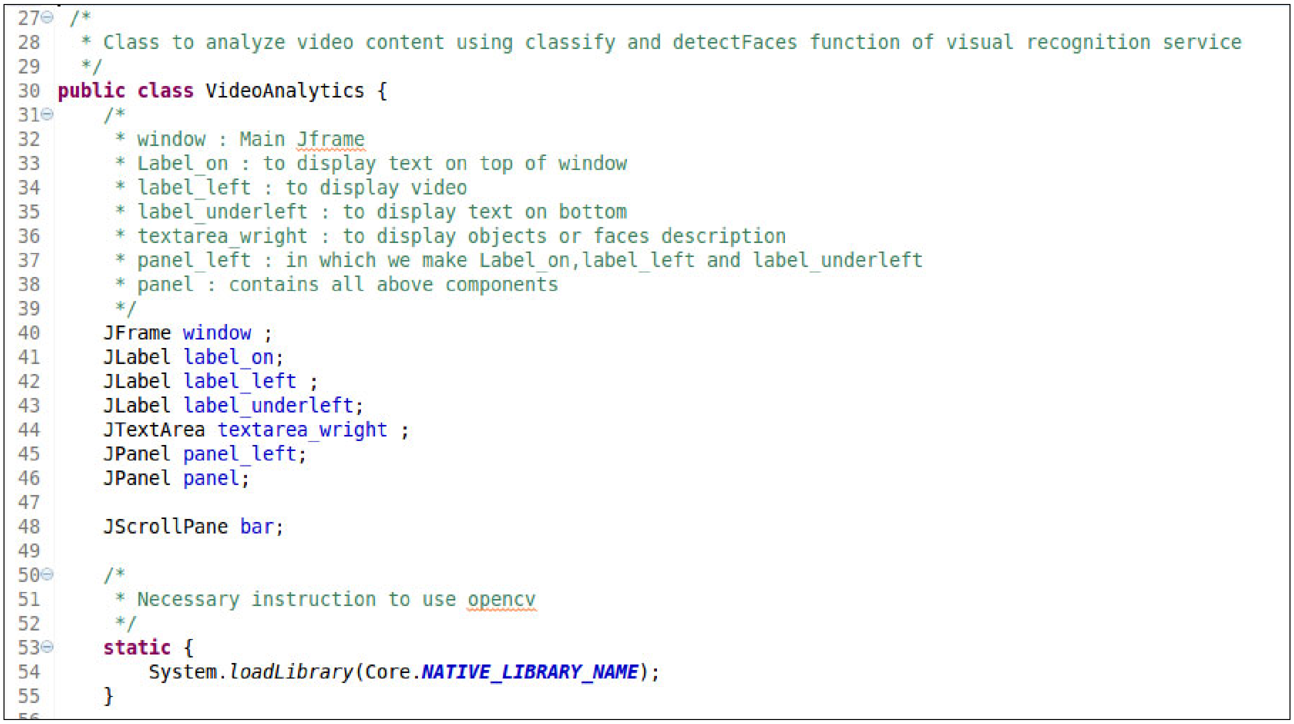
8. To enhance this application, a graphical interface (GUI) is created to display the video and the description of the content (Figure 4-50).



```

        //Create graphic interface by instantiate VideoAnalytics class
        VideoAnalytics VFrame=new VideoAnalytics();

