**Name:** Christopher Samuel Tendi

**Major:** Computer Science (Batch 2024)

Java Final Project Report

# **Table of Contents**

1. **Introduction**
   1. Background
   2. Program Function
2. **Solution Implementation**
   1. Parts of the Program
   2. Data Dictionary
3. **Explanation of Code**
   1. HouseholdSpending.java
   2. SpendingDatabase.java
4. **Evidence of Working Program**

# Introduction

**Background**

When the final project was first introduced to us students, to be perfectly honest, I was confused on what to make for it. Thus, I immediately decided to research and look for ideas that might spark my interest and the most common results were databases, which is why I decided to go for that. Afterwards, I tried looking for a problem that could be solved by making a program, then I remembered a relative of mine that struggles to monitor and track their expenses. She mentioned that every month her family seems to have spent way more that what she thought. That gave me an idea. What if I could make a program that could calculate as well as store a household’s/individual’s monthly expenses.

During my research, I also found out that there are 2 ways of making an SQL database, either by hardcoding it using SQLite or by generating a local server through myPHP. I settled for the former due to personal preference after seeing the tutorials for both. I used IntelliJ as my IDE to program this project.

All the code, documentation and video evidences of this project can be found under this GitHub repository: <https://github.com/christophertendi/HouseholdSpending>

**Program Function**

The main purpose of this program is to find the total spending and average spending of a household/individual. This program also allows users to look back on expenditure of previous months and alter the values stored for each month as this program implements a database.

When the user first opens the program, a database will be automatically generated. The user would have to fill in the necessary inputs, which are, a dropdown of the months of a year and the amount of money they spent from the 1st to the 4th week of the selected month. All the inputted data will be saved inside a database, which can be overwritten anytime. Users can also check the data stored in other months for their weekly expenses if they happen to either forget how much they spent, or want to change the inputted data.

The program specifications are as follows:

* **Software and Libraries used:**
  + IDE: IntelliJ by JetBrains
  + SQLite for database
* **Input:**
  + Month
  + Expenses from week 1 to 4
* **Output:**
  + Total Spending
  + Average Spending of the 4 weeks
  + Expenses from week 1 to 4 (when user wants to look at expenditure of other months by pressing the “load” button)

# Solution Implementation

**Parts of the Program**

The program will contain a GUI form, designed by using the IDE. The user has to fill in 5 mandatory inputs which are (in order):

* Month - when the user clicks on the arrow, a dropdown will be shown asking the user to select the intended month
* Week 1 - user has to fill in the expenditure of week 1 of the selected month
* Week 2 - user has to fill in the expenditure of week 2 of the selected month
* Week3 - user has to fill in the expenditure of week 3of the selected month
* Week 4 - user has to fill in the expenditure of week 4 of the selected month

After the user has filled in the necessary inputs and clicked on “Calculate”, the program will calculate the **total spending** during that month along with the **average spending** of that particular month. Additionally, the user is also able to save the inputs of a particular month by clicking on the **Save** button and load it again through the **Load** button.

