Stock.java

public class Stock implements	// Defines the class named "Stock" and
Comparable <stock>{</stock>	implements the Comparable interface with the
	generic type parameter of Stock
Constructor	Diff parameter, initialize variable of Stock class
Getter	Retrieve
Setter	Set
updateStock	// Method to update the stock by subtracting
	the quantity of the order from the volume of
	the stock
compareTo	// Method to compare the volume of this stock
	with another stock for sorting purposes
toString	// Method to provide a string representation of
	the Stock object

StockUtils.java

```
public class StockUtils {
                                                                                    These lines define the 'StockUtils' class
     private static final String API_KEY1 = "ZKOGFWAKA82MSEC9";
                                                                                    and declare two 'API KEY' constants
     private static final String API_KEY2 = "FEVX5MOLAQB0H6NH";
                                                                                    for accessing stock data.
                                                                                    The second API key is specifically for
                                                                                    educational purposes with a higher
                                                                                    request limit.
                                                                                    This method ('searchStock') takes an
   input = input.trim():
   try {
                                                                                    'input' parameter,
      String[] symbol = getStockSymbol(input.toUpperCase(), API_KEY2);
     if (symbol == null) {
                                                                                    which represents the stock to search
         System.out.println("Stock not found.");
                                                                                    for.
        return null:
                                                                                    It trims the input, gets the stock
     JSONObject stockData = getStockData(symbol[0], API KEY2);
                                                                                    symbol using `getStockSymbol`
         System.out.println("Error retrieving stock data. Please try again later. (
                                                                                    method,
         return null:
                                                                                    retrieves stock data using
     return createStockInfo(stockData, symbol[1]);
                                                                                    `getStockData` method,
                                                                                    and creates a 'Stock' object using
  } catch (IOException e) {
      System.out.println("Error: " + e.getMessage());
                                                                                    `createStockInfo` method.
                                                                                    It returns the created 'Stock' object if
   return null:
                                                                                    successful;
                                                                                    otherwise, it prints error messages and
                                                                                    returns null.
public static String[] getStockSymbol(String input, String apikey) throws IOException {
   String apiUrl = "https://www.alphavantage.co/query?function=SYMBOL_SEARCH&Reywords=" + input +
                                                                                    This method ('getStockSymbol') takes
        "&apikey=" + apikey;
                                                                                    an 'input' parameter (stock symbol or
  URL url = new URL(apiUrl);
                                                                                    name) and an 'apikey' parameter.
  HttpURLConnection connection = (HttpURLConnection) url.openConnection();
  connection setRequestMethod("GET")
                                                                                    It constructs the API URL,
  BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
                                                                                    makes an HTTP GET request to the
  String line;
                                                                                    Alpha Vantage API,
  while ((line = reader.readLine()) != null) {
                                                                                    retrieves the response,
  reader close().
                                                                                    and parses the JSON response to
  JSONObject jsonResponse = new JSONObject(response.toString());
                                                                                    extract the stock symbol and name.
  if (jsonResponse.has("bestMatches")) {
     JSONObject bestMatch = jsonResponse.getJSONArray("bestMatches").optJSONObject(0);
                                                                                    It returns an array of strings containing
     if (bestMatch != null) {
   String symbol = bestMatch.getString("1. symbol");
                                                                                    the symbol and name if a match is
       String name = bestMatch.getString("2. name");
            n new String[]{symbol, name};
                                                                                    found; otherwise,
    }
                                                                                    it returns null.
  return null;
     static JSONObject getStockData(String symbol, String apiKey) throws IOException {
                                                                                    This method ('getStockData') takes a
  String apiUrl = "https://www.alphavantage.co/query?function=GLOBAL_QUOTE&symbol=" + symbol +
        "&apikey=" + apiKey;
                                                                                     `symbol` (stock symbol) and `apiKey` as
  URL url = new URL(apiUrl):
                                                                                    parameters.
  HttpURLConnection connection = (HttpURLConnection) url.openConnection();
  connection.setRequestMethod("GET");
                                                                                    It constructs the API URL for retrieving
  BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
                                                                                    global quote data,
  StringBuilder response = new StringBuilder();
                                                                                    makes an HTTP GET request to the
  while ((line = reader.readLine()) != null) {
     response.append(line);
                                                                                    Alpha Vantage API,
   reader.close();
                                                                                    retrieves the response,
  JSONObject jsonResponse = new JSONObject(response.toString());
                                                                                    and parses the JSON response to
     return isonResponse.getJSONObject("Global Ouote"):
                                                                                    extract the global quote information.
                                                                                    It returns a 'JSONObject' containing
  return null:
                                                                                    the global quote data if available;
                                                                                    otherwise, it returns null.
```

```
vate static Stock createStockInfo(JSONObject stockData, String name) {
                                                                                          This method ('createStockInfo') takes a
  String symbol = stockData.getString("01. symbol");
   double price = stockData.getDouble("05. price");
                                                                                          'stockData' 'JSONObject' and a 'name'
