Portfolio.java

```java
import com.opencsv.CSVReader;

import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

/**
 * A portfolio is a collection of properties. It reads properties from a file on disk,
 * and it can be used to retrieve single properties.
 * <p>
 * The file name to read from is passed in at construction. Student name: Dylan Barker, K-
number: 20001430.
 *
 * @author Michael Kölling and Josh Murphy
 * @version 2.0
 */
public class Portfolio {
    private final List<Property> properties;

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

    /**
     * @param propertyNumber The index of the property in the portfolio.
     * @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<>();
        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 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]);

                Property currentProperty = new Property(id, host_id, host_name,
                        neighbourhood, latitude, longitude, room_type, price,
                        minimumNights);
                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;
        }
    }

```

Property.java

```java
/**
 * Property is a class that defines a property for display.
 * <p>
 * Student name: Dylan Barker, K-number: 20001430.
 *
 * @author Michael Kölling and Josh Murphy
 * @version 2.0
 */
public class Property {
    private final String id;
    private final String hostID;
    private final String hostName;
    private final String neighbourhood;
    private final double latitude;
    private final double longitude;
    private final String roomType;
    private final int price;
    private final int minimumNights;
    private boolean isFavourite;

    /**
     * Create a new property with specified initial values.
     *
     * @param id            unique identification for the property
     * @param hostID        unique identification for the host
     * @param hostName      name of the host
     * @param neighbourhood general area of the property
     * @param latitude      geographical y coordinate of property
     * @param longitude     geographical x coordinate of property
     * @param roomType      indicates the type of room: entire property or a private room
     * @param price         price of staying the minimum nights
     * @param minimumNights required amount of nights in order to book.
     */
    public Property(String id, String hostID, String hostName, String neighbourhood, double 
latitude, double longitude,
                        String roomType, int price, int minimumNights) {
        this.id = id;
        this.hostID = hostID;
        this.hostName = hostName;
        this.neighbourhood = neighbourhood;
        this.latitude = latitude;
        this.longitude = longitude;
        this.roomType = roomType;
        this.price = price;
        this.minimumNights = minimumNights;

        isFavourite = false;
    }

    /**
     * Accessed to retrieve the ID to display on GUI.
     *
     * @return the ID of this property.
     */
    public String getID() {
        return id;
    }

    /**
     * Accessed to retrieve the host's ID to display on GUI.
     *
     * @return the hostID of this property.
     */
    public String getHostID() {
        return hostID;
    }

    /**
     * Accessed to retrieve the latitude to display on GUI.
     *
     * @return the latitude of this property.
     */
    public double getLatitude() {
        return latitude;
    }

    /**
     * Accessed to retrieve the longitude to display on GUI.
     *
     * @return the longitude of this property.
     */
    public double getLongitude() {
        return longitude;
    }

    /**
     * Accessed to retrieve the price to display on GUI.
     *
     * @return the price of this property.
     */
    public int getPrice() {
        return price;
    }

    /**
     * Checks if the property is a favourite and displays on GUI.
     *
     * @return true if this property is currently marked as a favourite, false otherwise.
     */
    public boolean isFavourite() {
        return isFavourite;
    }

    /**
     * Accessed to retrieve the price to display on GUI.
     *
     * @return the host name of this property.
     */
    public String getHostName() {
        return hostName;
    }

    /**
     * Accessed to retrieve the general area to display on GUI.
     *
     * @return the neighbourhood of this property.
     */
    public String getNeighbourhood() {
        return neighbourhood;
    }

    /**
     * Accessed to retrieve the room type to display on GUI.
     *
     * @return the room type of this property.
     */
    public String getRoomType() {
        return roomType;
    }

    /**
     * Accessed to retrieve the minimum number of nights to book and displays on GUI.
     *
     * @return the minimum number of nights this property can be booked for.
     */
    public String getMinNights() {
        return "" + minimumNights;
    }

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

}