```

Figure 4-50 instantiate VideoAnalytics class to create graphical interface

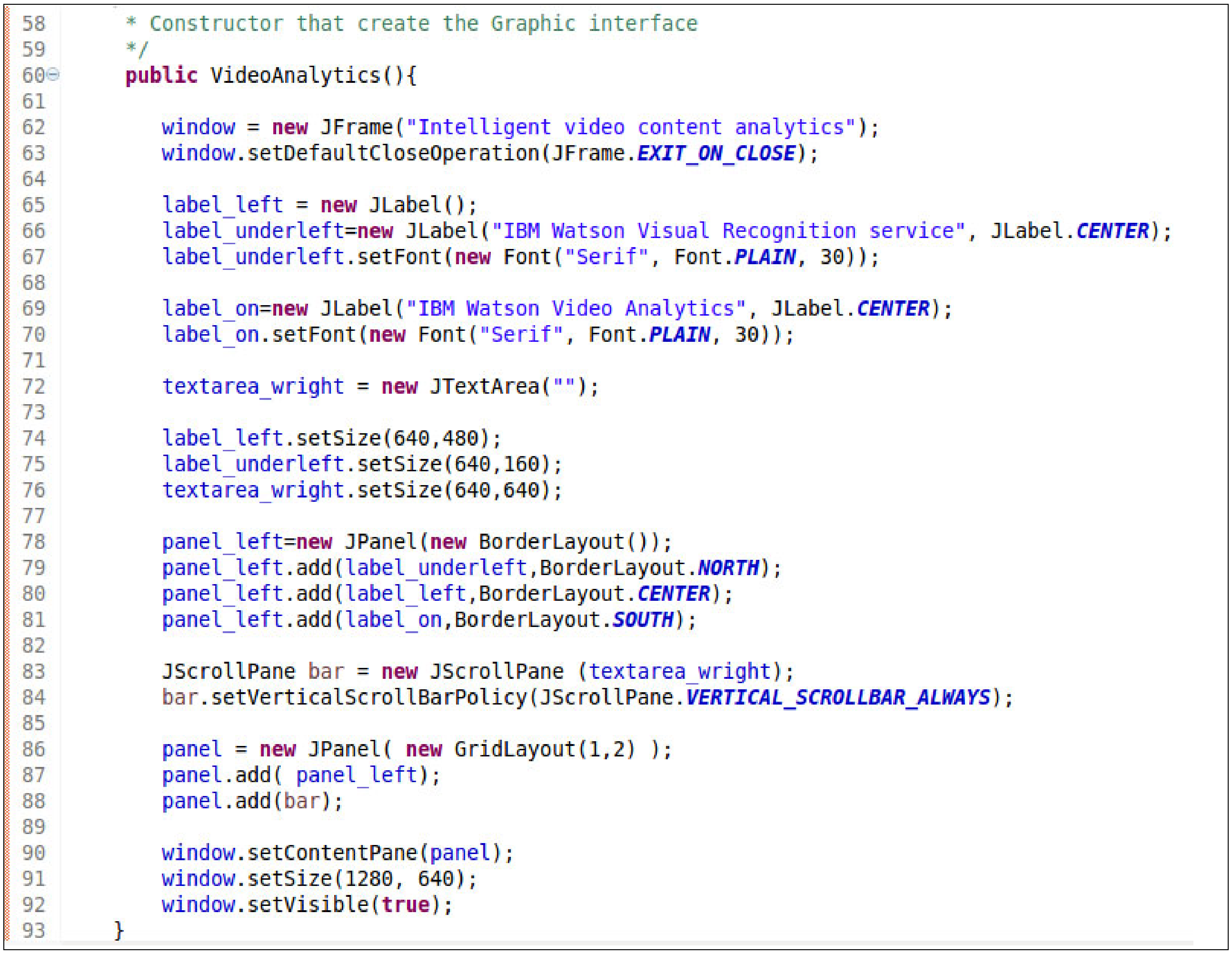
9. Declare all graphic components as class attributes (Figure 4-51).



```
/*
 * Class to analyze video content using classify and detectFaces function of visual recognition service
 */
public class VideoAnalytics {
    /*
     * window : Main Jframe
     * Label_on : to display text on top of window
     * label_left : to display video
     * label_underleft : to display text on bottom
     * textarea_wright : to display objects or faces description
     * panel_left : in which we make Label_on,label_left and label_underleft
     * panel : contains all above components
     */
    JFrame window ;
    JLabel label_on;
    JLabel label_left ;
    JLabel label_underleft;
    JTextArea textarea_wright ;
    JPanel panel_left;
    JPanel panel;

    JScrollPane bar;

    /*
     * Necessary instruction to use opencv
     */
    static {
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
    }
```

Figure 4-51 Graphic components declaration

Figure 4-52 shows the code in the class constructor that builds the graphical interface. The graphical interface is used to display video and its content description.



```
     * Constructor that create the Graphic interface
     */
    public VideoAnalytics(){

        window = new JFrame("Intelligent video content analytics");
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        label_left = new JLabel();
        label_underleft=new JLabel("IBM Watson Visual Recognition service", JLabel.CENTER);
        label_underleft.setFont(new Font("Serif", Font.PLAIN, 30));

        label_on=new JLabel("IBM Watson Video Analytics", JLabel.CENTER);
        label_on.setFont(new Font("Serif", Font.PLAIN, 30));

        textarea_wright = new JTextArea("");

        label_left.setSize(640,480);
        label_underleft.setSize(640,160);
        textarea_wright.setSize(640,640);

        panel_left=new JPanel(new BorderLayout());
        panel_left.add(label_underleft,BorderLayout.NORTH);
        panel_left.add(label_left,BorderLayout.CENTER);
        panel_left.add(label_on,BorderLayout.SOUTH);

        JScrollPane bar = new JScrollPane (textarea_wright);
        bar.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

        panel = new JPanel( new GridLayout(1,2) );
        panel.add( panel_left);
        panel.add(bar);

        window.setContentPane(panel);
        window.setSize(1280, 640);
        window.setVisible(true);
   }
```

