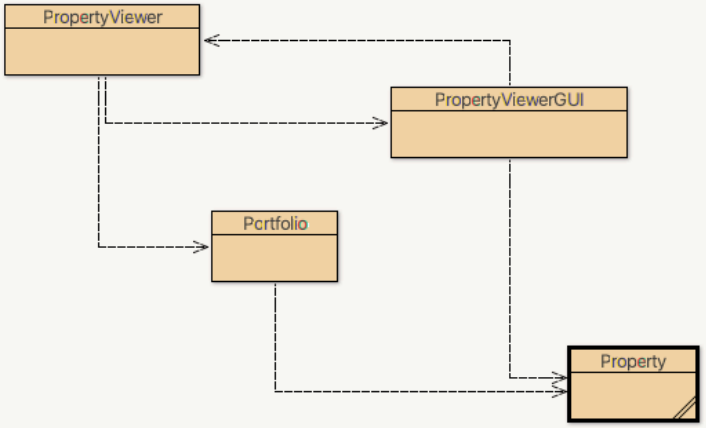
PropertyViewer
PropertyViewerGUI
Portfolio
Property

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

import java.awt.*;
import javax.swing.*; //imported from PropertyViewerGUI to create new JFrame

/**
 * This project implements a simple application. Properties from a fixed
 * file can be displayed.
 *
 *
 * @author Michael Kölling and Josh Murphy
 * @version 1.0
 *
 * Modified by Emre Salur K20082471
 * Did all base tasks and challenge tasks
 *
 */
public class PropertyViewer
{
    private PropertyViewerGUI gui;    //  the Graphical User Interface
    private Portfolio portfolio;      //  contatins an arrayList of properties

    private int index = 0;            //  index number for each property inside the portfolio
    private int propertyViewed = 1;   //  number of properties viewed in the application
    private int totalPrice = 0;       //  calculates the total price of each viewed property

    /**
     * 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(index));
        gui.showID(portfolio.getProperty(index));
        totalPrice += portfolio.getProperty(index).getPrice();
    }

    /**
     * Displays the next property and updates information accordingly
     */
    public void nextProperty()
    {
        if(index < (portfolio.numberOfProperties() - 1)){
```

```
48              index += 1;
49              gui.showProperty(portfolio.getProperty(index));
50              gui.showID(portfolio.getProperty(index));
51              gui.showFavourite(portfolio.getProperty(index));
52              propertyViewed += 1;
53              totalPrice += portfolio.getProperty(index).getPrice();
54          }
55      }
56
57      /**
58       * Displays the previous property and updates information accordingly
59       */
60      public void previousProperty()
61      {
62          if(index > 0){
63              index -= 1;
64              gui.showProperty(portfolio.getProperty(index));
65              gui.showID(portfolio.getProperty(index));
66              gui.showFavourite(portfolio.getProperty(index));
67              propertyViewed += 1;
68              totalPrice += portfolio.getProperty(index).getPrice();
69          }
70      }
71
72      /**
73       * Updates and displays whether a property is favourite or not
74       */
75      public void toggleFavourite()
76      {
77          portfolio.getProperty(index).toggleFavourite();
78          gui.showFavourite(portfolio.getProperty(index));
79      }
80
81      //----- methods for challenge tasks -----
82
83      /**
84       * This method opens the system's default internet browser
85       * The Google maps page should show the current properties location on the map.
86       */
87      public void viewMap() throws Exception
88      {
89          double latitude = portfolio.getProperty(index).getLatitude();
90          double longitude = portfolio.getProperty(index).getLongitude();
91
92          URI uri = new URI("https://www.google.com/maps/place/" + latitude + "," +
longitude);
93          java.awt.Desktop.getDesktop().browse(uri);
94      }
95
96      /**
```

