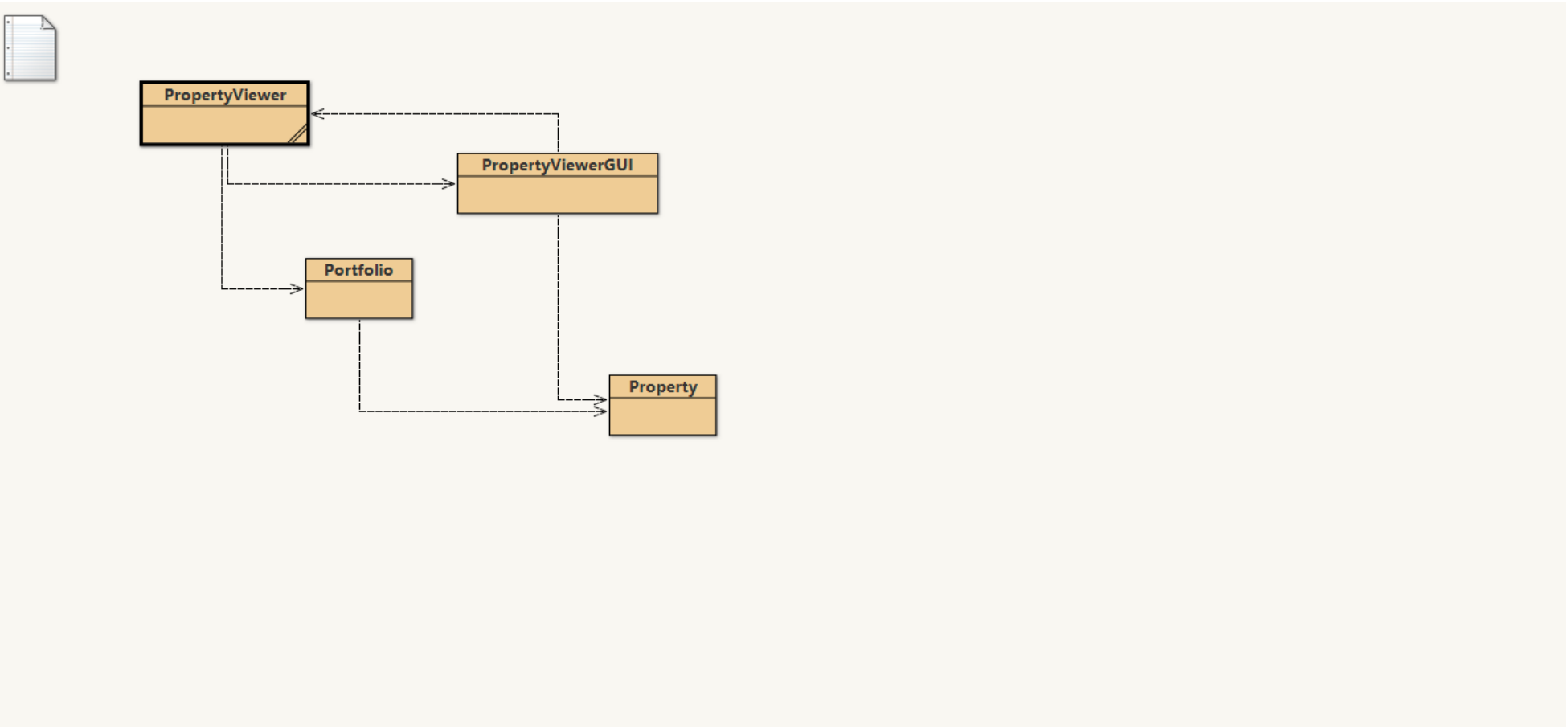
PropertyViewer
PropertyViewerGUI
Portfolio
Property

```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.
 *
 *
 * @author Michael Kölling and Josh Murphy and Yeshuai Cui(k1924221)
 * @version 1.0
 */
public class PropertyViewer
{
    private PropertyViewerGUI gui;     // the Graphical User Interface
    private Portfolio portfolio;
    private int propertyIndex=0;
    //first property with index 0
    private int viewTimes=0;
    //challenge task 1. record the total property viewed times
    private int priceSum=0;
    //challenge task 2. record the total price of property viewed
    /**
     * Create a PropertyViewer and display its GUI on screen.
     */
    public PropertyViewer()
    {
        gui = new PropertyViewerGUI(this);
        portfolio = new Portfolio("airbnb-london.csv");
        display();//display the content.
        priceSum = portfolio.getProperty(0).getPrice();
        //first property's price is added.
        viewTimes += 1;
        //viewTimes is 1 as first property is displayed.