Figure 4-52 Creation of graphic interface

10. Save the project (**File** → **Save**) and run the application as described in the next section.

4.3.6 Running the application

To run the Intelligent Video Content Analytics application, complete these steps:

1. Copy the path of your video or use the paths described in this project. You can get the video (IBM Intelligent Video Analytics Overview) at either of the following locations:
   - https://www.ibm.com/us-en/marketplace/video-analytics-for-security
   - https://youtu.be/fUKpGLk9Ml8
2. Run the project: Right-click the project and select **Run As** → **Run Configurations** (Figure 4-53).

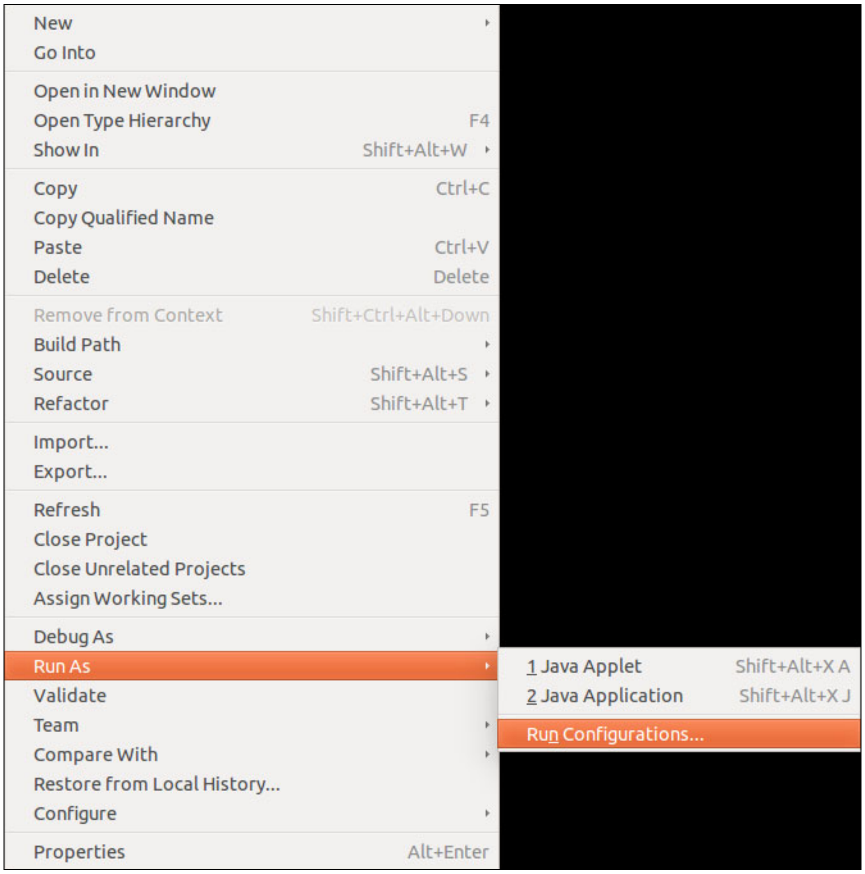
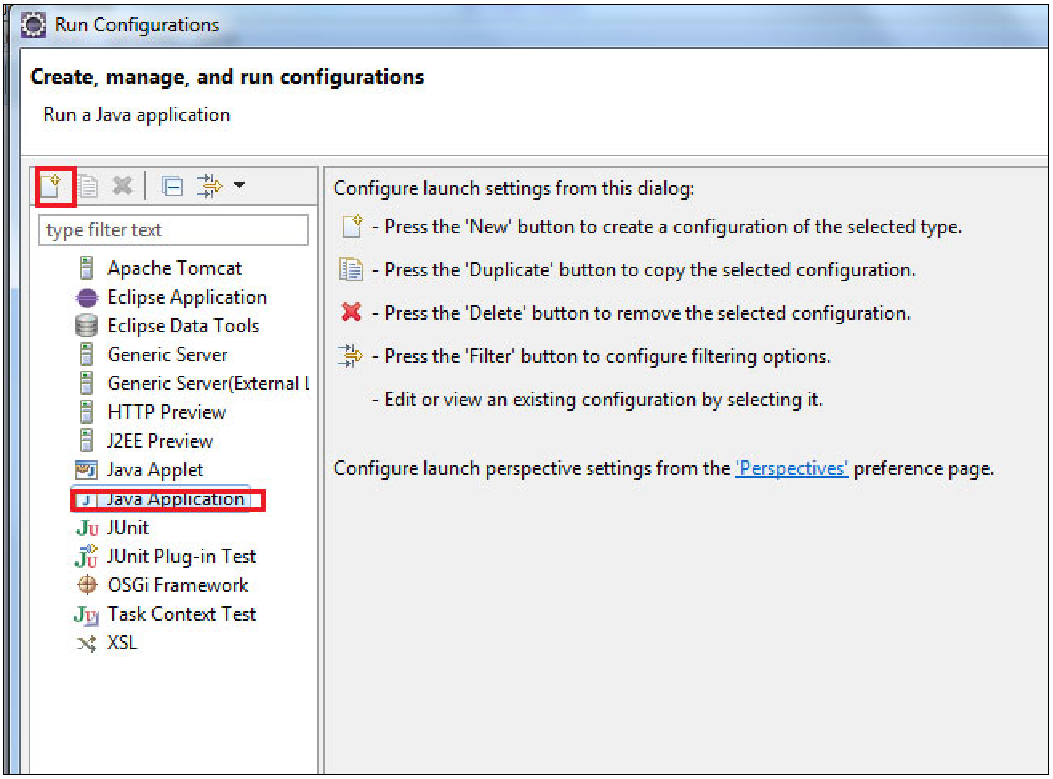


