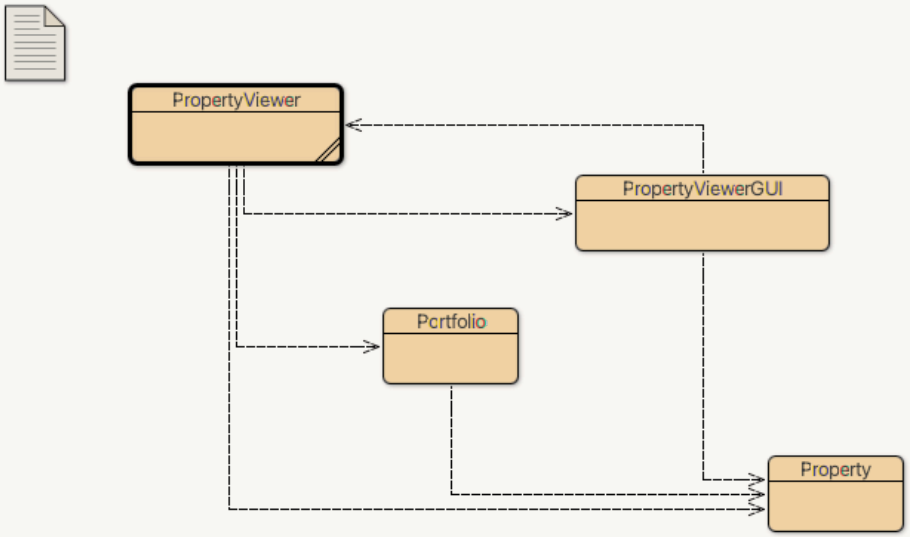
PropertyViewer
PropertyViewerGUI
Portfolio
Property

```java
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.
 * Details about the interaciton with properties can be viewed.
 *
 * @author Michael Kölling and Josh Murphy and Zahra Amaan K21011879
 * @version 1.0
 */
public class PropertyViewer
{
    private PropertyViewerGUI gui;     // the Graphical User Interface
    private Portfolio portfolio;
    private Property currentProperty;
    int propertyIndex = 0;
    int totalViews = 0;
    int totalPrice;

    /**
     * Create a PropertyViewer and display its GUI on screen.
     */
    public PropertyViewer()
    {
        gui = new PropertyViewerGUI(this);
        portfolio = new Portfolio("airbnb-london.csv");
        this.viewProperty(0);
    }

    /**
     * Displays a property, its ID and whether it is a favourite.
     */
    public void viewProperty(int index)
    {
        currentProperty = portfolio.getProperty(index);
        gui.showProperty(currentProperty);
        gui.showID(currentProperty);
        gui.showFavourite(currentProperty);
        totalViews += 1;
        totalPrice += currentProperty.getPrice();
    }

    /**
     * Called when the 'Next' button is clicked.
     * Displays the next property in the portfolio.
     */
    public void nextProperty()
    {
        // 'propertyIndex' is incremented by 1. When the 'Next' button is clicked while viewing the last property in the portfolio, the 'propertyIndex' is set to 0 and the first property in the portfolio is displayed.
        propertyIndex += 1;
```

```
53          if (propertyIndex == portfolio.numberOfProperties())
54          {
55              propertyIndex = 0;
56          }
57          viewProperty(propertyIndex);
58      }
59
60      /**
61       * Called when the 'Previuos' button is clicked.
62       * Displays the previous property in the portfolio.
63       */
64      public void previousProperty()
65      {
66          // 'propertyIndex' is decremented by 1. When the 'Previous' button is
clicked while viewing the first property in the portfolio, the last property in
the portfolio is displayed.
67          propertyIndex -= 1;
68          if (propertyIndex == -1)
69          {
70              propertyIndex = portfolio.numberOfProperties() - 1;
71          }
72          viewProperty(propertyIndex);
73      }
74
75      /**
76       * Called when the 'Toggle Favourite' button is clicked.
77       * If the current property is not a favourite, it becomes a favourite.
78       * If the current property is a favouite, it becomes not a favourite.
79       * When a property is a favourite, a message is displayed and when a property
is not a favourite, the message is removed.
80       */
81      public void toggleFavourite()
82      {
83          currentProperty.toggleFavourite();
84          gui.showFavourite(currentProperty);
85      }
86
87
88      //----- methods for challenge tasks -----
89
90      /**
91       * This method opens the system's default internet browser
92       * The Google maps page should show the current properties location on the
map.
93       */
94      public void viewMap() throws Exception
95      {
96          double latitude = currentProperty.getLatitude();
97          double longitude = currentProperty.getLongitude();
98
99          URI uri = new URI("https://www.google.com/maps/place/" + latitude + "," +
longitude);
100         java.awt.Desktop.getDesktop().browse(uri);
101     }
102
```

```
103     /**
104      * Returns the total number of properties that have been viewed so far.
105      */
106     public int getNumberOfPropertiesViewed()
107     {
108         return totalViews;
109     }
110 
111     /**
112      * Calculates and returns the average property price of all the properties
that have been viewed so far.
113      */
114     public int averagePropertyPrice()
115     {
116         int averagePrice = totalPrice/totalViews;
117         return averagePrice;
118     }
119 }
120 
```

```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();
    }

     /**
     * Return an ArrayList containing the rows in the AirBnB London data set csv file.
     */
```

```
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
```

```
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
```

```java
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.
 * It displays statistics about the properties.
 *
 * @author Michael Kölling, David J Barnes, Josh Murphy, and Zahra Amaan K21011879
 * @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;


    private JFrame statsFrame;

    private JTextField viewsLabelText;
    private JTextField averagePriceLabelText;

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