```

PropertyViewer.java

```java
import java.net.URI;
import java.util.ArrayList;

/**
 * This project implements a simple application. Properties from a fixed
 * file can be displayed.
 * <p>
 * Student name: Dylan Barker, K-number: 20001430.
 *
 * @author Michael Kölling and Josh Murphy
 * @version 2.0
 */
public class PropertyViewer {
    private final PropertyViewerGUI gui;           // the Graphical User Interface
    private final Portfolio portfolio;
    private final int portfolioSize;               // size of the property
    private final ArrayList<Integer> propertyPrices;
    private Property currentProperty;
    private int currentPropertyIndex;
    private int views;

    /**
     * Creates a PropertyViewer and displays the first property in its GUI.
     */
    public PropertyViewer() {
        gui = new PropertyViewerGUI(this);
        portfolio = new Portfolio();
        portfolioSize = portfolio.numberOfProperties();

        // collects property prices for the challenge method
        propertyPrices = new ArrayList<>();

        currentPropertyIndex = -1;
        this.nextProperty();
    }

    /**
     * Checks if next property index is within bounds and displays in its GUI.
     */
    public void nextProperty() {
        if (++currentPropertyIndex > portfolioSize - 1)
            currentPropertyIndex = 0;
        updateProperty();
    }

    /**
     * Checks if previous property index is within bounds and displays in its GUI.
     */
    public void previousProperty() {
        if (--currentPropertyIndex < 0)
            currentPropertyIndex = portfolioSize - 1;
        updateProperty();
    }

    /**
     * Updates GUI with the current property and its details.
     */
    private void updateProperty() {
        currentProperty = portfolio.getProperty(currentPropertyIndex);

        // updates views and prices for challenge methods
        views++;
        propertyPrices.add(currentProperty.getPrice());

        gui.showProperty(currentProperty);
        gui.showID(currentProperty);
        gui.showFavourite(currentProperty);
    }

    /**
     * Toggles the property favourite status and shows it in its GUI.
     */
    public void toggleFavourite() {
        currentProperty.toggleFavourite();
        gui.showFavourite(currentProperty);
    }


    //----- 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.
     *
     * @throws Exception if the default browser is not found or inaccessible.
     */
    public void viewMap() throws Exception {

        double latitude = currentProperty.getLatitude();
        double longitude = currentProperty.getLongitude();

        URI uri = new URI("https://www.google.com/maps/place/" + latitude + "," + longitude);
        java.awt.Desktop.getDesktop().browse(uri);
    }

    /**
     * Accessed to display how many property views there are which is displayed in a new
window.
     *
     * @return how many properties are viewed.
     */
    public int getNumberOfPropertiesViewed() {
        return views;
    }


    /**
     * Calculates the average price of the properties viewed so far which is displayed in new
window.
     *
     * @return the mean price.
     */
    public int averagePropertyPrice() {
        int sum = 0;
        for (int price : propertyPrices)
            sum += price;
        return sum / views;
    }

}

```

PropertyViewerGUI.java

```java
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.border.EtchedBorder;
import java.awt.*;

/**
 * PropertyViewerGUI provides the GUI for the project. It displays the property
 * and strings, and it listens to button clicks.
 * <p>
 * Student name: Dylan Barker, K-number: 20001430.
 *
 * @author Michael Kölling and Josh Murphy
 * @version 2.0
 */
public class PropertyViewerGUI {
    private final PropertyViewer viewer;
    // 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;

    /**
     * Create a PropertyViewer and display its GUI on screen.
     *
     * @param viewer The base property viewer for the application.
     */
    public PropertyViewerGUI(PropertyViewer viewer) {
        this.viewer = viewer;
        makeFrame();
        this.setPropertyViewSize(400, 250);
    }


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

    /**
     * Displays a given property.
     *
     * @param property The property to be displayed.
     */
    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());
        this.showID(property);
        this.showFavourite(property);
    }

    /**
     * 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.
     *
     * @param width  The selected width for the property panel
     * @param height The selected height for the property panel.
     */
    public void setPropertyViewSize(int width, int height) {
        propertyPanel.setPreferredSize(new Dimension(width, height));
        frame.pack();
    }

    /**
     * Show a message in the status bar at the bottom of the screen.
     *
     * @param property The currently displayed property.
     */
    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.
     *
     * @param property The currently displayed property.
     */
    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();
    }


    /**
     * Creates a new window, displaying the current property statistics.
     */
    private void viewStatistics() {
        JFrame window = new JFrame("Statistics Window");

        // JLabel with html to center and put spaces between statements
        window.add(new JLabel("<html><center>Properties viewed: " + viewer.getNumberOfPropertiesViewed() +
                    "<br/><br/>" + "Average price of properties viewed: £" + viewer.averagePropertyPrice()));

        window.pack();
        window.setLocationRelativeTo(null);
        window.setVisible(true);
    }

    // ---- 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(e -> nextButton());
        toolbar.add(nextButton);

        JButton previousButton = new JButton("Previous");
        previousButton.addActionListener(e -> previousButton());
        toolbar.add(previousButton);

        JButton mapButton = new JButton("View Property on Map");
        mapButton.addActionListener(e -> viewOnMapsButton());
        toolbar.add(mapButton);

        JButton favouriteButton = new JButton("Toggle Favourite");
        favouriteButton.addActionListener(e -> toggleFavouriteButton());
        toolbar.add(favouriteButton);

        JButton statisticsButton = new JButton("View Statistics");
        statisticsButton.addActionListener(e -> viewStatistics());
        toolbar.add(statisticsButton);


        // 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);
    }

}

```