Figure 4-53 Run Configurations

3. Select **Java Application** and click the **New** button (Figure 4-54) to create a configuration.



Run Configurations

Create, manage, and run configurations

Run a Java application

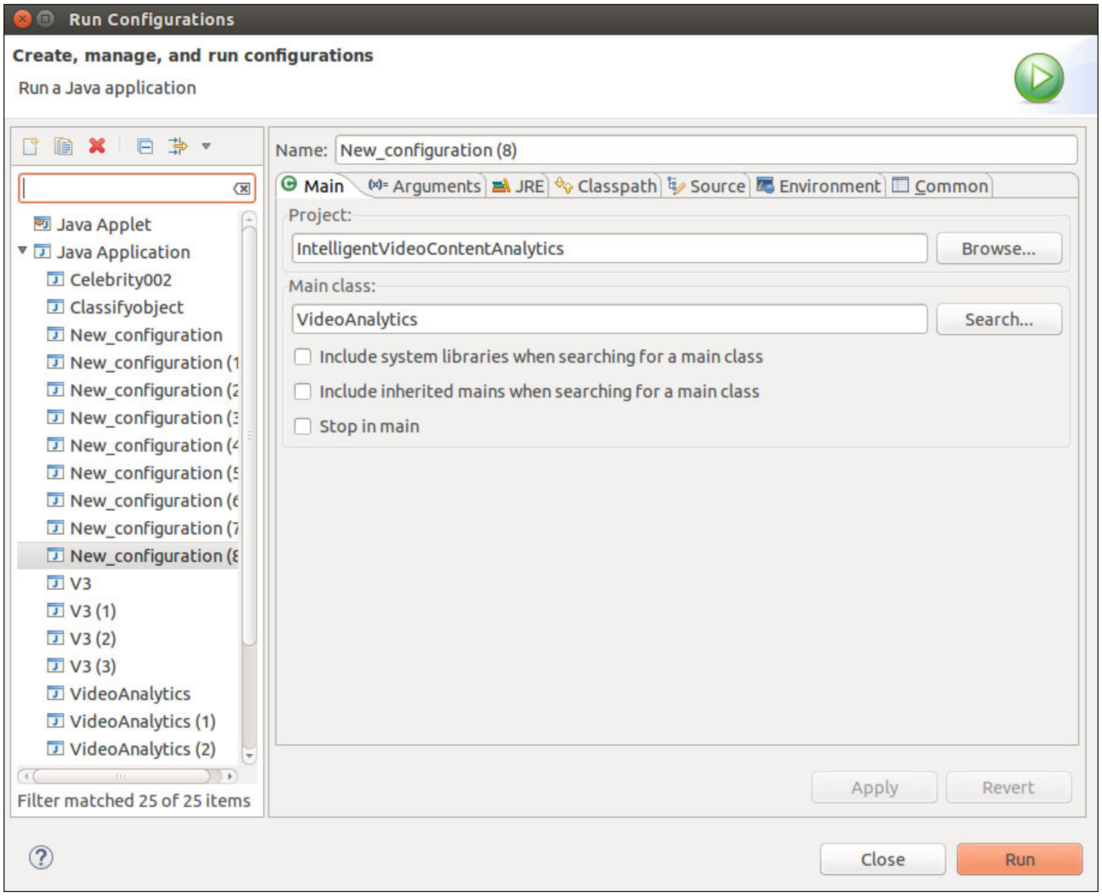
Configure launch settings from this dialog:

- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

Figure 4-54 The New button

4. On the Main page (Figure 4-55), click **Browse** to find and select the project (`IntelligentVideoContentAnalytics`), click **Search** to find and select the main class, and then click **Run**.



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: New_configuration (8)

Project: IntelligentVideoContentAnalytics

Main class: VideoAnalytics

□ Include system libraries when searching for a main class

□ Include inherited mains when searching for a main class

□ Stop in main

Filter matched 25 of 25 items

Figure 4-55 Select the project and main class

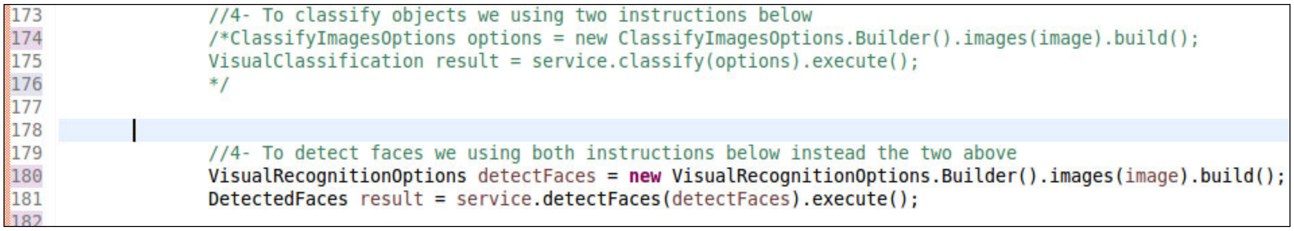
The program runs and displays the results shown in 4.1.3, “Expected results” on page 84.

4.4 Changing your application to detect faces

You can change the application to detect faces instead of performing object classification. Complete these steps:

1. To detect faces, use the `detectFaces()` method instead of the `classify()` method of `VisualRecognition` class.

   > **Update the code:** Comment out the first two lines of code (lines 174 and 175) and remove the comments for the next two (lines 180 and 181). Figure 4-56 shows what your code should look like after you update the code.



```
173                 //4- To classify objects we using two instructions below
174                 /*ClassifyImagesOptions options = new ClassifyImagesOptions.Builder().images(image).build();
175                 VisualClassification result = service.classify(options).execute();
176                 */
177
178
179                 //4- To detect faces we using both instructions below instead the two above
180                 VisualRecognitionOptions detectFaces = new VisualRecognitionOptions.Builder().images(image).build();
181                 DetectedFaces result = service.detectFaces(detectFaces).execute();
182
```