    }

    public void display()
    {
        gui.showProperty(portfolio.getProperty(propertyIndex));
        //task1. The property with propertyIndex is displayed, first property
        //is displayed at start as PropertyIndex is initialised with value 0.
        gui.showID(portfolio.getProperty(propertyIndex));
        //task2. Use method in PropertyViewerGUI to display ID around top of
        //the window.
        gui.showFavourite(portfolio.getProperty(propertyIndex));
        //task4. Use method in PropertyViewerGUI to display wherther or not
        //this is the favourite property.
    }

    /**
     *Show next property by increment the index.
     *else statement includes technic preventing wrong index go further.
     */
    public void nextProperty()
```

```
55      {
56          propertyIndex = propertyIndex + 1;//increment the index
57          if(propertyIndex>=0&&propertyIndex<portfolio.numberOfProperties()){
58              display();//update the content
59              viewTimes = viewTimes + 1;//increment the view times
60              priceSum = priceSum +
portfolio.getProperty(propertyIndex).getPrice();
61              //add the price of this property to the total price viewed.
62          }
63          else{
64              propertyIndex = portfolio.numberOfProperties()-1;
65              System.out.println("You've reach the limit, there's no more
property!");
66          }
67      }
68
69      /**
70       *Show previous property by decrement the index.
71       *else statement include technic preventing wrong index go further.
72       */
73      public void previousProperty()
74      {
75          propertyIndex = propertyIndex - 1;//decrement the index
76          if(propertyIndex>=0&&propertyIndex<portfolio.numberOfProperties()){
77              display();//update the content
78              viewTimes = viewTimes+1;//increment the view times
79              priceSum = priceSum +
portfolio.getProperty(propertyIndex).getPrice();
80          }
81          else{
82              propertyIndex = 0;
83              System.out.println("You've reach the limit, there's no more
property!");
84          }
85      }
86
87      /**
88       *Change the favourite status by calling method in Property
89       */
90      public void toggleFavourite()
91      {
92          portfolio.getProperty(propertyIndex).toggleFavourite();
93          //task3. the isFavourite field of that property is updated
94          display();// instantly display the content without going back again
95      }
96
97
98      //----- methods for challenge tasks -----
99
100     /**
101      * This method opens the system's default internet browser
102      * The Google maps page should show the current properties location on the
map.
103      */
```

```
104     public void viewMap() throws Exception
105     {
106        double latitude = portfolio.getProperty(propertyIndex).getLatitude();
107        //get the latitude from property by using getProperty method.
108        double longitude = portfolio.getProperty(propertyIndex).getLongitude();
109        //get the longitude from property by using getProperty method.
110        URI uri = new URI("https://www.google.com/maps/place/" + latitude + "," +
    longitude);
111        java.awt.Desktop.getDesktop().browse(uri);
112     }
113
114     /**
115      * Challenge Task 1. Get total view times by return viewTime field.
116      */
117     public int getNumberOfPropertiesViewed()
118     {
119         return viewTimes ;
120     }
121
122     /**
123      * Challenge task 2 return average value for viewed properties' prices,
124      * add first property's price here.
125      */
126     public double getAveragePropertyPrice()
127     {
128         return priceSum/viewTimes;
129     }
130
131 }
132
```

```
1  import java.util.List;
2  import java.util.ArrayList;
3  import java.io.File;
4  import java.io.BufferedReader;
5  import java.io.File;
6  import java.io.FileReader;
7  import java.io.IOException;
8  import java.net.URL;
9  import java.util.ArrayList;
10 import java.util.Arrays;
11 import com.opencsv.CSVReader;
12 import java.net.URISyntaxException;
13
14 /**
15  * A portfolio is a collection of properties. It reads properties from a file on
   disk,
16  * and it can be used to retrieve single properties.
17  *
18  * The file name to read from is passed in at construction.
19  *
20  * @author Michael Kölling and Josh Murphy
21  * @version 1.0
22  */
23 public class Portfolio
24 {
25     private List<Property> properties;
26
27     /**
28      * Constructor for objects of class Portfolio
29      */
30     public Portfolio(String directoryName)
31     {
32         properties = loadProperties();
33     }
34
35     /**
36      * Return a property from this Portfolio.
37      */
38     public Property getProperty(int propertyNumber)
39     {
40         return properties.get(propertyNumber);
41     }
42
43     /**
44      * Return the number of Properties in this Portfolio.
45      */
46     public int numberOfProperties()
47     {
48         return properties.size();
49     }
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    }
116
```

```
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, Josh Murphy, Yeshuai Cui(k1924221)
17  * @version 3.0
18  */
19 public class PropertyViewerGUI
20 {
21     // fields:
22     private JFrame frame;
23     private JPanel propertyPanel;
24     private JPanel statisticsPanel;
25     private JLabel idLabel;
26     private JLabel favouriteLabel;
27
28     private JTextField hostIDLabel;
29     private JTextField hostNameLabel;
30     private JTextField neighbourhoodLabel;
31     private JTextField roomTypeLabel;
32     private JTextField priceLabel;
33     private JTextField minNightsLabel;
34     private JTextField viewTimesLabel;
35     private JTextField averagePriceLabel;
36     private JTextArea descriptionLabel;
37
38     private Property currentProperty;
39     private PropertyViewer viewer;
40     private boolean fixedSize;
41
42     /**
43      * Create a PropertyViewer and display its GUI on screen.