```
97      * Returns the number of properties viewed since the start of the application
98      */
99     public int getNumberOfPropertiesViewed()
100    {
101        return propertyViewed;
102    }
103
104    /**
105     * Returns the average property price of viewed properties since the start of
    the application
106     */
107    public int averagePropertyPrice()
108    {
109        int averagePrice = totalPrice / propertyViewed;
110        return averagePrice;
111    }
112    /**
113     * Displays the statistics information from the getNumberOfPropertiesViewed and
    averagePropertyPrice methods
114     */
115    public void statistics()
116    {
117        JFrame statisticsFrame = new
    JFrame("Statistics");                                                    //
    creates new window
118        statisticsFrame.setPreferredSize(new Dimension(200,
    100));                                                  //  edits the size of
    window
119        statisticsFrame.setLayout(new
    FlowLayout());                                                          //
    edits flow of labels so they do not overlap
120
121        JLabel propertyViewedLabel = new JLabel("Number of Properties Viewed: " +
    getNumberOfPropertiesViewed());    //  initializing labels to add
122        JLabel averagePriceLabel = new JLabel("Average Property Price: £" +
    averagePropertyPrice());
123
124
    statisticsFrame.add(propertyViewedLabel);
                           //  adding labels to the window
125
    statisticsFrame.add(averagePriceLabel);
126
127
    statisticsFrame.setVisible(true);
                           //  the window becomes visible
128
    statisticsFrame.pack();
                           //  shows the complete new window
```

```
129         }
130
131 }
132
```

```java
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();
    }
```

```
50
51      /**
52       * Return an ArrayList containing the rows in the AirBnB London data set csv
         file.
53       */
54      public List<Property> loadProperties() {
55          System.out.print("Begin loading Airbnb london dataset...");
56          ArrayList<Property> listings = new ArrayList<Property>();
57          try{
58              URL url = getClass().getResource("airbnb-london.csv");
59              CSVReader reader = new CSVReader(new FileReader(new
                File(url.toURI()).getAbsolutePath()));
60              String [] line;
61              //skip the first row (column headers)
62              reader.readNext();
63              while ((line = reader.readNext()) != null) {
64                  String id = line[0];
65                  String name = line[1];
66                  String host_id = line[2];
67                  String host_name = line[3];
68                  String neighbourhood = line[4];
69                  double latitude = convertDouble(line[5]);
70                  double longitude = convertDouble(line[6]);
71                  String room_type = line[7];
72                  int price = convertInt(line[8]);
73                  int minimumNights = convertInt(line[9]);
74                  int availability365 = convertInt(line[10]);
75
76                  Property currentProperty = new Property(id, name, host_id,
                    host_name,
77                      neighbourhood, latitude,longitude, room_type, price,
78                      minimumNights, availability365);
79                  listings.add(currentProperty);
80              }
81          } catch(IOException | URISyntaxException e){
82              System.out.println("Failure! Something went wrong when loading the
                property file");
83              e.printStackTrace();
84          }
85          System.out.println("Success! Number of loaded records: " + listings.size());
86          return listings;
87      }
88
89      /**
90       *
91       * @param doubleString the string to be converted to Double type
92       * @return the Double value of the string, or -1.0 if the string is
93       * either empty or just whitespace
94       */
95      private Double convertDouble(String doubleString){
```

```
96          if(doubleString != null && !doubleString.trim().equals("")){
97              return Double.parseDouble(doubleString);
98          }
99          return -1.0;
100     }
101
102     /**
103      *
104      * @param intString the string to be converted to Integer type
105      * @return the Integer value of the string, or -1 if the string is
106      * either empty or just whitespace
107      */
108     private Integer convertInt(String intString){
109         if(intString != null && !intString.trim().equals("")){
110             return Integer.parseInt(intString);
111         }
112         return -1;
113     }
114 }
115
```

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4  import javax.swing.border.*;
5
6  import java.io.File;
7
8  import java.util.List;
9  import java.util.ArrayList;
10 import java.util.Iterator;
11
12 /**
13  * PropertyViewerGUI provides the GUI for the project. It displays the property
14  * and strings, and it listens to button clicks.
15
16  * @author Michael Kölling, David J Barnes, and Josh Murphy
17  * @version 3.0
18  *
19  * Modified by Emre Salur K20082471
20  *
21  * Added a void statistics method
22  * Created a statistictsButton inside the makeFrame method
23  *
24  */
25 public class PropertyViewerGUI
26 {
27     // fields:
28     private JFrame frame;
29     private JPanel propertyPanel;
30     private JLabel idLabel;
31     private JLabel favouriteLabel;
32
33     private JTextField hostIDLabel;
34     private JTextField hostNameLabel;
35     private JTextField neighbourhoodLabel;
36     private JTextField roomTypeLabel;
37     private JTextField priceLabel;
38     private JTextField minNightsLabel;
39     private JTextArea descriptionLabel;
40
41     private Property currentProperty;
42     private PropertyViewer viewer;
43     private boolean fixedSize;
44
45     /**
46      * Create a PropertyViewer and display its GUI on screen.
47      */
48     public PropertyViewerGUI(PropertyViewer viewer)
49     {
50         currentProperty = null;
```

```
51          this.viewer = viewer;
52          fixedSize = false;
53          makeFrame();
54          this.setPropertyViewSize(400, 250);
55      }
56
57
58      // ---- public view functions ----
59
60      /**
61       * Display a given property
62       */
63      public void showProperty(Property property)
64      {
65
66          hostIDLabel.setText(property.getHostID());
67          hostNameLabel.setText(property.getHostName());
68          neighbourhoodLabel.setText(property.getNeighbourhood());
69          roomTypeLabel.setText(property.getRoomType());
70          priceLabel.setText("£" + property.getPrice());
71          minNightsLabel.setText(property.getMinNights());
72          //descriptionLabel.setText(property.getDescription());
73      }
74
75      /**
76       * Set a fixed size for the property display. If set, this size will be used for
    all properties.
77       * If not set, the GUI will resize for each property.
78       */
79      public void setPropertyViewSize(int width, int height)
80      {
81          propertyPanel.setPreferredSize(new Dimension(width, height));
82          frame.pack();
83          fixedSize = true;
84      }
85
86      /**
87       * Show a message in the status bar at the bottom of the screen.
88       */
89      public void showFavourite(Property property)
90      {
91          String favouriteText = " ";
92          if (property.isFavourite()){
93              favouriteText += "This is one of your favourite properties!";
94          }
95          favouriteLabel.setText(favouriteText);
96      }
97
98      /**
99       * 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();
}
/**
 * Called when the 'Statistics' button was clicked.
 */
private void statisticsButton(){
    viewer.statistics();
}