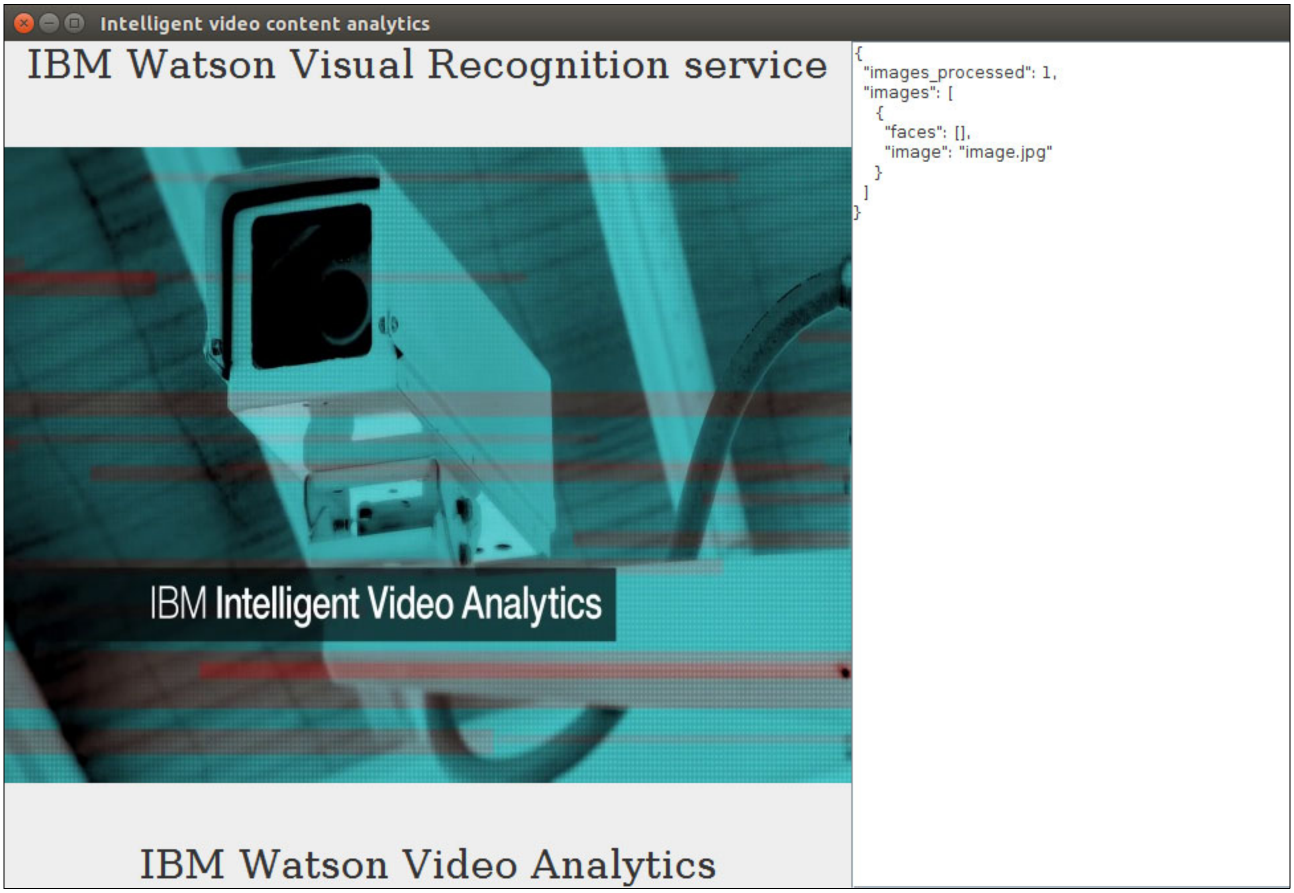
Figure 4-56 Change to detectFaces instead classify

2. Understand the code. Figure 4-56 shows the code that detects faces in the video frame:
   - The first line of code shows how to create a `VisualRecognitionOptions` object based on the current video frame image.
     - To create the new options for the new image, instantiate a new builder (`VisualRecognitionOptions.Builder()`), call the `images()` method to set the new image to analyze. This function accepts an image file as the parameter and returns the builder.
     - At the end, call the `build()` method with any argument that builds the profile options and returns the profile options (`VisualRecognitionOptions`).
   - The second line of code shows how to call the Watson Visual Recognition service which performs the actual detection of faces within the current video frame image. The result of the face detection is saved in the `result` variable.
     - To detect faces, call the `detectFaces()` method:

       `service.detectFaces(VisualRecognitionOptions)`

     - It accepts options (`VisualRecognitionOptions`) as argument and returns a `DetectedFaces` JSON object. The `detectFaces()` method of the Visual Recognition service analyzes images and detects faces.
     - The `execute()` function is used to run the service.