**Data Dictionary**

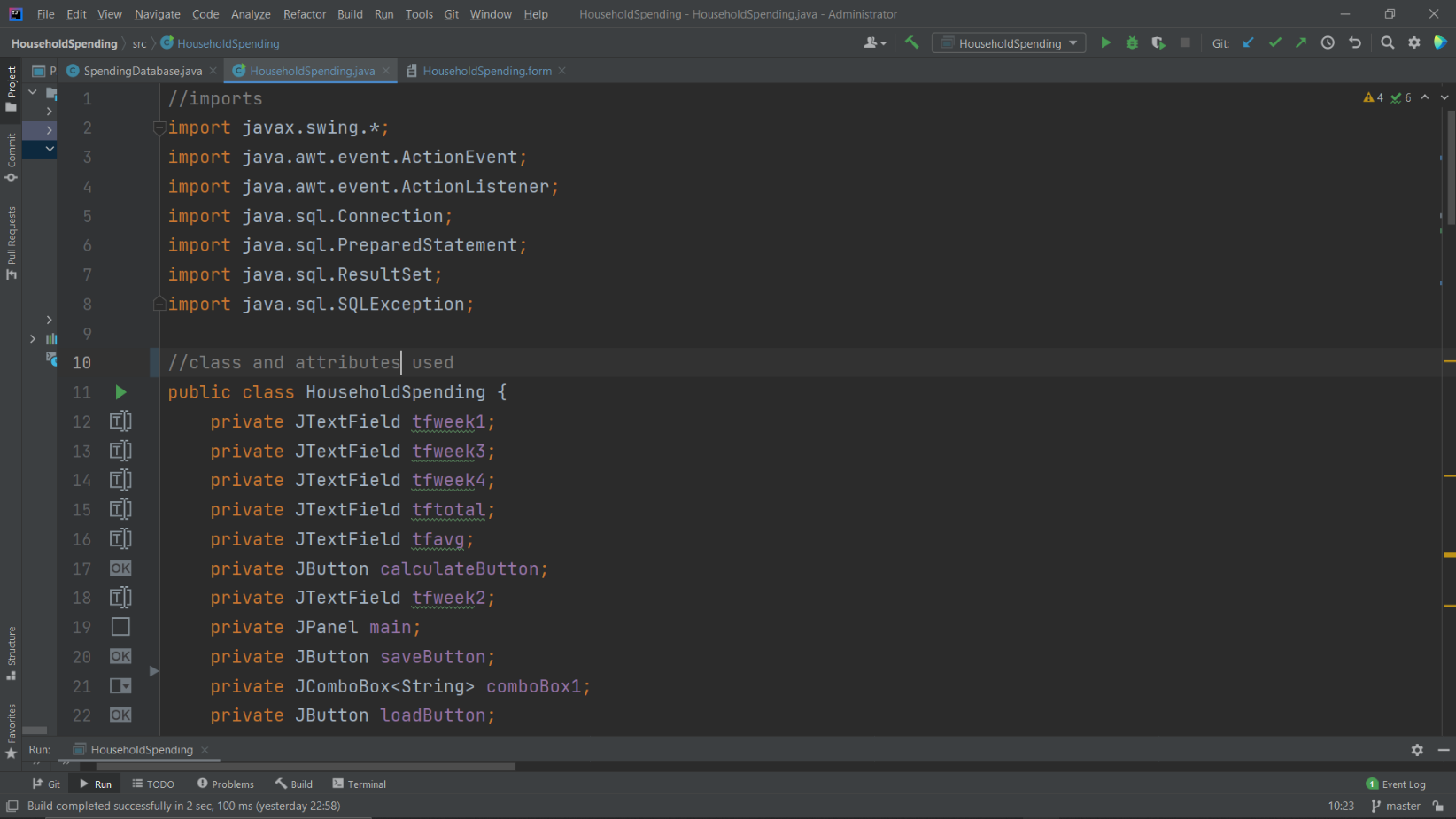
Spending.db

|  |  |  |
| --- | --- | --- |
| Field Name | Data Type | Description |
| month | VARCHAR | Selected month |
| Week1 | DOUBLE | User expenditure on week 1 |
| Week2 | DOUBLE | User expenditure on week 2 |
| Week3 | DOUBLE | User expenditure on week 3 |
| Week4 | DOUBLE | User expenditure on week 4 |

# Explanation of Code

HouseholdSpending.java

Figure 1. Declaration of Imports and Classes



The figure above shows the imports used in this program, which is pretty self-explanatory. The public class HouseholdSpending is the main class in this program, it contains attributes that are used in the GUI form.

