```
import java.awt.Desktop;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * This project implements a simple application. Properties from a fixed
 * file can be displayed.
 *
 *
 * @author Michael Kölling and Josh Murphy
 * @version 1.0
 */

public class PropertyViewer
{
   private PropertyViewerGUI gui;     // the Graphical User Interface
   private Portfolio portfolio;
   private Property toggleFavourite;
   private int counter= 1;
   private int PropertiesViewed=0;
   private int totalPropertyPrice=0;
   /**
    * Create a PropertyViewer and display its GUI on screen.
    */
   public PropertyViewer( )
   {
      gui = new PropertyViewerGUI(this);
      portfolio = new Portfolio("airbnb-london.csv");
      gui.showProperty(portfolio.getProperty(counter));
      gui.showID(portfolio.getProperty(counter));
      PropertiesViewed+=1;
      totalPropertyPrice+= portfolio.getProperty(counter).getPrice();
   }


   /**
    * This method is for the function of clicking the "next" button. It will show the property's
name and ID. After clicking "favourite"
    * button, the specific property will be added into the favourite list.
    * The total viewed times of each property will also be accumulated.
    */
   public void nextProperty()
   {
     counter=counter+1;
     gui.showProperty(portfolio.getProperty(counter));
     gui.showID(portfolio.getProperty(counter));
     gui.showFavourite(portfolio.getProperty(counter));
```

```
     PropertiesViewed+=1;
     totalPropertyPrice+= portfolio.getProperty(counter).getPrice();

   }

   /**
    * This method is for the function of clicking the "previous" button. It will also show the
property's name and ID on the screen. After
    * clicking "favourite" button, the specific property will be added into the favourite list.
The total viewed times of each property will be
    * accumulated.
    */
   public void previousProperty()
   {
     counter=counter-1;
     gui.showProperty(portfolio.getProperty(counter));
     gui.showID(portfolio.getProperty(counter));
     gui.showFavourite(portfolio.getProperty(counter));
     PropertiesViewed+=1;
     totalPropertyPrice+= portfolio.getProperty(counter).getPrice();

   }

   /**
    * This method is for adding the property into the favourite list by clicking toggleFavourite
button.
    */
   public void toggleFavourite()
   {
    (portfolio.getProperty(counter)).toggleFavourite();
    gui.showFavourite(portfolio.getProperty(counter));
   }


   //----- methods for challenge tasks -----

   /**
    * This method opens the system's default internet browser
    * The Google maps page should show the current properties location on the map.
    */
   public void viewMap() throws Exception
   {

     double latitude = portfolio.getProperty(counter).getLatitude();
     double longitude = portfolio.getProperty(counter).getLongitude();

     URI uri = new URI("https://www.google.com/maps/place/" + latitude + "," + longitude);
```

```
      java.awt.Desktop.getDesktop().browse(uri);
   }

   /**
    * This method shows the total number viewed of all properties.
    */
   public int getNumberOfPropertiesViewed()
   {
      return PropertiesViewed;
   }

   /**
    * This method calculates the average price by total property price devided by properties
viewed.
    */
   public int averagePropertyPrice()
   {
      int averagePrice=totalPropertyPrice/PropertiesViewed;
      return averagePrice;
   }
}
```

```
import java.awt.*;
import java.awt.image.*;
import javax.swing.*;
import javax.imageio.*;
import java.io.*;

/**
 * Property is a class that defines a property for display.
 *
 * @author  Michael Kölling and Josh Murphy
 * @version 2.0
 */
public class Property
{

   private String id;
   private String description;
   private String hostID;
   private String hostName;
   private String neighbourhood;
   private double latitude;
   private double longitude;
   private String roomType;
   private int price;
   private int minimumNights;
   private int availability365;
   private boolean isFavourite;

   /**
    * Create a new property with specified initial values.
    */
   public Property(String id, String name, String hostID, String hostName,
         String neighbourhood, double latitude, double longitude, String roomType,
         int price, int minimumNights, int availability365){
      this.id = id;
      this.description = name;
      this.hostID = hostID;
      this.hostName = hostName;
      this.neighbourhood = neighbourhood;
      this.latitude = latitude;
      this.longitude  = longitude;
      this.roomType = roomType;
      this.price = price;
      this.minimumNights = minimumNights;
      this.availability365 = availability365;

      isFavourite = false;
```

```
}

/**
 * Return the Id of this property.
 */
public String getID(){
   return id;
}

/**
 * Return the hostId of this property.
 */
public String getHostID(){
   return hostID;
}

/**
 * Return the latitude of this property.
 */
public double getLatitude(){
   return latitude;
}

/**
 * Return the longitude of this property.
 */
public double getLongitude(){
   return longitude;
}

/**
 * Return the price of this property.
 */
public int getPrice(){
   return price;
}

/**
 * Returns true if this property is currently marked as a favourite, false otherwise.
 */
public boolean isFavourite(){
   return isFavourite;
}

/**
 * Return the host name of this property.