```

```
150     // ---- swing stuff to build the frame and all its components ----
151
152     /**
153      * Create the Swing frame and its content.
154      */
155     private void makeFrame()
156     {
157         frame = new JFrame("Portfolio Viewer Application");
158         JPanel contentPane = (JPanel)frame.getContentPane();
159         contentPane.setBorder(new EmptyBorder(6, 6, 6, 6));
160
161         // Specify the layout manager with nice spacing
162         contentPane.setLayout(new BorderLayout(6, 6));
163
164         // Create the property pane in the center
165         propertyPanel = new JPanel();
166         propertyPanel.setLayout(new GridLayout(6,2));
167
168         propertyPanel.add(new JLabel("HostID: "));
169         hostIDLabel = new JTextField("default");
170         hostIDLabel.setEditable(false);
171         propertyPanel.add(hostIDLabel);
172
173         propertyPanel.add(new JLabel("Host Name: "));
174         hostNameLabel = new JTextField("default");
175         hostNameLabel.setEditable(false);
176         propertyPanel.add(hostNameLabel);
177
178         propertyPanel.add(new JLabel("Neighbourhood: "));
179         neighbourhoodLabel = new JTextField("default");
180         neighbourhoodLabel.setEditable(false);
181         propertyPanel.add(neighbourhoodLabel);
182
183         propertyPanel.add(new JLabel("Room type: "));
184         roomTypeLabel = new JTextField("default");
185         roomTypeLabel.setEditable(false);
186         propertyPanel.add(roomTypeLabel);
187
188         propertyPanel.add(new JLabel("Price: "));
189         priceLabel = new JTextField("default");
190         priceLabel.setEditable(false);
191         propertyPanel.add(priceLabel);
192
193         propertyPanel.add(new JLabel("Minimum nights: "));
194         minNightsLabel = new JTextField("default");
195         minNightsLabel.setEditable(false);
196         propertyPanel.add(minNightsLabel);
197
198         propertyPanel.setBorder(new EtchedBorder());
199         contentPane.add(propertyPanel, BorderLayout.CENTER);
```

```
200
201         // Create two labels at top and bottom for the file name and status message
202         idLabel = new JLabel("default");
203         contentPane.add(idLabel, BorderLayout.NORTH);
204
205         favouriteLabel = new JLabel(" ");
206         contentPane.add(favouriteLabel, BorderLayout.SOUTH);
207
208         // Create the toolbar with the buttons
209         JPanel toolbar = new JPanel();
210         toolbar.setLayout(new GridLayout(0, 1));
211
212         JButton nextButton = new JButton("Next");
213         nextButton.addActionListener(new ActionListener() {
214                                 public void actionPerformed(ActionEvent e) {
nextButton(); }
215                             });
216         toolbar.add(nextButton);
217
218         JButton previousButton = new JButton("Previous");
219         previousButton.addActionListener(new ActionListener() {
220                                 public void actionPerformed(ActionEvent e) {
previousButton(); }
221                             });
222         toolbar.add(previousButton);
223
224         JButton mapButton = new JButton("View Property on Map");
225         mapButton.addActionListener(new ActionListener() {
226                                 public void actionPerformed(ActionEvent e) {
viewOnMapsButton(); }
227                             });
228         toolbar.add(mapButton);
229
230         JButton favouriteButton = new JButton("Toggle Favourite");
231         favouriteButton.addActionListener(new ActionListener() {
232                                 public void actionPerformed(ActionEvent e) {
toggleFavouriteButton(); }
233                             });
234         toolbar.add(favouriteButton);
235
236         JButton statisticsButton = new JButton("Statistics");       //create
Statistics button
237         statisticsButton.addActionListener(new ActionListener() {
238                                 public void actionPerformed(ActionEvent e) {
statisticsButton(); }
239                             });
240         toolbar.add(statisticsButton);
241
242         // Add toolbar into panel with flow layout for spacing
243         JPanel flow = new JPanel();
```

```
244         flow.add(toolbar);
245
246         contentPane.add(flow, BorderLayout.WEST);
247
248         // building is done - arrange the components
249         frame.pack();
250
251         // place the frame at the center of the screen and show
252         Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
253         frame.setLocation(d.width/2 - frame.getWidth()/2, d.height/2 -
frame.getHeight()/2);
254         frame.setVisible(true);
255     }
256 }
257
```

```
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;
    }