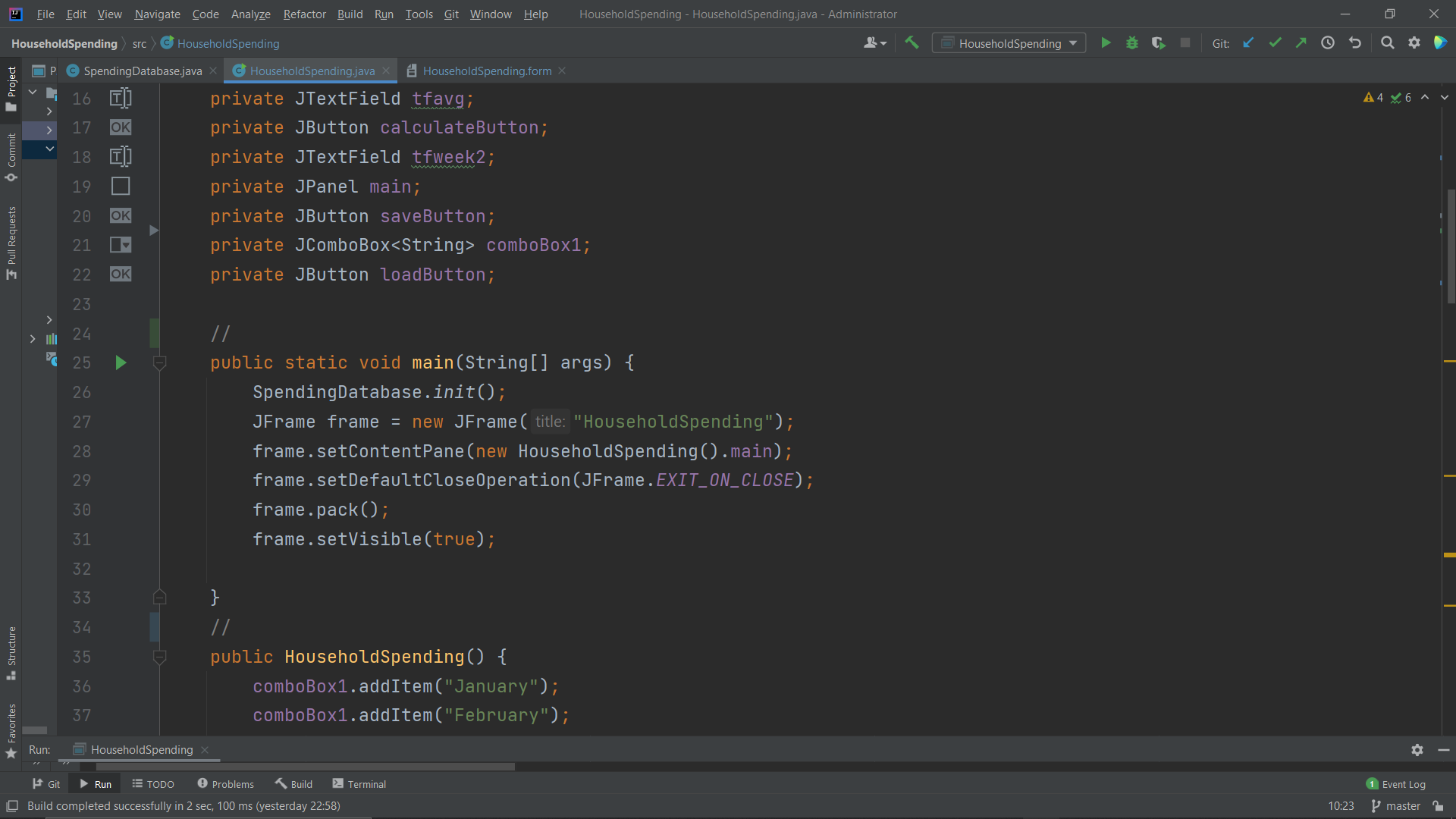
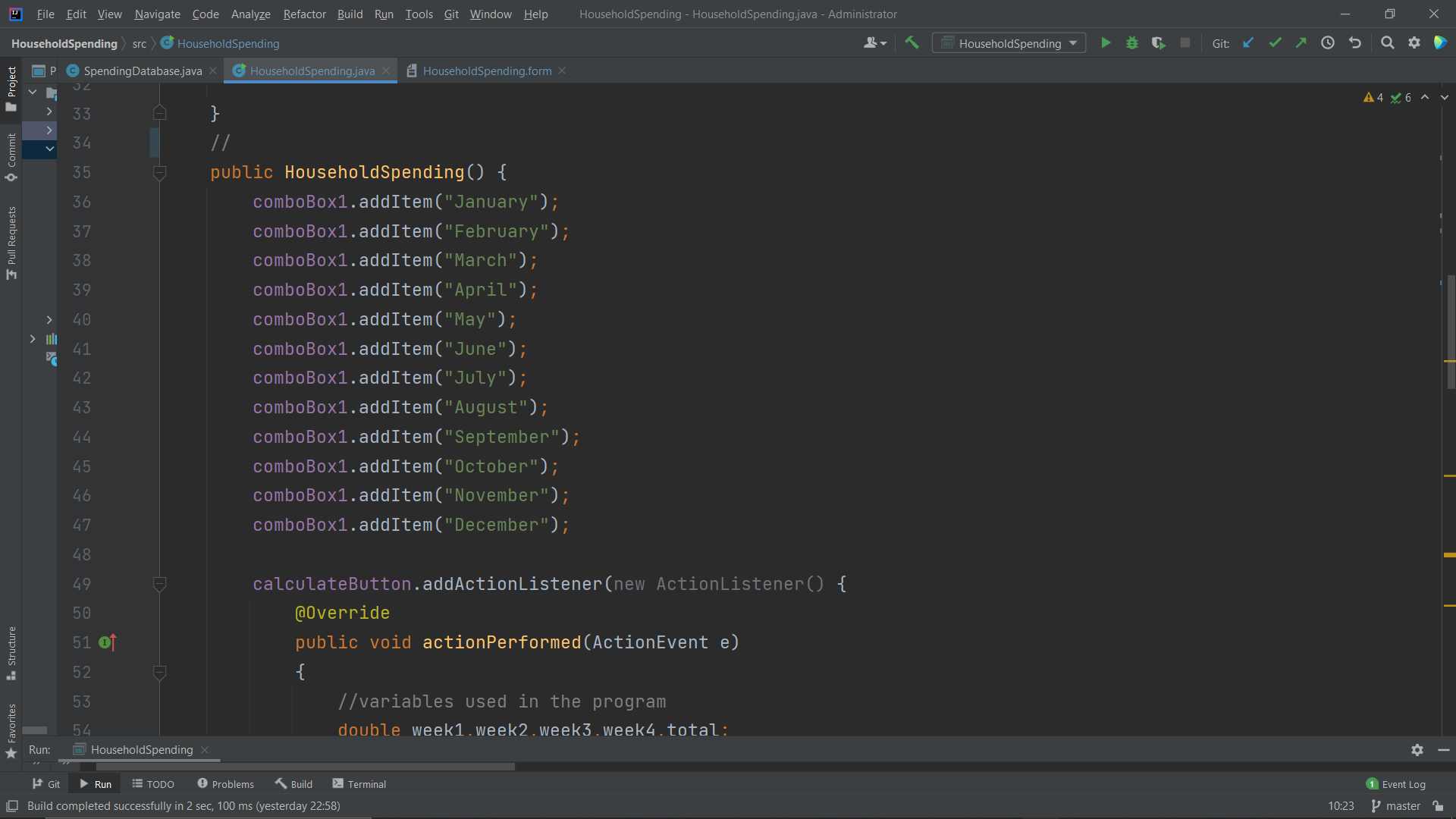
