Homework 07 – Money Minded

Authors: Sreya, Ananay, Ariel, Owen, Allan
Topics: Recursion, Basic Java FX

Solution Description

Please make sure to read the document fully before starting!
This homework will consist of two parts: one part focused on recursion and one on javafx. Create files Money.java and BankAccount.java. You will be creating several fields and methods for each file. Based on the description given for each variable and method, you will have to decide whether the variables/method should be static, and what visibility modifier applies. To make these decisions, you should carefully follow the guidelines on these keywords as taught in lecture. In some cases, your program will still function with an incorrect keyword. Private helper methods are allowed and encouraged.

Money.java

This class will consist of different functions related to dealing with money

Methods:
All methods in this class should be public and must be written recursively or call a recursive method to receive full credit. You may use any helper methods you would like:

- `countTotal()`
  - This method should take in an array of integers, representing an array of cents
    - [3, 5, 14] means 3 cents, 5 cents, and 14 cents
  - Recursively find the total amount of money in dollars contained in the array
  - Return the total amount of money
- `findDifference()`
  - This method should take a double, representing the amount of money required for a purchase in dollars, and an array of integers, representing an array of cents used to pay for the purchase
    - 6.25 [3, 5, 14] means $6.25, 3 cents, 5 cents, and 14 cents
  - Recursively find money the customer **has left to pay** and return it.
    - The above input would return 6.03
  - If the total money in the array is greater than the money required for the purchase, it means the customer will fully pay for the item and have some change left. You should stop recursion when you have exhausted all change, and you should return the amount of change the customer has left, as a negative value.
  - Return the difference, in dollars
- `makeChange()`
  - This method should take in an integer, representing the change you are forming
  - Recursively find a way to make this change using as few coins as possible
  - Assume that change can be made with quarters, dimes, nickels, and pennies
  - Return the minimum number of coins
- `makeChange()`
  - This method should take in an integer, representing the change you are forming, and an array of integers representing values of possible coins

- [14, 5, 3] means that the coins you can use have values of 3 cents, 5 cents, and 14 cents
- The arrays will be ordered from greatest to least
- You can assume that the change can be formed with the coins passed in with no remainder

- Use the greedy method -- use the biggest possible coins before moving onto a smaller coin. E.g 10 cents with an array of [6, 5, 1] should return 5 (6, 1, 1, 1, 1) not 2 (5,5)
- Recursively find a way to make this change using as few coins as possible
- Return the number of coins using the greedy method

BankAccount.java

This class will be a JavaFX program, where you will create a simple visualization of a bank account using shapes. Make sure you have JavaFX downloaded, and follow the instructors on Canvas for installation.

Specifications:

- The title of the window should be "Bank Account"
- The size of your window should have a minimum height of 500
- Include a background image of your choice
- Include a rectangle in the middle of the screen, which is where transactions would be posted
  - You do not need to include transactions, just the rectangle
- Include a circle in the top right corner of the window
  - There should be a text field with a number in this circle that represents that account balance. It can be any number of your choice.

Checkstyle

You must run checkstyle on your submission (To learn more about Checkstyle, check out cs1331-style-guide.pdf under CheckStyle Resources in the Modules section of Canvas.) **The Checkstyle cap for this assignment is 40 points.** This means there is a maximum point deduction of 40. If you don't have Checkstyle yet, download it from Canvas -> Modules -> CheckStyle Resources -> checkstyle-8.28.jar. Place it in the same folder as the files you want to run Checkstyle on. Run checkstyle on your code like so:

```
$ java -jar checkstyle-8.28.jar yourFileName.java
Starting audit...
Audit done.
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up a bove this message, and the number at the end would be the points we would take off (limited by the ch eckstyle cap mentioned in the Rubric section). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

Additionally, you must Javadoc your code.

Run the following to only check your Javadocs:

```
$ java -jar checkstyle-8.28.jar -j yourFileName.java
```

Run the following to check both Javadocs and Checkstyle:

```
$ java -jar checkstyle-8.28.jar -a yourFileName.java
```

For additional help with Checkstyle see the CS 1331 Style Guide.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- `Money.java`
- `BankAccount.java`
- `Whatever image you choose for your background`

Make sure you see the message stating the assignment was submitted successfully. From this point, Gradescope will run a basic autograder on your submission as discussed in the next section. **Any autograder test are provided as a courtesy to help "sanity check" your work and you may not see all the test cases used to grade your work.** You are responsible for thoroughly testing your submission on your own to ensure you have fulfilled the requirements of this assignment. If you have questions about the requirements given, reach out to a TA or Professor via Piazza for clarification. You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your latest submission. **Be sure to submit every file each time you resubmit**.

Gradescope Autograder

If an autograder is enabled for this assignment, you may be able to see the results of a few basic test cases on your code. Typically, tests will correspond to a rubric item, and the score returned represents the performance of your code on those rubric items only. If you fail a test, you can look at the output to determine what went wrong and resubmit once you have fixed the issue.
The Gradescope tests serve two main purposes:

- Prevent upload mistakes (e.g. non-compiling code)
- Provide basic formatting and usage validation

In other words, the test cases on Gradescope are by no means comprehensive. Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or Checkstyle your code; you can do that locally on your machine. Other portions of your assignment can also be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Allowed Imports

You are allowed to import from the `javafx` library.

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`
- `System.arraycopy`

Collaboration

Only discussion of the Homework (HW) at a conceptual high level is allowed. You can discuss course concepts and HW assignments broadly, that is, at a conceptual level to increase your understanding. If you find yourself dropping to a level where specific Java code is being discussed, that is going too

far. Those discussions should be reserved for the instructor and TAs. To be clear, you should never exchange code related to an assignment with anyone other than the instructor and TAs.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files
- Test your code in addition to the basic checks on Gradescope
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- **Check on Ed Discussion for a note containing all official clarifications and sample outputs**

It is expected that everyone will follow the Student-Faculty Expectations document, and the Student Code of Conduct. The professor expects a **positive, respectful, and engaged academic environment** inside the classroom, outside the classroom, in all electronic communications, on all file submissions, and on any document submitted throughout the duration of the course. No inappropriate language is to be used, and any assignment, deemed by the professor, to contain inappropriate, offensive language or threats will get a zero. You are to use professionalism in your work. Violations of this conduct policy will be turned over to the Office of Student Integrity for misconduct.