 */
```

```
    public String getHostName(){
        return hostName;
    }

    /**
     * Return the neighbourhood of this property.
     */
    public String getNeighbourhood(){
        return neighbourhood;
    }

    /**
     * Return the room type of this property.
     */
    public String getRoomType(){
        return roomType;
    }

    /**
     * Return the minimum number of nights this property can be booked for.
     */
    public String getMinNights(){
        return "" + minimumNights;
    }

    /**
     * Return the description of this property.
     */
    public String getDescription(){
        return description;
    }

    /**
     * Toggles whether this property is marked as a favourite or not.
     */
    public void toggleFavourite()
    {
        isFavourite = !isFavourite;
    }

}
```

```
import java.util.List;
import java.util.ArrayList;
import java.io.File;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Arrays;
import com.opencsv.CSVReader;
import java.net.URISyntaxException;

/**
 * A portfolio is a collection of properties. It reads properties from a file on disk,
 * and it can be used to retrieve single properties.
 *
 * The file name to read from is passed in at construction.
 *
 * @author Michael Kölling and Josh Murphy
 * @version 1.0
 */
public class Portfolio
{
   private List<Property> properties;

   /**
    * Constructor for objects of class Portfolio
    */
   public Portfolio(String directoryName)
   {
      properties = loadProperties();
   }

   /**
    * Return a property from this Portfolio.
    */
   public Property getProperty(int propertyNumber)
   {
      return properties.get(propertyNumber);
   }

   /**
    * Return the number of Properties in this Portfolio.
    */
   public int numberOfProperties()
   {
```

```
        return properties.size();
    }

    /**
     * Return an ArrayList containing the rows in the AirBnB London data set csv file.
     */
    public List<Property> loadProperties() {
        System.out.print("Begin loading Airbnb london dataset...");
        ArrayList<Property> listings = new ArrayList<Property>();
        try{
            URL url = getClass().getResource("airbnb-london.csv");
            CSVReader reader = new CSVReader(new FileReader(new
File(url.toURI()).getAbsolutePath()));
            String [] line;
            //skip the first row (column headers)
            reader.readNext();
            while ((line = reader.readNext()) != null) {
                String id = line[0];
                String name = line[1];
                String host_id = line[2];
                String host_name = line[3];
                String neighbourhood = line[4];
                double latitude = convertDouble(line[5]);
                double longitude = convertDouble(line[6]);
                String room_type = line[7];
                int price = convertInt(line[8]);
                int minimumNights = convertInt(line[9]);
                int availability365 = convertInt(line[10]);

                Property currentProperty = new Property(id, name, host_id, host_name,
                    neighbourhood, latitude,longitude, room_type, price,
                    minimumNights, availability365);
                listings.add(currentProperty);
            }
        } catch(IOException | URISyntaxException e){
            System.out.println("Failure! Something went wrong when loading the property file");
            e.printStackTrace();
        }
        System.out.println("Success! Number of loaded records: " + listings.size());
        return listings;
    }

    /**
     *
     * @param doubleString the string to be converted to Double type
     * @return the Double value of the string, or -1.0 if the string is
     * either empty or just whitespace
```

```
     */
    private Double convertDouble(String doubleString){
        if(doubleString != null && !doubleString.trim().equals("")){
            return Double.parseDouble(doubleString);
        }
        return -1.0;
    }

    /**
     *
     * @param intString the string to be converted to Integer type
     * @return the Integer value of the string, or -1 if the string is
     * either empty or just whitespace
     */
    private Integer convertInt(String intString){
        if(intString != null && !intString.trim().equals("")){
            return Integer.parseInt(intString);
        }
        return -1;
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

import java.io.File;

import java.util.List;
import java.util.ArrayList;
import java.util.Iterator;

/**
 * PropertyViewerGUI provides the GUI for the project. It displays the property
 * and strings, and it listens to button clicks.

 * @author Michael Kölling, David J Barnes, and Josh Murphy
 * @version 3.0
 */
public class PropertyViewerGUI
{
   // fields:
   private JFrame frame;
   private JPanel propertyPanel;
   private JLabel idLabel;
   private JLabel favouriteLabel;

   private JTextField hostIDLabel;
   private JTextField hostNameLabel;
   private JTextField neighbourhoodLabel;
   private JTextField roomTypeLabel;
   private JTextField priceLabel;
   private JTextField minNightsLabel;
   private JTextArea descriptionLabel;

   private Property currentProperty;
   private PropertyViewer viewer;
   private boolean fixedSize;

   /**
    * Create a PropertyViewer and display its GUI on screen.
    */
   public PropertyViewerGUI(PropertyViewer viewer)
   {
      currentProperty = null;
      this.viewer = viewer;
      fixedSize = false;
      makeFrame();
```

```
    this.setPropertyViewSize(400, 250);
}


// ---- public view functions ----

/**
 * Display a given property
 */
public void showProperty(Property property)
{
    hostIDLabel.setText(property.getHostID());
    hostNameLabel.setText(property.getHostName());
    neighbourhoodLabel.setText(property.getNeighbourhood());
    roomTypeLabel.setText(property.getRoomType());
    priceLabel.setText("£" + property.getPrice());
    minNightsLabel.setText(property.getMinNights());
    //descriptionLabel.setText(property.getDescription());
}

/**
 * Set a fixed size for the property display. If set, this size will be used for all properties.