```

```
50      /**
51       * Return the Id of this property.
52       */
53      public String getID(){
54          return id;
55      }
56
57      /**
58       * Return the hostId of this property.
59       */
60      public String getHostID(){
61          return hostID;
62      }
63
64      /**
65       * Return the latitude of this property.
66       */
67      public double getLatitude(){
68          return latitude;
69      }
70
71      /**
72       * Return the longitude of this property.
73       */
74      public double getLongitude(){
75          return longitude;
76      }
77
78      /**
79       * Return the price of this property.
80       */
81      public int getPrice(){
82          return price;
83      }
84
85      /**
86       * Returns true if this property is currently marked as a favourite, false
    otherwise.
87       */
88      public boolean isFavourite(){
89          return isFavourite;
90      }
91
92      /**
93       * Return the host name of this property.
94       */
95      public String getHostName(){
96          return hostName;
97      }
98
```

```
99      /**
100      * Return the neighbourhood of this property.
101      */
102     public String getNeighbourhood(){
103         return neighbourhood;
104     }
105
106     /**
107      * Return the room type of this property.
108      */
109     public String getRoomType(){
110         return roomType;
111     }
112
113     /**
114      * Return the minimum number of nights this property can be booked for.
115      */
116     public String getMinNights(){
117         return "" + minimumNights;
118     }
119
120     /**
121      * Return the description of this property.
122      */
123     public String getDescription(){
124         return description;
125     }
126
127     /**
128      * Toggles whether this property is marked as a favourite or not.
129      */
130     public void toggleFavourite()
131     {
132         isFavourite = !isFavourite;
133     }
134
135 }
136
```

```
class ProperyViewer:

    variables:
        PropertyViewerGUI gui: the Graphical User Interface
        Portfolio portfolio: contatins an arrayList of properties
        int index: index number for each property inside the portfolio
        int propertyViewed: number of properties viewed in the application
        int totalPrice: calculates the total price of each viewed property

    methods:

        PropertyViewer(): Acts as a constructor by creating a PropertyViewer object and displaying its GUI on screen.
        nextProperty(): Displays the next property and updates information accordingly
        previousProperty(): Displays the previous property and updates information accordingly
        toggleFavourite(): Updates and displays whether a property is favourite or not
        viewMap(): Displays the location of the property in Google Maps
        getNumberOfPropertiesViewed(): Returns the number of properties viewed since the start of the application
        averagePropertyPrice(): Returns the average property price of viewed properties since the start of the application
        statistics(): Displays the statistics information from the getNumberOfPropertiesViewed and averagePropertyPrice methods
            JFrame statisticsFrame: New window to display statistics
            JLabel propertyViewedLabel: Label that shows the number of properties viewed
            JlLabel averagePriceLabel: Label that shows the average price of the viewed properties

class PropertyViewerGUI:

    methods:
        statisticsButton(): Called when the 'Statistics' button was clicked.
        makeFrame(): Added a statistics button by looking at how the other buttons were initialized
```