Addressing Code Smells

Nabeela：

Currency Conversion Page:
Problem Summary: The conversion rates are wrong/not accurate due to the hard-coded values.
Refactoring: In the ConvertCurrency.java file, I created a method named ‘fetchcurrencyrates()’ in which real-time conversion rates are called via an external API. In the Curreny.java file, I removed all the hard-coded conversion rates and replaced it with the currency codes. This way the conversion rates are real-time and up to date.

1. Extract interface for conversion logic
Code Smell/Design Issue: Tight Coupling and Single Responsibility Principle Violation
Before: ConvertCurrency class handles many tasks including UI interaction, conversion logic, and fetching conversion rates. This makes the class overly complex making it hard for unit testing and scalability.
Refactoring: The currency converting logic and the rate fetching mechanism can be extracted into a separate interface.
After: the Convertcurrency class now interacts with the separate interface for conversion tasks, simplifying the class and making it easier to test.

2. Code Smell: Repetitive Code and Scalability Concerns
Before: The initialization and management of XYChart.Series for different currencies are handled explicitly in the initialize method, leading to repetitive code.
Refactoring Applied: Implement a Factory Pattern to encapsulate the creation of XYChart.Series instances that initialize and return a new series based on the given currency.
After: Adding or removing currencies and their associated series becomes easier and cleaner, with less repetitive code in the initialize method. This approach enhances the application's scalability and readability.

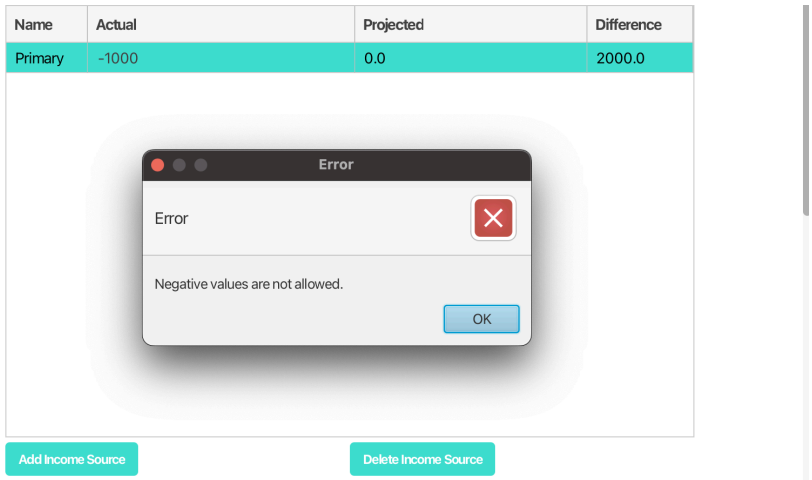
3. Code smell: Primitive obsession.
Before: The initial Currency enum only provided a basic name and conversion rate to Indian Rupees, relying on primitive data types for handling currency attributes.
Refactoring Applied: Implement a method in which currency rates are fetched from an external API in order to not rely on primitive data.
After: After adding a method to fetch real time rates for conversion, the code is much more accurate and clean.

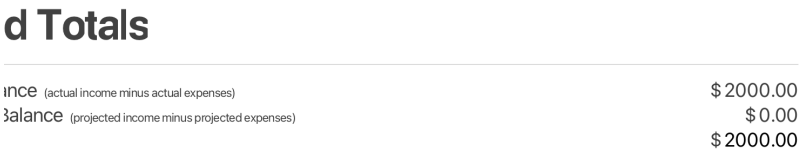
Addressing Code Smells

Ahmed：

1. **Code Smell: Income table bug**

Before: By definition, it shouldn't be possible to have a negative actual but the field allows for the input of a double below 0 acting as an expense.
Refactoring Applied: Implement a specific if statement and error prompt to notify and to control the user input so it acts as a positive integer and functions accordingly
After: After adding the clause the user will now get notified that he is not allowed to use negative values so the functionality of the application acts as expected.

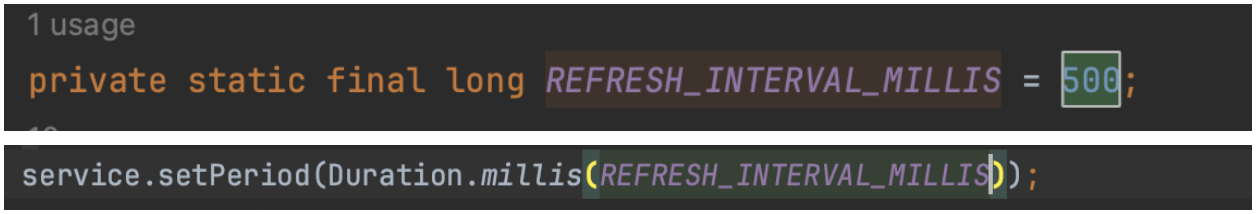




The user can't add a negative income which would previously act as an expense.