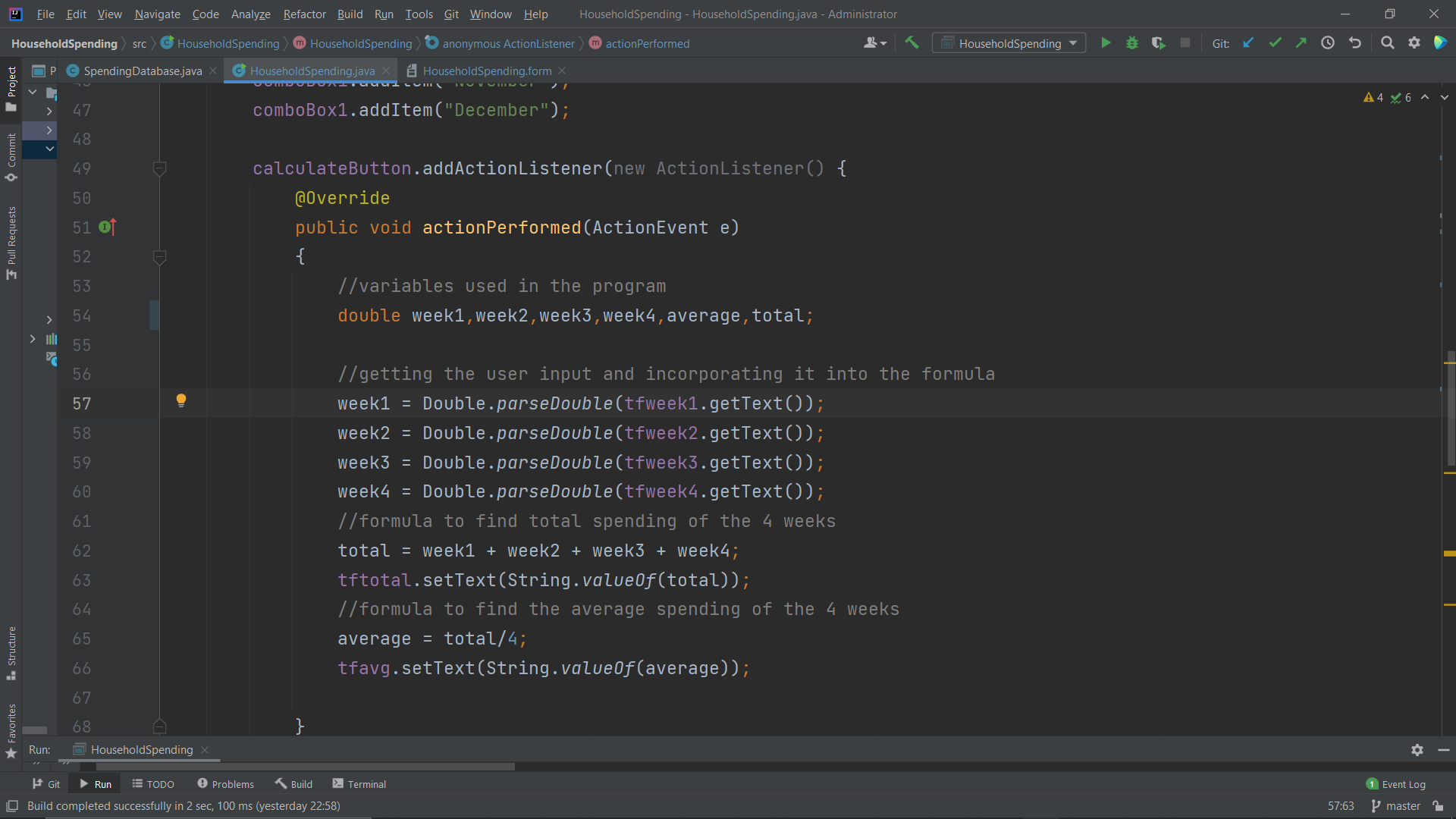


