Author: Mikey Joyce

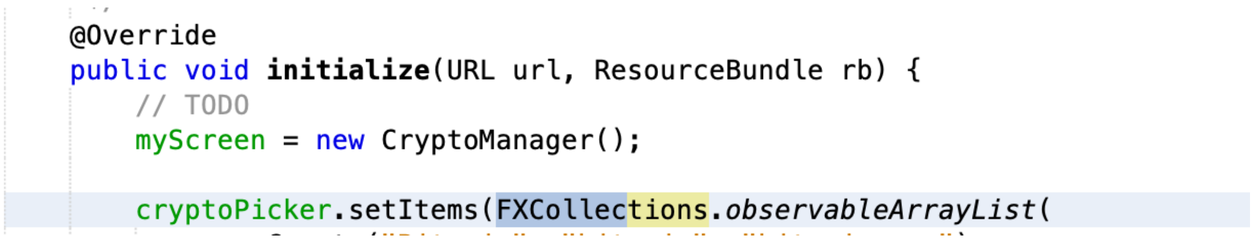
Pawprint: Mpjyky

Project Title: CryptoWatch

CS 3330 Final Project:

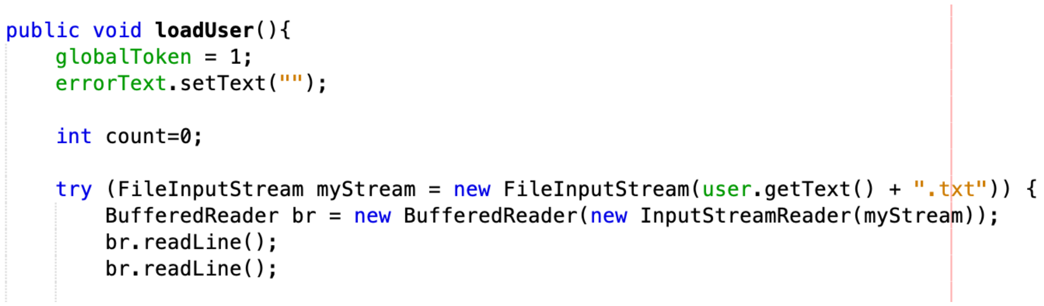
1. Requirement of Classes:
    a. There are 9 classes in this project titled:
        i. AboutController.java
        ii. Crypto.java
        iii. CryptoManager.java
        iv. HomePageController.java
        v. Login.java
        vi. LoginController.java
        vii. LoginModel.java
        viii. MpjykyCryptoWatch.java
        ix. Switchable.java (REFERENCED FROM PROF WERGELES)
2. Requirement of Subclasses:
    a. This requirement was not met
3. Requirement of 1 abstract class:
    a. There are 2 abstract classes in this project titled:
        i. Login.java
        ii. Switchable.java (REFERENCED FROM PROF WERGELES)

4. Requirement of 1 interface:
    a. There is 1 interface titled:
        i. Manager.java
5. Requirement of 1 collections class:
    a. There is 1 instance of a collection class its location is:
        i. Within the file HomePageController.java it is the second line of code in the initialize method



```
@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
    myScreen = new CryptoManager();

    cryptoPicker.setItems(FXCollections.observableArrayList(
```

6. Requirement of Exception Handling
    a. There are many instances of exception handling, the following is one of them:
        i. In the file HomePageController.java in the loadUser() function there is a try/catch block that helps to load the user from a file.



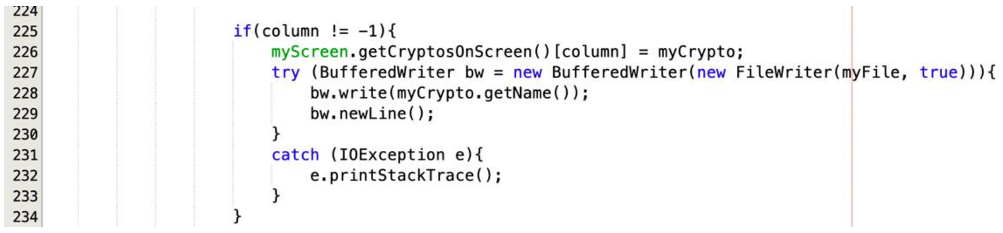
```
public void loadUser(){
    globalToken = 1;
    errorText.setText("");

    int count=0;

    try (FileInputStream myStream = new FileInputStream(user.getText() + ".txt")) {
        BufferedReader br = new BufferedReader(new InputStreamReader(myStream));
        br.readLine();
        br.readLine();
```