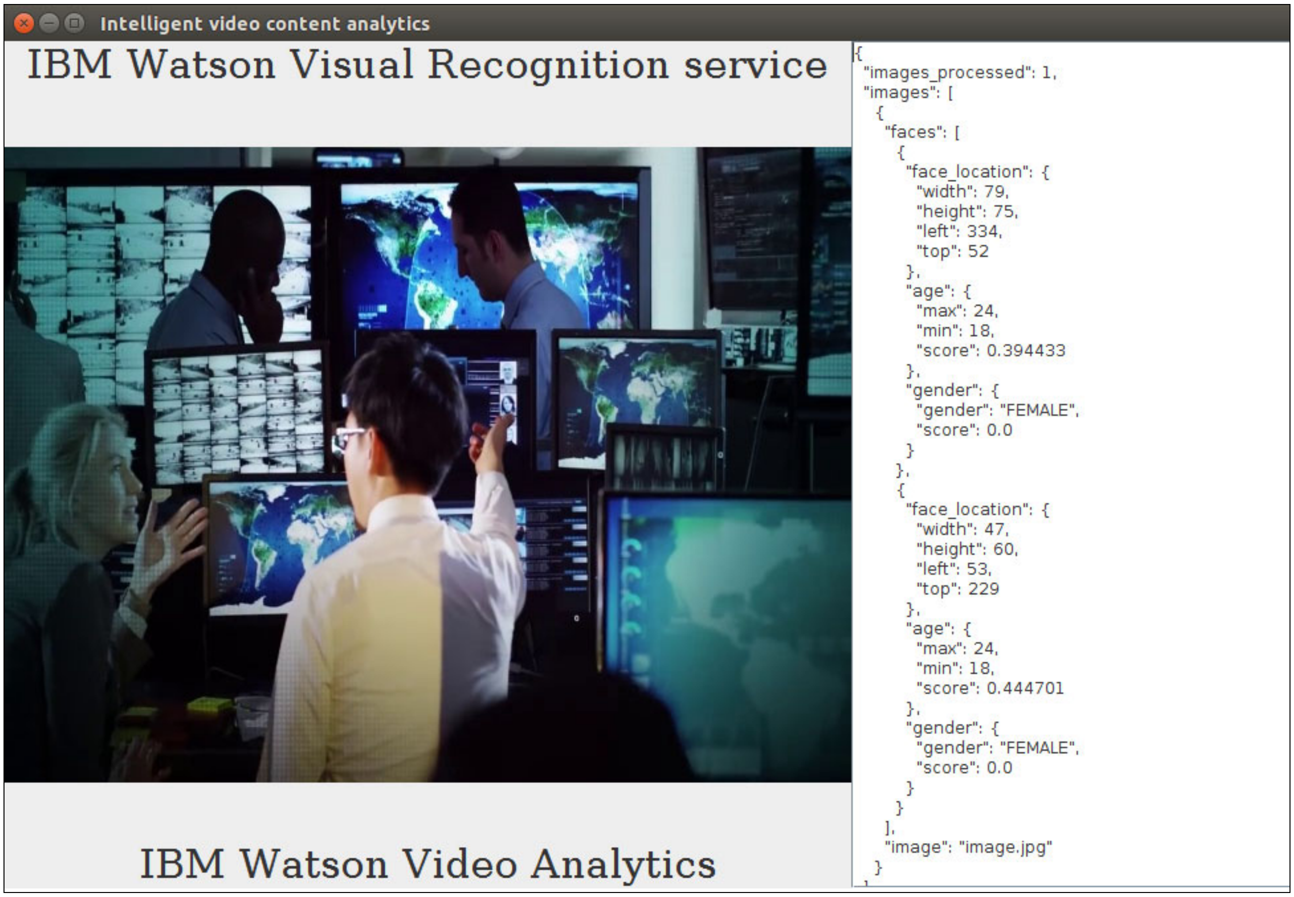
3. After you change your code to detect faces instead of classifying objects, save the change and rerun the program. The results for the same input video but if no faces are detected in the video frame are shown in Figure 4-57.


Intelligent video content analytics
IBM Watson Visual Recognition service
IBM Intelligent Video Analytics
IBM Watson Video Analytics

```
{
 "images_processed": 1,
 "images": [
  {
   "faces": [],
   "image": "image.jpg"
  }
 ]
}
```

Figure 4-57 Result if no person is in the scene

If a person appears in the video, the results differ, as shown in Figure 4-58.