Figure 2. main function to run program

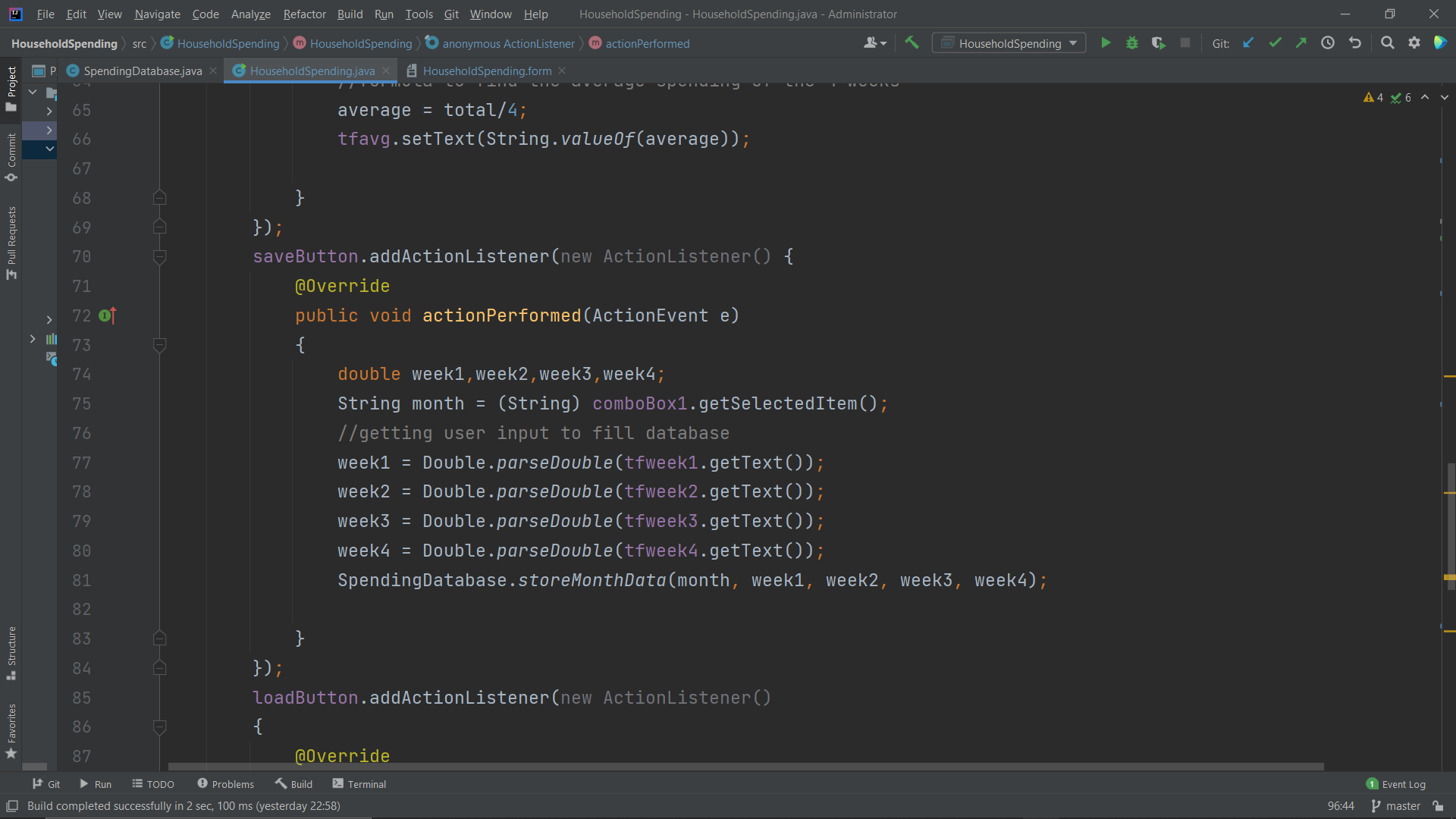
The figure above shows the main method of the program. SpendingDatabase.init() is the function used to initialize or call the functions in the SpendingDatabase.java class. JFrame frame is a variable that calls a window or frame for the program’s GUI.



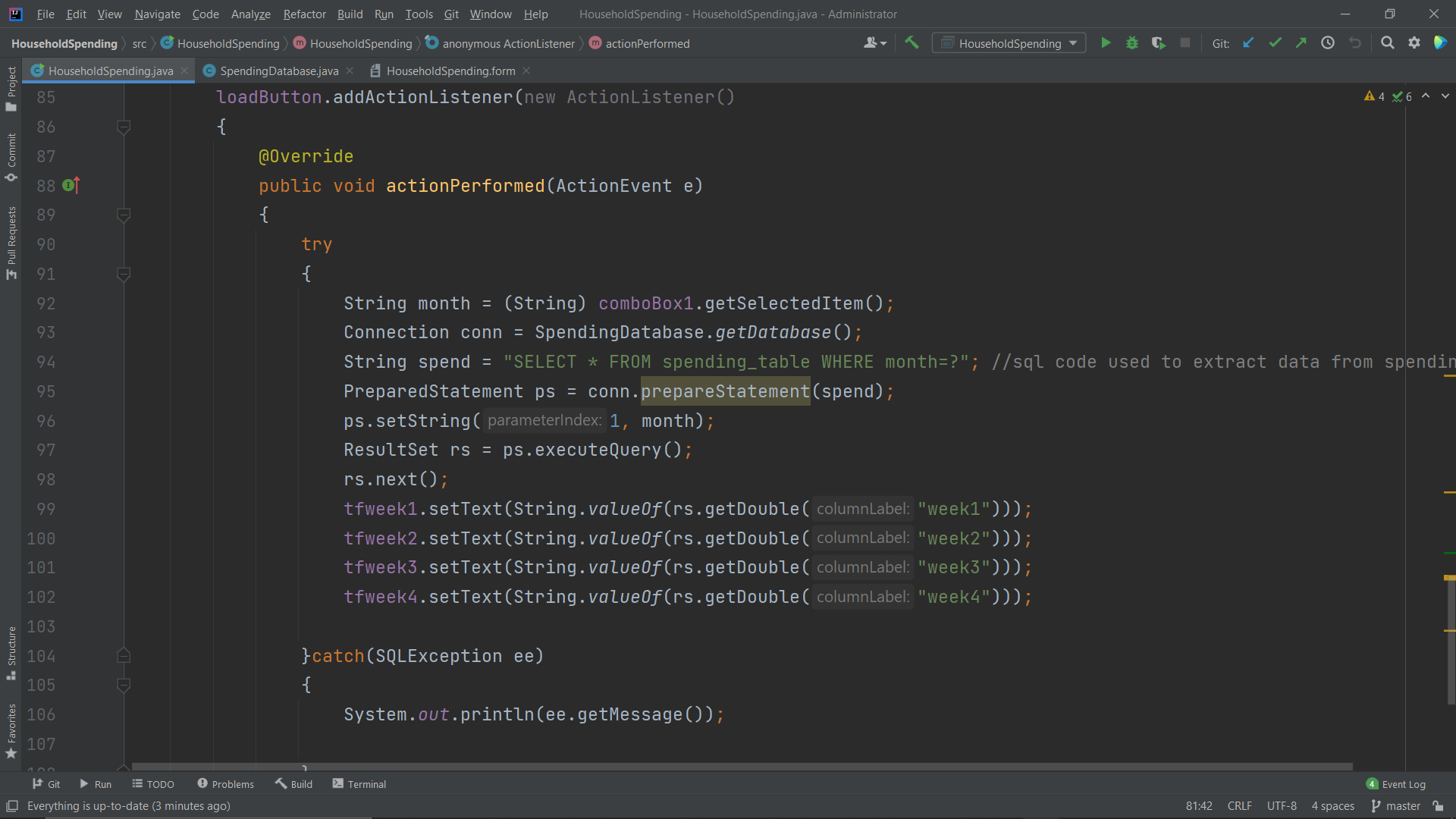
The figure above shows the start of the HouseholdSpending method. comboBox1 is essentially the dropdown box that contains the months of a year from which the user could choose from. comboBox1.addItem adds the entered string, for example January, into the dropdown.