   long volume = stockData.getLong("06. volume");
   double change = stockData.getDouble("09. change");
  String changePercentString = stockData.getString("10. change percent");
                                                                                          It extracts specific data fields (symbol,
   double changePercent = Double.parseDouble(changePercentString.replace("%", ""));
                                                                                          price, volume, change, change
  return new Stock(symbol, name, price, volume, change, changePercent);
                                                                                          percentage) from the 'stockData' JSON
                                                                                          and creates a new 'Stock' object with
                                                                                          the extracted information.
                                                                                          It returns the created 'Stock' object.
                                                                                          This method ('getStockCurrentPrice')
     String apiUrl = "https://www.alphavantage.co/query?function=GLOBAL OUOTE&symbol=" + stockSymbol +
                                                                                          takes a 'stockSymbol' parameter and
     URL url = new URL(apiUrl);
                                                                                          retrieves the current price of the
     HttpURLConnection connection = (HttpURLConnection) url.openConnection();
     connection.setRequestMethod("GET"):
                                                                                          specified stock.
     BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
StringBuilder response = new StringBuilder();
                                                                                          It constructs the API URL for retrieving
                                                                                          global quote data,
     while ((line = reader.readLine()) != null) {
       response.append(line);
                                                                                          makes an HTTP GET request to the
     reader.close();
                                                                                          Alpha Vantage API,
     JSONObject jsonResponse = new JSONObject(response.toString());
     if (jsonResponse.has("Global Quote")) {
    JSONObject stockData = jsonResponse.get35
    return stockData.getDouble("05. price");
                                                                                          retrieves the response,
                                                                                          and parses the JSON response to
  } catch (MalformedURLException e){
    System.out.println("Invalid URL: " + e.getMessage()):
                                                                                          extract the current stock price.
 } catch (IOException e) {
   System.out.println("Error retrieving stock price: " + e.getMessage());
                                                                                          It returns the current stock price as a
                                                                                          double value.
nublic static List<Stock> scraneStock()
                                                                                          This method (`scrapeStock`) scrapes
  String url = "https://finance.yahoo.com/most-active?offset=0&count=50":
                                                                                          the stock data from the Yahoo Finance
                                                                                          most active stocks page.
      // Send a GET request to the Yahoo Finance trending tickers page
     Document document = Jsoup.connect(url).get();
                                                                                          It sends a GET request to the specified
     // Find the table containing the stock data
     Element table = document.select("table").first();
                                                                                          URL,
                                                                                          retrieves the HTML document using
      // Extract the stock data from the table
                                                                                          Jsoup,
     for (int i = 1; i < rows.size(); i++) {
        Element row = rows.get(i);
Elements columns = row.select("td");
                                                                                          and then extracts the stock data from
        String symbol = columns.get(0).text().trim();
String name = columns.get(1).text().trim();
                                                                                          the HTML table.
        String priceSt = columns.get(2).text().trim().replace(",", "");
         double price = parseDoubleValue(priceSt);
                                                                                          It iterates over the table rows, extracts
        String changeSt = columns.get(3).text().trim().replace("+", "");
                                                                                          the symbol, name, price, change,
        double change = parseDoubleValue(changeSt);
        String changePercentageSt = columns.get(4).text().trim().replace("%", "").replace("+", "");
double changePercentage = parseDoubleValue(changePercentageSt);
                                                                                          change percentage,
        String volumeSt = columns.get(5).text().trim().replace(".", "").replace(",", "").replace("M", "0000");
long volume = parselongValue(volumeSt);
                                                                                          and volume values from the table cells.
                                                                                          creates a 'Stock' object with the
        Stock stock = new Stock(symbol, name, price, volume, change, changePercentage);
        stockData.add(stock);
                                                                                          extracted data,
  } catch (IOException e) {
                                                                                          and adds it to the 'stockData' list.
     e.printStackTrace()
                                                                                          Finally, it returns the list of 'Stock'
                                                                                          objects.
  return stockData:
private static double parseDoubleValue(String value) {
                                                                                          These are helper methods
     if (value.equalsIgnoreCase("N/A")) {
                                                                                          (`parseDoubleValue` and
         return 0.0; // Set a default value or handle it as needed
                                                                                          `parseLongValue`) used to parse string
     return Double.parseDouble(value);
                                                                                          values
}
                                                                                          to their respective numerical types.
private static long parseLongValue(String value) {
                                                                                          They handle cases where the value is
    if (value.equalsIgnoreCase("N/A")) {
                                                                                          "N/A" by returning a default value
         return OL; // Set a default value or handle it as needed
                                                                                          (0.0 for 'double' and 0L for 'long')
     return Long.parseLong(value);
                                                                                          or you can handle it as needed in your
                                                                                          application.
```