44      */
45     public PropertyViewerGUI(PropertyViewer viewer)
46     {
47         currentProperty = null;
48         this.viewer = viewer;
49         fixedSize = false;
50         makeFrame();
51         this.setPropertyViewSize(400, 250);
52     }
53
54
```

```
55      // ---- public view functions ----
56
57      /**
58       * Display a given property
59       */
60      public void showProperty(Property property)
61      {
62          hostIDLabel.setText(property.getHostID());
63          hostNameLabel.setText(property.getHostName());
64          neighbourhoodLabel.setText(property.getNeighbourhood());
65          roomTypeLabel.setText(property.getRoomType());
66          priceLabel.setText("£" + property.getPrice());
67          minNightsLabel.setText(property.getMinNights());
68          //descriptionLabel.setText(property.getDescription());
69      }
70
71      /**
72       * Set a fixed size for the property display. If set, this size will be used
    for all properties.
73       * If not set, the GUI will resize for each property.
74       */
75      public void setPropertyViewSize(int width, int height)
76      {
77          propertyPanel.setPreferredSize(new Dimension(width, height));
78          frame.pack();
79          fixedSize = true;
80      }
81
82      /**
83       * Set a fixed size for the Statistics dispaly.
84       */
85      public void setStatisticsViewSize()
86      {
87          statisticsPanel.setPreferredSize(new Dimension(250,400));
88          frame.pack();
89          fixedSize = true;
90      }
91
92      /**
93       * Show a message in the status bar at the bottom of the screen.
94       */
95      public void showFavourite(Property property)
96      {
97          String favouriteText = " ";
98          if (property.isFavourite()){
99              favouriteText += "This is one of your favourite properties!";
100         }
101         else{
102             favouriteText += "This is not one of your favourite properties!";
103         }
104         //display whether or not this is the favourite property.
105         favouriteLabel.setText(favouriteText);
106         //display the text
107     }
```

```
108
109     /**
110      * Show the ID in the top of the screen.
111      */
112     public void showID(Property property){
113         idLabel.setText("Current Property ID:" + property.getID());
114     }
115
116
117
118     // ---- implementation of button functions ----
119
120     /**
121      * Called when the 'Next' button was clicked.
122      */
123     private void nextButton()
124     {
125         viewer.nextProperty();
126     }
127
128     /**
129      * Called when the 'Previous' button was clicked.
130      */
131     private void previousButton()
132     {
133         viewer.previousProperty();
134     }
135
136     /**
137      * Called when the 'View on Map' button was clicked.
138      */
139     private void viewOnMapsButton()
140     {
141         try{
142          viewer.viewMap();
143         }
144         catch(Exception e){
145             System.out.println("URL INVALID");
146         }
147
148     }
149
150     /**
151      * Called when the 'Toggle Favourite' button was clicked.
152      */
153     private void toggleFavouriteButton(){
154         viewer.toggleFavourite();
155     }
156
157     /**
158      * Called when the "statistics" button was clicked.
159      */
160     private void statisticsButton(){
161         makeStatisticsFrame();
```

```
162         viewTimesLabel.setText(viewer.getNumberOfPropertiesViewed()+"");
163         averagePriceLabel.setText("£"+viewer.getAveragePropertyPrice());
164     }
165     // ---- swing stuff to build the frame and all its components ----
166 
167     /**
168      * Create the Swing frame and its content.