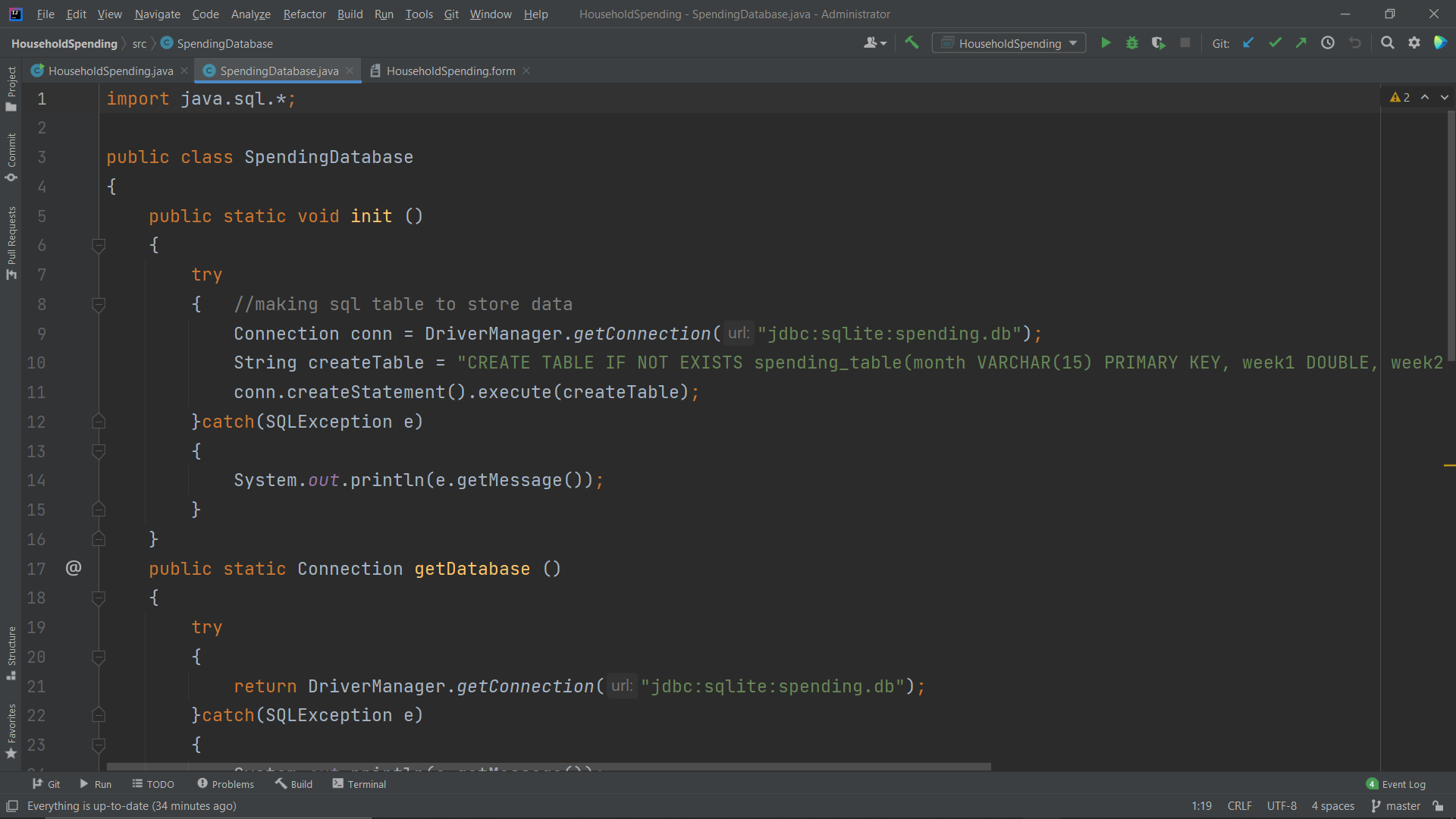
The figure above shows the calculateButton attribute used in the program. @Override means that a child class is over-writing its parent class, this process is called inheritance. The. addActionListener adds a notifier that notifies the program that a button is clicked. Double is the data type used for all the inputs of the program. Double.parseDouble initializes a new variable for the inputted value. getText() extracts this inputted value, which is then used in the **total** and **average** variables to calculate the total and average amount spent. valueOf() is used to convert any given data type into a string.



