Faculty of Science

Course:	CSCI 2020u: Software Systems Development and Integration
Component:	Practice Final Exam - Lab Exam Portion

Overview

In this lab exam, you will download JSON data from the Internet, and display the results in a bar chart. Your application will also save that data to a CSV file.

Instructions

You can use any operating system for this lab exam. You can use IntelliJ for this lab exam, but you can also use another IDE (or just a plain text editor and the command line). Use what you know best. This is an assessment, and the instructor will be unable to solve your technical issues for you. It is your responsibility to know an operating system and environment well enough to complete this exam.

Overview

Your program will perform the following basic functions:

1. Programmatically, download data about housing prices from the following URL:
   - `http://csundergrad.science.uoit.ca/csci2020u/data/housing_prices.json`
   - The results come back in JSON format (see listing 1 for a sample)
2. Process the JSON data using the SimpleJSON library
   - Hint: Refer back to the 'More Socket Examples' demo
   - See listing 3 for a build.gradle file that adds the SimpleJSON dependency
3. Display the data in a JavaFX bar chart (by drawing rectangles using 2D graphics)
   - The bar chart must include a legend, as shown in the output in figure 1
   - Hint: refer back to Lab 06 for reference
4. Save data to a comma-separate values (CSV) file
   - Use filename: `data.csv`
   - See listing 2 for a sample of the output

Listing 1 – Sample JSON data

```
[
 {
   "Year": 2002,
   "1 bed flats": 166,
   "2 bed flats": 213,
   "2 bed houses": 265,
   "3 bed houses": 332,
   "4 bed houses": 416
 },
 {
   "Year": 2003,
   "1 bed flats": 156,
```

```
    "2 bed flats": 216,
    "2 bed houses": 285,
    "3 bed houses": 328,
    "4 bed houses": 380
  },
  ...
  {
    "Year": 2015,
    "1 bed flats": 201,
    "2 bed flats": 335,
    "2 bed houses": 438,
    "3 bed houses": 524,
    "4 bed houses": 707
  }
]
```

Listing 2 – Sample Comma-separated text (CSV) output file (data.csv)

```
Year,1 bed flats,1 bed flats,2 bed houses,3 bed houses,4 bed houses
2002,166,213,265,332,416
2003,156,216,285,328,380
2004,166,219,257,351,448
2005,121,224,264,364,458
2006,163,259,303,375,486
2007,170,235,326,402,496
2008,255,327,391,510,638
2009,237,321,408,528,758
2010,225,291,386,517,651
2011,197,328,417,510,698
2012,196,331,385,485,647
2013,224,340,407,477,702
2014,199,340,377,458,691
2015,201,335,438,524,707
```

Listing 3 – The gradle file (build.gradle)

```
apply plugin: 'java'

repositories {
   mavenCentral()
}

dependencies {
   compile group: 'org.json',
           name: 'json',
           version: '20160810'
}

task(run, dependsOn: 'classes', type: JavaExec) {
   main = 'dataviz.Main'
   classpath = sourceSets.main.runtimeClasspath
}
```

Detailed Requirements

The program will be broken down into 3 classes:

- `Main` – this is the UI for the application
- `HousingPricesLoader` – this class provides the method for downloading and processing the JSON data from the web
- `FileStorage` – this class provides the method for saving the data to .csv format

Note: *All of these classes must be placed into a package called '`dataviz`'.*

The Source JSON Data

The JSON data to be used as input to this program will be downloaded from the web (by your program) at the following URL:
`http://csundergrad.science.uoit.ca/csci2020u/data/housing_prices.json`.
The data contains 5 series of data: 1 bed flats, 2 bed flats, 2 bed houses, 3 bed houses, and 4 bed houses. These will be the 5 series in your bar chart. Each series has 14 years of price data, 2002-2015.

Main
This class will be the user interface for the entire application. All user interface-related functionality is to be placed in this class. The user interface consists simply of a Canvas, to which we will draw our bar chart and legend. This class will also invoke the code to load the housing prices, using `HousingPricesLoader`.