7. Requirement of Models in MVC:
    a. There are 3 models within this project they are the files titled:
        i. Crypto.java

        ii. CryptoManager.java
        iii. LoginModel.java

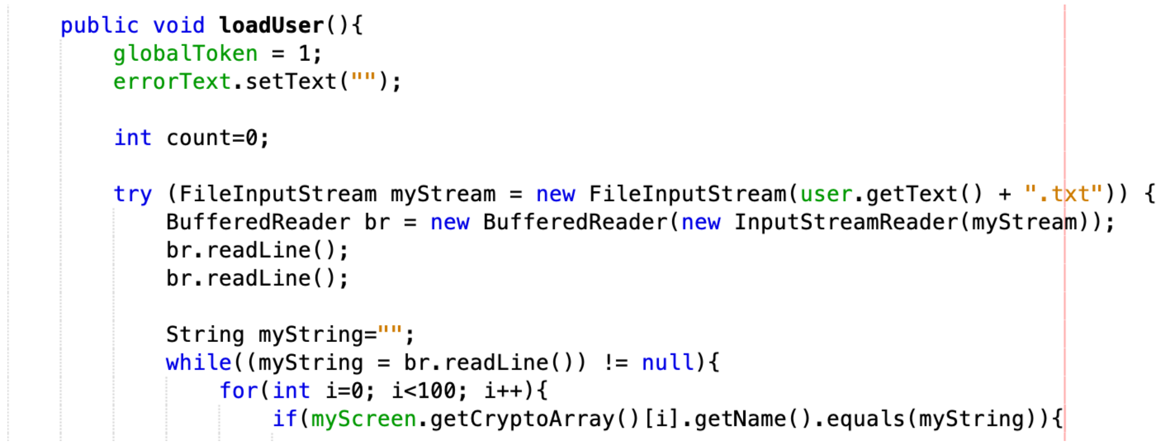
8. Requirement of multiple scenes:
    a. There are 3 scenes in this project titled:
        i. About.fxml
        ii. HomePage.fxml
        iii. Login.fxml
    b. HomePage.fxml is dynamic. You can add and delete cryptos to the screen. If a crypto is on the screen it will update the price every 1 minute (CoinGecko API doesn't update the prices very often)
9. Requirement of About information:
    a. There is an about scene titled:
        i. About.fxml
    b. To access this scene while using the app you must click the "CryptoWatch" logo in the top left corner of the screen. To get back to the cryptos page click the same logo at the top of the screen.
10. Requirement of Saving and Loading Data:
    a. This project saves data to a file based on the username of the user (username.txt). Once data has been saved to the file you can exit the app and reopen it to experience the load feature. To load the data of your user you must click on the username that is displayed on the top left part of the screen on the HomePage.fxml.

b. The write feature is contained within the addDelCrypto() function of the HomePageController.java file:



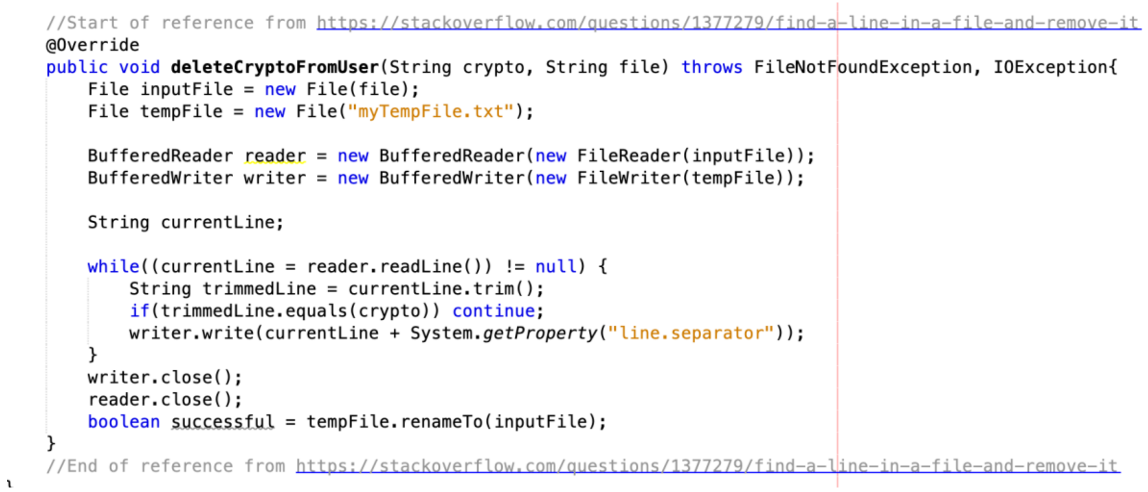
```java
if(column != -1){
    myScreen.getCryptosOnScreen()[column] = myCrypto;
    try (BufferedWriter bw = new BufferedWriter(new FileWriter(myFile, true))){
        bw.write(myCrypto.getName());
        bw.newLine();
    }
    catch (IOException e){
        e.printStackTrace();
    }
}
```