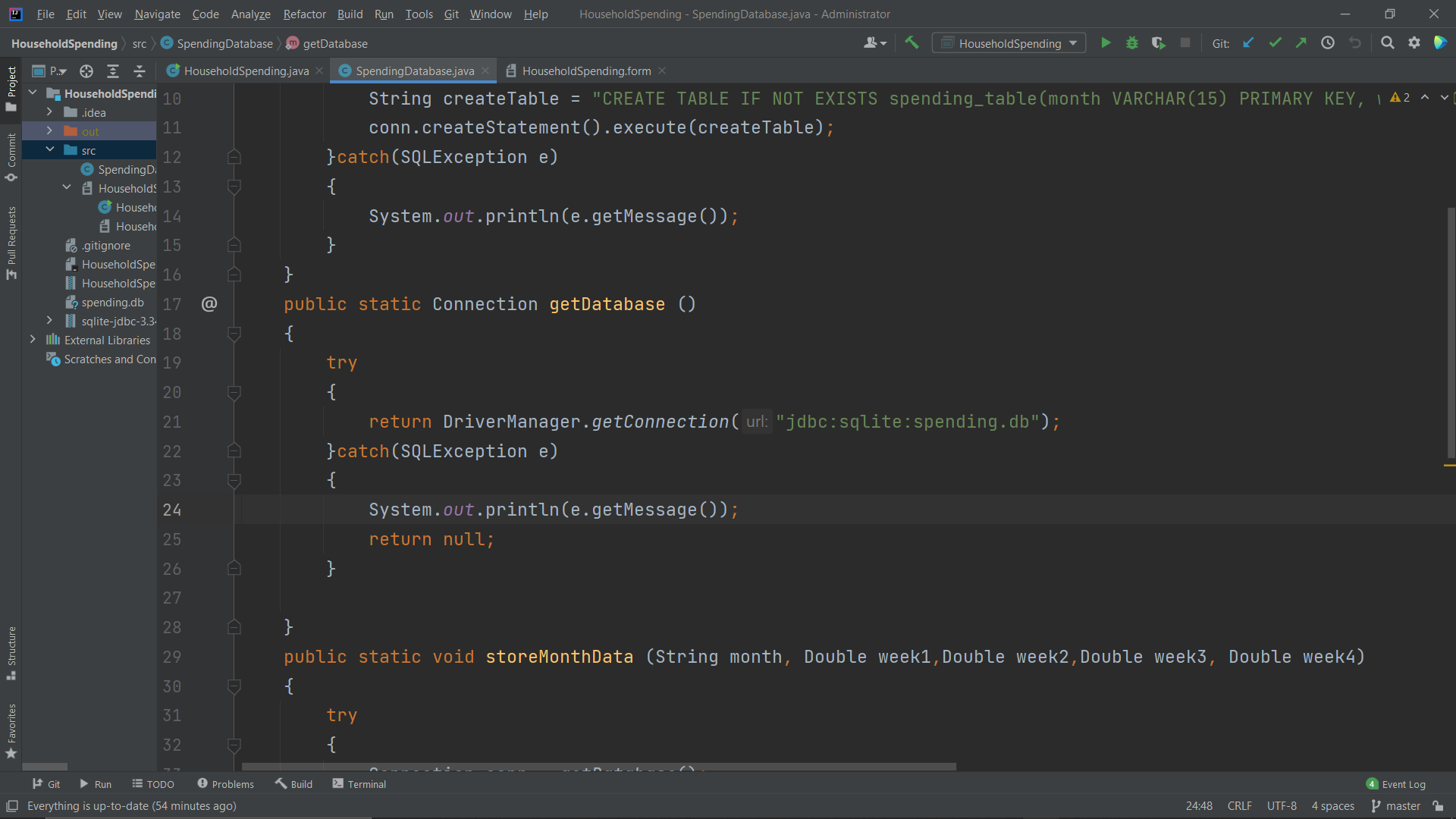
The figure above shows the saveButton attribute used in the program. Similar to calculateButton, saveButton is essentially used to save the inputted data into the database spending.db. SpendingDatabase.storeMonthData inserts the data into a table inside the storeMonthData method in the SpendingDatabase class, which we will get into when explaianing this class later on.