2. **Code Smell: Magic numbers**

Before: In this line, 500 is a magic number. It specifies the period for a ScheduledService without context.
Refactoring Applied: The approach was to replace the "magic number" with a named constant that explains its purpose.
After: The user should understand the code better keeping things more clear and concise.



3. **Code Smell: expense table**

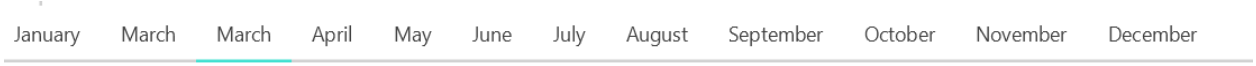
Before: The table displays a specific character n when not occupied by a row.
Refactoring Applied: After a mindless search I have come to the conclusion that the bug is an implemented issue of FXML and not within the code.

Name	Actual Cost	Budget	Difference
Bills	0.0	0.0	0.0
Transportation	0.0	0.0	0.0
Insurance	0.0	0.0	0.0
Food	0.0	0.0	0.0
Self-Care	0.0	0.0	0.0
Entertainment	0.0	0.0	0.0
Loans	0.0	0.0	0.0
Taxes	0.0	0.0	0.0
Investments	0.0	0.0	0.0
	0.0	0.0	0.0
E2E-EDITED	1000.0	3000.0	2000.0
			n

4. **Code Smell: Duplicate month bug)**

Before: The table displayed a duplicate month which was march.

Refactoring Applied: After searching for the issue I have found the issue to be a “rollover”. Calendar Instance was Not Resetting because the day wasn't applied.




```
public void initTabs() {
  months.getTabs().clear();
  var cal = Calendar.getInstance();
  for(int i = 0; i < 12; i++) {
    cal.set(Calendar.MONTH, i);
    Tab newTab = new Tab(new SimpleDateFormat( pattern: "MM").format(cal.getTime()));
    newTab.setClosable(false);
    months.getTabs().add(newTab);
    System.out.println("d"+newTab.getText());
  }
```

Fix:

```
public void initTabs() {
  var cal = Calendar.getInstance();
  for(int i = 0; i < 12; i++) {
    cal.set(Calendar.DAY_OF_MONTH, 1);
    cal.set(Calendar.MONTH, i);
    Tab newTab = new Tab(new SimpleDateFormat( pattern: "MMMM", Locale.CANADA).format(cal.getTime()));
```

Oubai:

Prediction Page:

Problem summary: Whenever an input is invalid, the application only prints an error in the command prompt in PredictionController.java

Resolution

Status: Closed

Resolution: I added an error alert in case the user tries to enter invalid input, now it displays the error to the user in GUI.

Resolved By: Oubai
Resolution Date: 30-03-2024
Comments: None

Portfolio page:

Problem summary: Displays invalid Investment value that is absolute text in portfolioController.java

Resolution

Status: Closed
Resolution: Update to count how much was invested in the time period.

Portfolio page:

Problem summary: addDataToBarChart and addDataToLineChart in portfolioController.java have unnecessary parameters and code that is never used from previous versions.

Resolved By: Oubai
Resolution Date: 30-03-2024
Comments: None

Resolution

Status: Closed
Resolution: Update code to remove unnecessary parameters and unused code.
Resolved By: Oubai
Resolution Date: 30-03-2024
Comments: None

Sarimah

Loan Calculator page

Problem Summary: The actions for the Add and Calculate buttons contain duplicate code for parsing input fields and calculating loan details.

Resolution

Status: Closed
Resolution: I refactored the code to remove the redundant Calculate button. By doing this, adding loans to the table is now only handled by the add button eliminating the duplication and improving the manageability of the code.
Resolved By: Sarimah
Resolution Date: 30-03-2024
Comments: None

Kennie

Savings page

Problem Summary: The code does not print an error message to negative values

Resolution

Status: Closed
Refactored code: I refactored the code to remove direct user input for values. Instead, it now automatically retrieves income and expense data from the database. This approach ensures all calculations within the savings page are based on pre-existing records, enhancing accuracy and preventing issues like negative values from user input. This change streamlines the process, making it more reliable when compared to savings goals.
Resolved By: Kennie
Resolution Date: 30-03-2024
Comments: None