c. The load feature is contained within the loadUser() function of the HomePageController.java file:



```java
public void loadUser(){
    globalToken = 1;
    errorText.setText("");

    int count=0;

    try (FileInputStream myStream = new FileInputStream(user.getText() + ".txt")) {
        BufferedReader br = new BufferedReader(new InputStreamReader(myStream));
        br.readLine();
        br.readLine();

        String myString="";
        while((myString = br.readLine()) != null){
            for(int i=0; i<100; i++){
                if(myScreen.getCryptoArray()[i].getName().equals(myString)){
```

d. The delete item from file feature is contained within the deleteCryptoFromUser() function of the CryptoManager.java file:

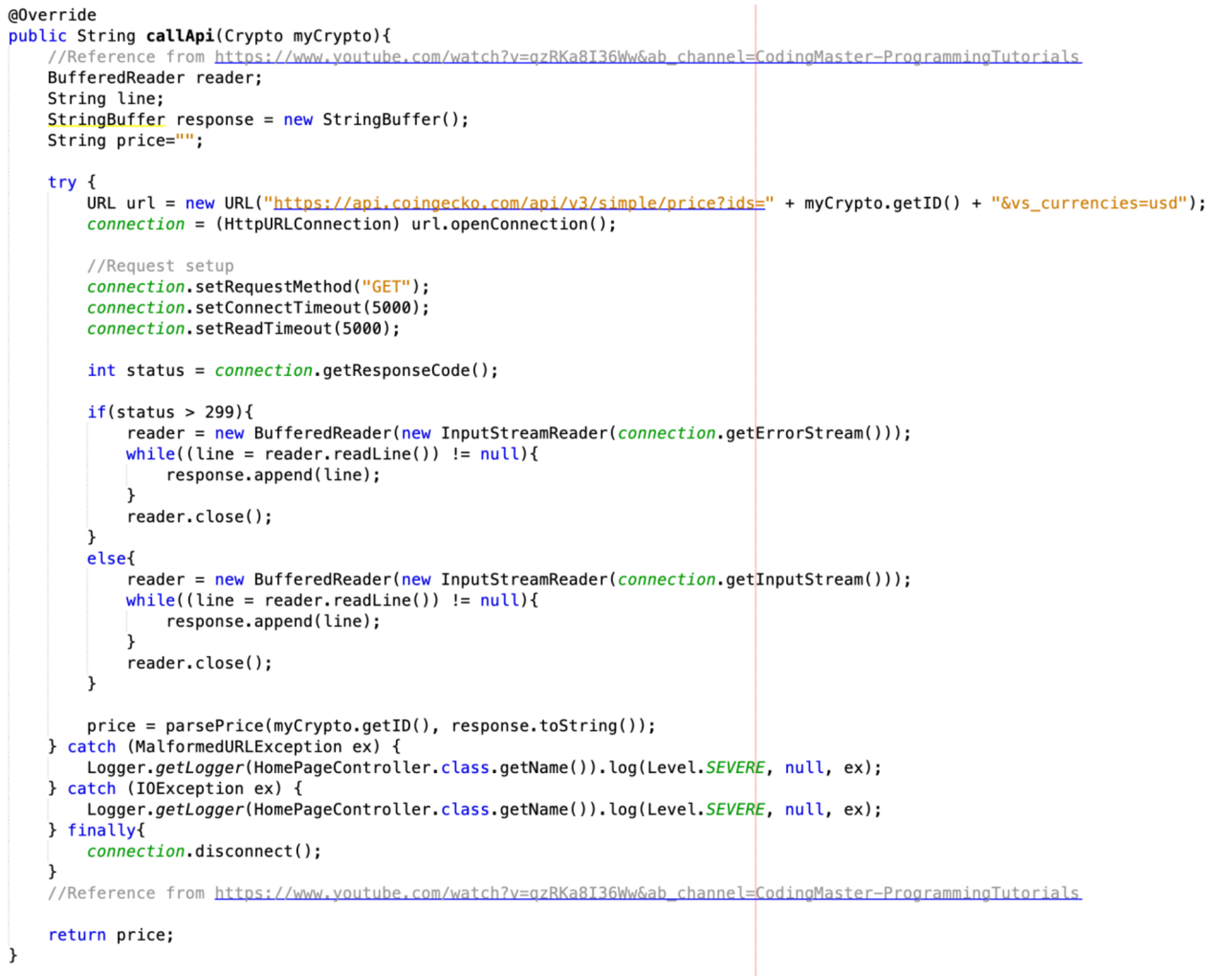


```java
//Start of reference from https://stackoverflow.com/questions/1377279/find-a-line-in-a-file-and-remove-it
@Override
public void deleteCryptoFromUser(String crypto, String file) throws FileNotFoundException, IOException{
    File inputFile = new File(file);
    File tempFile = new File("myTempFile.txt");

    BufferedReader reader = new BufferedReader(new FileReader(inputFile));
    BufferedWriter writer = new BufferedWriter(new FileWriter(tempFile));

    String currentLine;

    while((currentLine = reader.readLine()) != null) {
        String trimmedLine = currentLine.trim();
        if(trimmedLine.equals(crypto)) continue;
        writer.write(currentLine + System.getProperty("line.separator"));
    }
    writer.close();
    reader.close();
    boolean successful = tempFile.renameTo(inputFile);
}
//End of reference from https://stackoverflow.com/questions/1377279/find-a-line-in-a-file-and-remove-it
```