Intelligent video content analytics
IBM Watson Visual Recognition service
IBM Watson Video Analytics

```
{
 "images_processed": 1,
 "images": [
  {
   "faces": [
    {
     "face_location": {
      "width": 79,
      "height": 75,
      "left": 334,
      "top": 52
     },
     "age": {
      "max": 24,
      "min": 18,
      "score": 0.394433
     },
     "gender": {
      "gender": "FEMALE",
      "score": 0.0
     }
    },
    {
     "face_location": {
      "width": 47,
      "height": 60,
      "left": 53,
      "top": 229
     },
     "age": {
      "max": 24,
      "min": 18,
      "score": 0.444701
     },
     "gender": {
      "gender": "FEMALE",
      "score": 0.0
     }
    }
   ],
   "image": "image.jpg"
  }
```

Figure 4-58 Result if a person appears in the scene

Using video from the camera: You can extend this program to use video from the camera:

1. Find this instruction:

   ```
   VideoCapture camera = new VideoCapture("path of video file ")
   ```

2. Change that instruction as follows:

   ```
   VideoCapture camera = new VideoCapture()
   ```

This program can be extended to other use cases.

4.5 References

See the following resources:

- OpenCV 3.0.0-dev documentation (Using OpenCV Java with Eclipse):

  http://docs.opencv.org/3.0-beta/doc/tutorials/introduction/java_eclipse/java_eclipse.html

- Move your Java application into a hybrid cloud using Bluemix, Part 3 (IBM developerWorks):

  http://www.ibm.com/developerworks/cloud/library/cl-move-java-app-hybrid-cloud3-bluemix-trs/

- Watson Developer Cloud:

  https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/

A

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The following Git repositories are available to help you with the examples in this book:

- For Chapter 3, “Image Content Description” on page 35:

  `https://github.com/snippet-java/redbooks-vis-301-ImageContentDescription`

- For Chapter 4, “Intelligent Video Content Analytics” on page 83:

  `https://github.com/snippet-java/redbooks-vis-301-IntelligentVideoContentAnalytics`

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

The volumes in the *Building Cognitive Applications with IBM Watson APIs* series:

- *Volume 1 Getting Started*, SG24-8387
- *Volume 2 Conversation*, SG24-8394
- *Volume 3 Visual Recognition*, SG24-8393
- *Volume 4 Natural Language Classifier*, SG24-8391
- *Volume 5 Language Translator*, SG24-8392
- *Volume 6 Speech to Text and Text to Speech*, SG24-8388
- *Volume 7 Natural Language Understanding*, SG24-8398

You can search for, view, download or order these documents and other Redbooks, Redpapers™, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- Classify an image topic in Watson Developer Cloud:

  https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/#classify_an_image

- Detect faces topic in Watson Developer Cloud:

  https://www.ibm.com/watson/developercloud/visual-recognition/api/v3/#detect_faces

- Create or log in to IBM Bluemix account:

  https://console.ng.bluemix.net/

- Visual Recognition getting started tutorials:

  https://www.ibm.com/watson/developercloud/doc/visual-recognition/getting-started.html

- Download Eclipse:

  https://eclipse.org/downloads/

- Getting Started with Eclipse:

  https://eclipse.org/users/

- Getting Started with Java Programming:

  http://www.oracle.com/technetwork/topics/newtojava/learn-141096.html
- *Watson Developer Cloud Node.js SDK:*

  https://www.npmjs.com/package/watson-developer-cloud
- Node.js usage examples of the Watson APIs:

  https://github.com/watson-developer-cloud/node-sdk
- Eclipse IDE Luna:

  http://www.eclipse.org/luna
- Move your Java application into a hybrid cloud using Bluemix:

  http://www.ibm.com/developerworks/cloud/library/cl-move-java-app-hybrid-cloud3-bluemix-trs/
- Download and install the Cloud Foundry command-line interface (CLI).

  https://console.ng.bluemix.net/docs/starters/install_cli.html
- OpenCV 3.x.x for Java:

  http://opencv-java-tutorials.readthedocs.io/en/latest/01-installing-opencv-for-java.html
- Using OpenCV Java with Eclipse

  http://docs.opencv.org/3.0-beta/doc/tutorials/introduction/java_eclipse/java_eclipse.html

Also see the list of online resources for the following chapters in this book:

- Basics of Watson Visual Recognition API:1.6, “References” on page 27
- Classify images with a custom classifier: 2.3, “References” on page 34
- Image Content Description: 3.5, “References” on page 81
- Intelligent Video Content Analytics: 4.5, “References” on page 124

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

Building Cognitive Applications with IBM Watson Services: Volume 3 Visual Recognition

(0.2”spine)
0.17”<->0.473”
90<->249 pages

SG24-8393-00

ISBN 0738442577

Printed in U.S.A.