169      */
170     private void makeFrame()
171     {
172         frame = new JFrame("Portfolio Viewer Application");
173         JPanel contentPane = (JPanel)frame.getContentPane();
174         contentPane.setBorder(new EmptyBorder(6, 6, 6, 6));
175 
176         // Specify the layout manager with nice spacing
177         contentPane.setLayout(new BorderLayout(6, 6));
178 
179         // Create the property pane in the center
180         propertyPanel = new JPanel();
181         propertyPanel.setLayout(new GridLayout(6,2));
182 
183         propertyPanel.add(new JLabel("HostID: "));
184         hostIDLabel = new JTextField("default");
185         hostIDLabel.setEditable(false);
186         propertyPanel.add(hostIDLabel);
187 
188         propertyPanel.add(new JLabel("Host Name: "));
189         hostNameLabel = new JTextField("default");
190         hostNameLabel.setEditable(false);
191         propertyPanel.add(hostNameLabel);
192 
193         propertyPanel.add(new JLabel("Neighbourhood: "));
194         neighbourhoodLabel = new JTextField("default");
195         neighbourhoodLabel.setEditable(false);
196         propertyPanel.add(neighbourhoodLabel);
197 
198         propertyPanel.add(new JLabel("Room type: "));
199         roomTypeLabel = new JTextField("default");
200         roomTypeLabel.setEditable(false);
201         propertyPanel.add(roomTypeLabel);
202 
203         propertyPanel.add(new JLabel("Price: "));
204         priceLabel = new JTextField("default");
205         priceLabel.setEditable(false);
206         propertyPanel.add(priceLabel);
207 
208         propertyPanel.add(new JLabel("Minimum nights: "));
209         minNightsLabel = new JTextField("default");
210         minNightsLabel.setEditable(false);
211         propertyPanel.add(minNightsLabel);
212 
213         propertyPanel.setBorder(new EtchedBorder());
214         contentPane.add(propertyPanel, BorderLayout.CENTER);
215 
```

```
216         // Create two labels at top and bottom for the file name and status
    message
217         idLabel = new JLabel("default");
218         contentPane.add(idLabel, BorderLayout.NORTH);
219
220         favouriteLabel = new JLabel(" ");
221         contentPane.add(favouriteLabel, BorderLayout.SOUTH);
222
223         // Create the toolbar with the buttons
224         JPanel toolbar = new JPanel();
225         toolbar.setLayout(new GridLayout(0, 1));
226
227         JButton nextButton = new JButton("Next");
228         nextButton.addActionListener(new ActionListener() {
229                             public void actionPerformed(ActionEvent e) {
    nextButton(); }
230                         });
231         toolbar.add(nextButton);
232
233         JButton previousButton = new JButton("Previous");
234         previousButton.addActionListener(new ActionListener() {
235                             public void actionPerformed(ActionEvent e) {
    previousButton(); }
236                         });
237         toolbar.add(previousButton);
238
239         JButton mapButton = new JButton("View Property on Map");
240         mapButton.addActionListener(new ActionListener() {
241                             public void actionPerformed(ActionEvent e) {
    viewOnMapsButton(); }
242                         });
243         toolbar.add(mapButton);
244
245         JButton favouriteButton = new JButton("Toggle Favourite");
246         favouriteButton.addActionListener(new ActionListener() {
247                             public void actionPerformed(ActionEvent e) {
    toggleFavouriteButton(); }
248                         });
249         toolbar.add(favouriteButton);
250
251         JButton statisticsButton = new JButton("Statistics");
252         statisticsButton.addActionListener(new ActionListener() {
253                             public void actionPerformed(ActionEvent e) {
    statisticsButton(); }
254                         });
255         toolbar.add(statisticsButton);
256
257
258         // Add toolbar into panel with flow layout for spacing
259         JPanel flow = new JPanel();
260         flow.add(toolbar);
261
262         contentPane.add(flow, BorderLayout.WEST);
263
```

```
264         // building is done - arrange the components
265         frame.pack();
266 
267         // place the frame at the center of the screen and show
268         Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
269         frame.setLocation(d.width/2 - frame.getWidth()/2, d.height/2 -
    frame.getHeight()/2);
270         frame.setVisible(true);
271     }
272 
273     /**
274      * create window for statistics
275      */
276     private void makeStatisticsFrame()
277     {
278         frame = new JFrame("Statistics Viewer Application");
279         JPanel contentPane = (JPanel)frame.getContentPane();
280         contentPane.setBorder(new EmptyBorder(6, 6, 6, 6));
281 
282         // Specify the layout manager with nice spacing
283         contentPane.setLayout(new BorderLayout(6, 6));
284 
285         // Create the property pane in the center
286         statisticsPanel = new JPanel();
287         statisticsPanel.setLayout(new GridLayout(6,2));
288 
289         statisticsPanel.add(new JLabel("total view times"));
290         viewTimesLabel = new JTextField("default");
291         viewTimesLabel.setEditable(false);
292         statisticsPanel.add(viewTimesLabel);
293 
294         statisticsPanel.add(new JLabel("average price of property viewed"));
295         averagePriceLabel = new JTextField("default");
296         averagePriceLabel.setEditable(false);
297         statisticsPanel.add(averagePriceLabel);
298 
299         statisticsPanel.setBorder(new EtchedBorder());
300         contentPane.add(statisticsPanel, BorderLayout.CENTER);
301         frame.pack();
302 
303         // place the frame at the center of the screen and show
304         Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
305         frame.setLocation(d.width/2 - frame.getWidth()/2, d.height/2 -
    frame.getHeight()/2);
306         frame.setVisible(true);
307     }
308 }
309 
```

```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(){
```

```
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
```