11. Bonus Requirement: Use an API:

a. The CoinGecko api is used to retrieve the current price for the cryptocurrencies.

   It is referenced within the function callApi() of the CryptoManager.java file:



```java
@Override
public String callApi(Crypto myCrypto){
    //Reference from https://www.youtube.com/watch?v=qzRKa8I36Ww&ab_channel=CodingMaster-ProgrammingTutorials
    BufferedReader reader;
    String line;
    StringBuffer response = new StringBuffer();
    String price="";

    try {
        URL url = new URL("https://api.coingecko.com/api/v3/simple/price?ids=" + myCrypto.getID() + "&vs_currencies=usd");
        connection = (HttpURLConnection) url.openConnection();

        //Request setup
        connection.setRequestMethod("GET");
        connection.setConnectTimeout(5000);
        connection.setReadTimeout(5000);

        int status = connection.getResponseCode();

        if(status > 299){
            reader = new BufferedReader(new InputStreamReader(connection.getErrorStream()));
            while((line = reader.readLine()) != null){
                response.append(line);
            }
            reader.close();
        }
        else{
            reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
            while((line = reader.readLine()) != null){
                response.append(line);
            }
            reader.close();
        }

        price = parsePrice(myCrypto.getID(), response.toString());
    } catch (MalformedURLException ex) {
        Logger.getLogger(HomePageController.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(HomePageController.class.getName()).log(Level.SEVERE, null, ex);
    } finally{
        connection.disconnect();
    }
    //Reference from https://www.youtube.com/watch?v=qzRKa8I36Ww&ab_channel=CodingMaster-ProgrammingTutorials

    return price;
}
```

12. Bonus Requirement: Good UI

a. Use the app in my opinion it is a nice-looking app. Originally, I wanted the cryptos that were saved to load right when the HomePage.fxml scene loaded in. But this didn't work because it would read the wrong file. So instead I made it load by clicking the username in the top left part of the screen which is not ideal, but it was what I was working with and I think the rest of the UI turned out extremely clean.