HomeController.java

and in the state of the state o	The 'Harra Controller' along in
public class HomeController implements Initializable {	The `HomeController` class is a
	controller for the home view in a
	JavaFX application. It implements
	the `Initializable` interface, which
	allows initializing the controller and
	its components when the
	associated FXML file is loaded.
private User user;	This code defines a `User` object
<pre>public void setUser(User user){</pre>	and a setter method `setUser()` to
this.user = user;	set the currently logged-in user.
}	l l l l l l l l l l l l l l l l l l l
@FXML	These lines declare several labels
	that are defined in the associated
	FXML file. The `@FXML` annotation
	is used to inject the labels defined
	in the FXML into the controller.
private Timeline timeline;	This declares a `Timeline` object
private rimeline timeline;	
	that will be used to update the time
	and market status labels
private TableCalumn(Stack Long) calumn valumn	periodically.
private TableColumn <stock, long=""> column_volume;</stock,>	These lines declare a `TableView`
List <stock> popularStocks = StockUtils.scrapeStock();</stock>	and several `TableColumn` objects
	used for displaying stock data. The
	`@FXML` annotation is used to
	inject them from the FXML file. The
	`popularStocks` list is populated
	with stock data using the
	`scrapeStock()` method from the
	`StockUtils` class.
@Override	This is the initialization method that
<pre>public void initialize(URL url, ResourceBundle resourceBundle) {</pre>	is called when the associated FXML
//	file is loaded. It initializes the
}	
	controller and sets up the initial
timeline new Timeline/new MayEmpmo(inveformin) Downties and (4)	state of the components.
<pre>timeline = new Timeline(new KeyFrame(javafx.util.Duration.seconds(1), event -> { updateDateTime();</pre>	This code creates a 'Timeline' that
updateMarketStatus();	updates the time and market status
}));	labels every second. It uses a
<pre>timeline.setCycleCount(Timeline.INDEFINITE); // Run indefinitely timeline.play();</pre>	`KeyFrame` with a duration of one
cameance, pady ()	second and defines an event
	handler to call the
	`updateDateTime()` and
	`updateMarketStatus()` methods.
	,

TradeController.java

public class TradeController implements Initializable {	This line declares the class `TradeController` and specifies that it implements the `Initializable` interface.
<pre>private User user; public void setUser(User user){ this.user = user; this.user.displayUserPortfolio(); System.out.println("load user successful"); label_balance.setText("\$" + String.format("%.2f",user.getBalance())); label_profit.setText("\$" + String.format("%.2f",user.getPnL())); label_loss.setText("" + String.format("%.2f",user.getPoint())); }</pre>	These lines declare a private instance variable `user` of type `User` and define a setter method `setUser()` to set the user object. Inside the `setUser()` method, the `user` instance variable is assigned the provided `User` object, and some user-related information is displayed in labels.
<pre>private TradingFunctions tf; public void setTf(TradingFunctions tf){ this.tf = tf; }</pre>	These lines declare a private instance variable `tf` of type `TradingFunctions` and define a setter method `setTf()` to set the `TradingFunctions` object.
private Timeline timeline;	This line declares a private instance variable `timeline` of type `Timeline` for updating the time and market status labels.
btn_setting → label_companyName	These lines annotate and declare various JavaFX UI components, such as buttons and labels, with `@FXML` annotations for use in the controller.
label_stockCode → btn_search	These lines declare additional UI components related to stock information.
<pre>@FXML private ChoiceBox<string> cBox_action; private String[] action = {"BUY", "SELL"};</string></pre>	These lines declare a choice box UI component and a string array containing the available actions (buy and sell).
tf_symbol → btn_executeTrade	These lines declare UI components related to trade functionality, such as text fields and buttons.