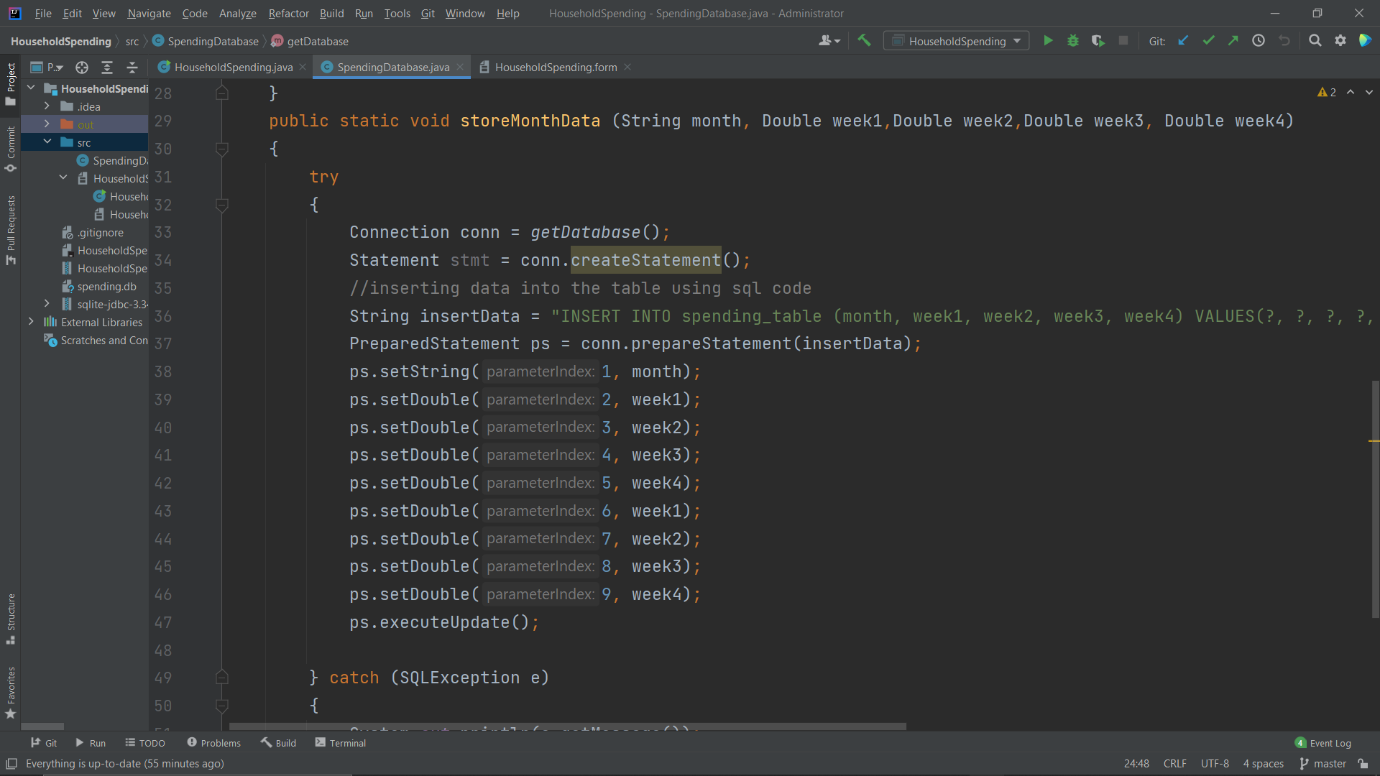
The figure above shows the localButton attribute in the program. The try catch exception in java allows programmers to run their code despite having an error. Everything under “try” will be run whilst its errors are being tested. “catch” basically means that if an error occurs, the code under “catch” is the one that’s going to be run, in this case if there is an error, the system will print out a information regarding the SQL error. The String month variable is used to get the items in the month combo box. Connection conn is a variable used to get the values inside a Statement. The spend variable contains sql code which translates to select everything from the spending\_table (which contains the values stored for week 1-4) from a specific month. “?” means that the selected month depends on the users choice. Prepared statement is used to excecute a query whereas ResultSet is used to call upon the specified row in the spending\_table that contains the selected month. “.next()” is used to point towards the next row in the table.

SpendingDatabase.java

The figure above shows the code for SpendingDatabase class. The init() method contains a try and catch exception, which is explained in the previous class. The DriverManager.getConnection function sets the jdbc driver used to store the spending.db database. The createTable variable contains SQL code to create a table named “spending\_table” inside the database, which is absolutely necessary in order to be able to store values inside the database.



The figure above shows the getDatabase method, which is used to establish a connection between the sql driver and the database



The figure above sjows the storeMonthData method, which is used to store the inputted values month and the 4 weeks as mentioned in the explanation for the previous class. PrepareStatement ps is a variable that contains the data values before it is stored inside the database. There are 8 parameter indexes in the setDouble function for the weeks as 4 are used to store the initial values and the other 4 is used to store the newly inputted values if the user wishes to change the weekly expenditure of a particular month. executeUpdate is used to apply that change into the spending\_table.

# Evidence of Working Program

# 