The code to be included in this class:

- Create and initialize the user interface
- Invoke `HousingPricesLoader::loadPrices()` to obtain the data to be plotted
- Invoke `FileStorage::saveData()` to save the data to a CSV file
- Invoke `Main::drawBarChart()` to draw each series (1 bed flats, 2 bed flats, 2 bed houses, 3 bed houses, and 4 bed houses) as a single bar chart
  - The bar chart should include the years as labels along the bottom, as well as two black axis lines (see figure 1 for details)
  - The colour scheme for the bar chart is provided in Table 1
- Invoke `Main::drawLegend()` to draw a legend which maps the colours of the bars to the series names (1 bed flats, 2 bed flats, 2 bed houses, 3 bed houses, and 4 bed houses)

Series	Colour
`1 bed flats`	`Color.RED`
`2 bed flats`	`Color.BLUE`
`2 bed houses`	`Color.ORANGE`
`3 bed houses`	`Color.GREEN`
`4 bed houses`	`Color.YELLOW`

Table 1 – Colour scheme to use for the bar chart series

HousingPricesLoader
This class will implement one method, `loadPrices()`, which returns all 5 series of data loaded from the JSON data. The data will be obtained by programmatically downloading it from the URL given above, and processing it using SimpleJSON. The data will be returned as a `Map<String,List<Integer>>` object.

FileStorage
This class will implement one method, `saveData()`, which will save all 5 series of data into a CSV file, matching the format from listing 2 (including the header line). To generate .csv output, simply print the data in with commas (and newlines) in the appropriate places.

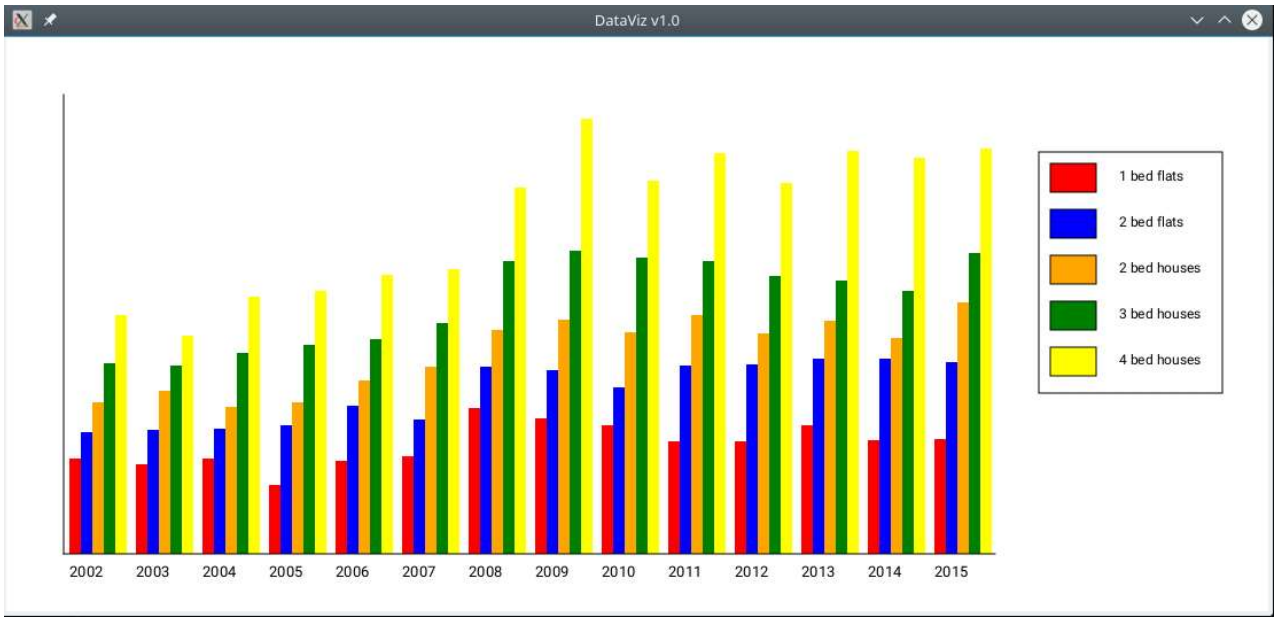

Figure 1: The application window