```
55          this.setPropertyViewSize(400, 250);
56      }
57
58
59      // ---- public view functions ----
60
61      /**
62       * Display a given property
63       */
64      public void showProperty(Property property)
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.
100      */
101     public void showID(Property property){
102         idLabel.setText("Current Property ID:" + property.getID());
103     }
104
105     // ---- implementation of button functions ----
106
107     /**
```

```
108         * Called when the 'Next' button was clicked.
109         */
110        private void nextButton()
111        {
112            viewer.nextProperty();
113        }
114
115        /**
116         * Called when the 'Previous' button was clicked.
117         */
118        private void previousButton()
119        {
120            viewer.previousProperty();
121        }
122
123        /**
124         * Called when the 'View on Map' button was clicked.
125         */
126        private void viewOnMapsButton()
127        {
128            try{
129             viewer.viewMap();
130            }
131            catch(Exception e){
132                System.out.println("URL INVALID");
133            }
134
135        }
136
137        /**
138         * Called when the 'Toggle Favourite' button was clicked.
139         */
140        private void toggleFavouriteButton(){
141            viewer.toggleFavourite();
142        }
143
144        /**
145         * Called when the 'Statistics' button was clicked.
146         * Displasys a GUI on screen.
147         */
148        private void statisticsButton(){
149            statsFrame();
150
151            String stringViews = String.valueOf(viewer.getNumberOfPropertiesViewed());// gets the integer value for 'views' from the viewer class and converts it to a String.
152            viewsLabelText.setText(stringViews);
153
154            String stringAverage = String.valueOf(viewer.averagePropertyPrice());// gets the integer value for 'averagePropertyPrice' from the viewer class and converts it to a String.
155            averagePriceLabelText.setText(stringAverage);
156        }
157
158        // ---- swing stuff to build the frame and all its components ----
```

```
159
160     /**
161      * Create the Swing frame and its content.
162      */
163     private void makeFrame()
164     {
165         frame = new JFrame("Portfolio Viewer Application");
166         JPanel contentPane = (JPanel)frame.getContentPane();
167         contentPane.setBorder(new EmptyBorder(6, 6, 6, 6));
168
169         // Specify the layout manager with nice spacing
170         contentPane.setLayout(new BorderLayout(6, 6));
171
172         // Create the property pane in the center
173         propertyPanel = new JPanel();
174         propertyPanel.setLayout(new GridLayout(6,2));
175
176         propertyPanel.add(new JLabel("HostID: "));
177         hostIDLabel = new JTextField("default");
178         hostIDLabel.setEditable(false);
179         propertyPanel.add(hostIDLabel);
180
181         propertyPanel.add(new JLabel("Host Name: "));
182         hostNameLabel = new JTextField("default");
183         hostNameLabel.setEditable(false);
184         propertyPanel.add(hostNameLabel);
185
186         propertyPanel.add(new JLabel("Neighbourhood: "));
187         neighbourhoodLabel = new JTextField("default");
188         neighbourhoodLabel.setEditable(false);
189         propertyPanel.add(neighbourhoodLabel);
190
191         propertyPanel.add(new JLabel("Room type: "));
192         roomTypeLabel = new JTextField("default");
193         roomTypeLabel.setEditable(false);
194         propertyPanel.add(roomTypeLabel);
195
196         propertyPanel.add(new JLabel("Price: "));
197         priceLabel = new JTextField("default");
198         priceLabel.setEditable(false);
199         propertyPanel.add(priceLabel);
200
201         propertyPanel.add(new JLabel("Minimum nights: "));
202         minNightsLabel = new JTextField("default");
203         minNightsLabel.setEditable(false);
204         propertyPanel.add(minNightsLabel);
205
206         propertyPanel.setBorder(new EtchedBorder());
207         contentPane.add(propertyPanel, BorderLayout.CENTER);
208
209         // Create two labels at top and bottom for the file name and status
    message
210         idLabel = new JLabel("default");
211         contentPane.add(idLabel, BorderLayout.NORTH);
212
```