 * If not set, the GUI will resize for each property.
 */
public void setPropertyViewSize(int width, int height)
{
    propertyPanel.setPreferredSize(new Dimension(width, height));
    frame.pack();
    fixedSize = true;
}

/**
 * Show a message in the status bar at the bottom of the screen.
 */
public void showFavourite(Property property)
{
    String favouriteText = " ";
    if (property.isFavourite()){
        favouriteText += "This is one of your favourite properties!";
    }
    favouriteLabel.setText(favouriteText);
}

/**
 * Show the ID in the top of the screen.
 */
public void showID(Property property){
```

```
    idLabel.setText("Current Property ID:" + property.getID());
}

// ---- implementation of button functions ----

/**
 * Called when the 'Next' button was clicked.
 */
private void nextButton()
{
    viewer.nextProperty();
}

/**
 * Called when the 'Previous' button was clicked.
 */
private void previousButton()
{
    viewer.previousProperty();
}

/**
 * Called when the 'View on Map' button was clicked.
 */
private void viewOnMapsButton()
{
    try{
     viewer.viewMap();
    }
    catch(Exception e){
        System.out.println("URL INVALID");
    }

}

/**
 * Called when the 'Toggle Favourite' button was clicked.
 */
private void toggleFavouriteButton(){
    viewer.toggleFavourite();
}

// ---- swing stuff to build the frame and all its components ----

/**
 * Create the Swing frame and its content.
 */
```

```
private void makeFrame()
{
   frame = new JFrame("Portfolio Viewer Application");
   JPanel contentPane = (JPanel)frame.getContentPane();
   contentPane.setBorder(new EmptyBorder(6, 6, 6, 6));

   // Specify the layout manager with nice spacing
   contentPane.setLayout(new BorderLayout(6, 6));

   // Create the property pane in the center
   propertyPanel = new JPanel();
   propertyPanel.setLayout(new GridLayout(6,2));

   propertyPanel.add(new JLabel("HostID: "));
   hostIDLabel = new JTextField("default");
   hostIDLabel.setEditable(false);
   propertyPanel.add(hostIDLabel);

   propertyPanel.add(new JLabel("Host Name: "));
   hostNameLabel = new JTextField("default");
   hostNameLabel.setEditable(false);
   propertyPanel.add(hostNameLabel);

   propertyPanel.add(new JLabel("Neighbourhood: "));
   neighbourhoodLabel = new JTextField("default");
   neighbourhoodLabel.setEditable(false);
   propertyPanel.add(neighbourhoodLabel);

   propertyPanel.add(new JLabel("Room type: "));
   roomTypeLabel = new JTextField("default");
   roomTypeLabel.setEditable(false);
   propertyPanel.add(roomTypeLabel);

   propertyPanel.add(new JLabel("Price: "));
   priceLabel = new JTextField("default");
   priceLabel.setEditable(false);
   propertyPanel.add(priceLabel);

   propertyPanel.add(new JLabel("Minimum nights: "));
   minNightsLabel = new JTextField("default");
   minNightsLabel.setEditable(false);
   propertyPanel.add(minNightsLabel);

   propertyPanel.setBorder(new EtchedBorder());
   contentPane.add(propertyPanel, BorderLayout.CENTER);

   // Create two labels at top and bottom for the file name and status message
```

```
idLabel = new JLabel("default");
contentPane.add(idLabel, BorderLayout.NORTH);

favouriteLabel = new JLabel(" ");
contentPane.add(favouriteLabel, BorderLayout.SOUTH);

// Create the toolbar with the buttons
JPanel toolbar = new JPanel();
toolbar.setLayout(new GridLayout(0, 1));

JButton nextButton = new JButton("Next");
nextButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) { nextButton(); }
            });
toolbar.add(nextButton);

JButton previousButton = new JButton("Previous");
previousButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) { previousButton(); }
            });
toolbar.add(previousButton);

JButton mapButton = new JButton("View Property on Map");
mapButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) { viewOnMapsButton(); }
            });
toolbar.add(mapButton);

JButton favouriteButton = new JButton("Toggle Favourite");
favouriteButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) { toggleFavouriteButton(); }
            });
toolbar.add(favouriteButton);


// Add toolbar into panel with flow layout for spacing
JPanel flow = new JPanel();
flow.add(toolbar);

contentPane.add(flow, BorderLayout.WEST);

// building is done - arrange the components
frame.pack();

// place the frame at the center of the screen and show
Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
frame.setLocation(d.width/2 - frame.getWidth()/2, d.height/2 - frame.getHeight()/2);
```

```
        frame.setVisible(true);
    }
}
```