```
213         favouriteLabel = new JLabel(" ");
214         contentPane.add(favouriteLabel, BorderLayout.SOUTH);
215 
216         // Create the toolbar with the buttons
217         JPanel toolbar = new JPanel();
218         toolbar.setLayout(new GridLayout(0, 1));
219 
220         JButton nextButton = new JButton("Next");
221         nextButton.addActionListener(new ActionListener() {
222                                 public void actionPerformed(ActionEvent e) {
nextButton(); }
223                                 });
224         toolbar.add(nextButton);
225 
226         JButton previousButton = new JButton("Previous");
227         previousButton.addActionListener(new ActionListener() {
228                                 public void actionPerformed(ActionEvent e) {
previousButton(); }
229                                 });
230         toolbar.add(previousButton);
231 
232         JButton mapButton = new JButton("View Property on Map");
233         mapButton.addActionListener(new ActionListener() {
234                                 public void actionPerformed(ActionEvent e) {
viewOnMapsButton(); }
235                                 });
236         toolbar.add(mapButton);
237 
238         JButton favouriteButton = new JButton("Toggle Favourite");
239         favouriteButton.addActionListener(new ActionListener() {
240                                 public void actionPerformed(ActionEvent e) {
toggleFavouriteButton(); }
241                                 });
242         toolbar.add(favouriteButton);
243 
244         JButton statisticsButton = new JButton("Statistics");
245         statisticsButton.addActionListener(new ActionListener() {
246                                 public void actionPerformed(ActionEvent e) {
statisticsButton(); }
247                                 });
248         toolbar.add(statisticsButton);
249 
250         // Add toolbar into panel with flow layout for spacing
251         JPanel flow = new JPanel();
252         flow.add(toolbar);
253 
254         contentPane.add(flow, BorderLayout.WEST);
255 
256         // building is done - arrange the components
257         frame.pack();
258 
259         // place the frame at the center of the screen and show
260         Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
```

```
261             frame.setLocation(d.width/2 - frame.getWidth()/2, d.height/2 -
    frame.getHeight()/2);
262             frame.setVisible(true);
263         }
264 
265         /**
266          * Creates a Swing frame and its contents.
267          * Displays the statistics: total number of properties viewed and their
    average price.
268          * This code is adapted from the 'makeFrame' method in the 'PropertyViewerGUI'
    class.
269          */
270         private void statsFrame(){
271             statsFrame = new JFrame("Statistics");
272             JPanel contentPanel = (JPanel)statsFrame.getContentPane();
273             contentPanel.setBorder(new EmptyBorder(6, 6, 6, 6));
274 
275             // Specify the layout manager with nice spacing
276             contentPanel.setLayout(new BorderLayout(6, 6));
277 
278             // Create the statsPanel
279             JPanel statsPanel = new JPanel();
280             statsPanel.setLayout(new GridLayout(6,2));
281 
282             // Create a label and text field to display the total views and add to
    statsPanel
283             statsPanel.add(new JLabel("Views: "));
284             viewsLabelText = new JTextField("default");
285             viewsLabelText.setEditable(false);
286             statsPanel.add(viewsLabelText);
287 
288             // Create a label and text field to display the average propety price and
    add to statsPanel
289             statsPanel.add(new JLabel("Average price: "));
290             averagePriceLabelText = new JTextField("default");
291             averagePriceLabelText.setEditable(false);
292             statsPanel.add(averagePriceLabelText);
293 
294             //add statsPanel to contentPanel
295             contentPanel.add(statsPanel, BorderLayout.CENTER);
296 
297             // arrange the components and make the frame visible
298             statsFrame.pack();
299             statsFrame.setVisible(true);
300         }
301 }
302 
```

```java
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;
```

```
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
```

```
1
```