CSC171 — Project 1

Savings Project: Due Oct 6th 1159pm

Interest. It's in the news. It's in our junk mail. It's on the billboards. We pay interest on loans and credit cards, and we collect interest on savings accounts and other forms of investments. One very practical use of computing is to make financial calculations - and that is the subject of your first project in CSC 171.

You will be constructing a program that calculates and displays values within a typical bank savings account. This will give you practice with strings, integers, and iteration. There are also constraints on the format of the data, so you will also gain experience working with the printf method.

1 Project Requirements

- Reads a line of input from the user that includes the following information (in order): initial deposit, monthly deposit, interest rate, compounding frequency, and a number of years to calculate.
  - Interest compounds after a specified length of time, the compounding frequency will be specified as one of the following all lowercase strings: `"yearly"`, `"monthly"`, or `"daily"`.
  - Interest will be specified using a double which you should interpret as representing a percentage – for example, the value 5.7 should be interpreted to mean 5.7% and it should NOT be interpreted to mean 570%.
  - The initial and monthly deposits will be specified as doubles.
- Your program must produce a monthly report of the projected account value and the total interest earned. The report should simply be the a list numbered by month of the three values at the end of each month, printed one month per line.
- To simplify the calculations, you can ignore leap years and things like that. You can assume monthly means 12 times per year, daily means 360 times, and of course annually means once per year. You should assume that monthly deposits happen before interest is calculated. See sections on example inputs/outputs and sample algorithms for more information.

2 Examples of Program Input/Output

As stated earlier, your program should accept the simulation parameters in a fixed order: principal (double), monthly deposit (double), annual interest rate (double), compounding rate (string), and the number of years to report (integer).

Your program should print a line of instructions for the user, and then read the data as specified. Your program should then print out a summary of the parameters and then begin the simulation.

You should follow the summary and simulation output format exactly. As in, very exactly. Failure to follow this exact format may result in a loss of a points. Below is an example run of the program. Note that the line beginning with the arrow represents user input. Here is the first example run:

```
Enter the following data:  principal monthly apy compounding years
=> 1000 100 10 yearly 10
Simulating $1,000.00 plus $100.00 per month at 10.00% APY compounded yearly for 10 years:
Year 1 ending balance is $2,420.00 with $220.00 from interest.
Year 2 ending balance is $3,982.00 with $582.00 from interest.
Year 3 ending balance is $5,700.20 with $1,100.20 from interest.
Year 4 ending balance is $7,590.22 with $1,790.22 from interest.
Year 5 ending balance is $9,669.24 with $2,669.24 from interest.
Year 6 ending balance is $11,956.17 with $3,756.17 from interest.
Year 7 ending balance is $14,471.78 with $5,071.78 from interest.
Year 8 ending balance is $17,238.96 with $6,638.96 from interest.
Year 9 ending balance is $20,282.86 with $8,482.86 from interest.
Year 10 ending balance is $23,631.14 with $10,631.14 from interest.
```

Here is another example run, demonstrating the floating point nature of the first three parameters:

```
Enter the following data:  principal monthly apy compounding years
=> 3141.59 265.35 8.979 daily 3
Simulating $3,141.59 plus $265.35 per month at 8.98% APY compounded daily for 3 years:
Year 1 ending balance is $6,780.68 with $454.89 from interest.
Year 2 ending balance is $10,761.59 with $1,251.60 from interest.
Year 3 ending balance is $15,116.44 with $2,422.25 from interest.
```

3 Submission Instructions

You should submit your program as `SavingsProject.java` directly to blackboard before the deadline. Students working as a team should see below for details on team submissions. You must ensure that your program compiles without any error exactly as submitted. Programs that do not compile will receive a zero. Do not submit work which does not compile. You should ensure that your source file begins with a comment including your full name and UR email.

4 Collaboration and Academic Honesty

You are allowed to work either alone or in teams of two for this project.

4.1 Rules on Teams

Both members of a team must be from the same lab section. Both members of a team will receive the same grade for the project. Both members of a team must accept full responsibility for maintaining the academic integrity of the team submission.

4.2 Submitting Work as a Team

Only one member of a team should submit the project. The other team member must submit a plain text file indicating their name, and their team member's names and school emails. All files must have both team members names and emails at the top. Students working in a team must include a description of individual team member contributions in the block comment at the top of their primary source file. Failure to follow these instructions may result in a loss of points.

4.3 Avoid Sharing Code

You are welcome, and encouraged, to discuss your project at the level of ideas with your peers outside your team; however, you may not share any code. If your program crashes with exceptions, or fails to compile, you may discuss the error messages with your peers; however, again, you are not allowed to share your code. Sharing code for this project will be considered a violation of the University of Rochester Academic Honesty policy.

In short: Don't share code outside your team, and don't use someone else's code as your own. Do the work yourself.

5 General Advice and Information

5.1 Information about Interest

For the scope of this project, interest is the money that someone with a savings account earns through storing their money in a bank. Depending on the account and bank, it can "compound" every year, every quarter, every month, or even daily. In the equation below, V is the final value of the savings account, r is the annual interest, n is the number of times interest compounds over the year, P is the initial deposit, and t is the number of years over which the investment is projected.

$$V = P(1 + r/n)^{nt} \tag{1}$$

For an initial investment of $1000 at an annual rate of 4% compounded quarterly over ten years, the annual value would be...

After 1 year: $1,040.60
After 2 years: $1,082.86
And so on!

5.2 Typical Interest Rates

According to `Bankrate.com`, here are some recent historical average rates of return:

- Stocks - 13.8%
- 1 Year CDs - 3.25%
- Savings accounts - 2.0%

5.3 Program Design

Should you choose to code this project without a plan beforehand, you may be left with a fairly messy final product. If you wish to avoid this, it would be a good idea to draw out a flowchart of the different parts of your program. Observe pieces of the flowchart that achieve similar goals. Pieces that occur multiple times may be good to use in methods.

For help in figuring out Java syntax and methods, refer to the Java oracle tutorials. Here are two specific links that you may find of particular use:

https://docs.oracle.com/javase/tutorial/java/data/numberformat.html

https://docs.oracle.com/javase/tutorial/java/data/converting.html

5.4 Program Style

Completing this course means learning not only how to code, but how to code *well*. Good code is exemplified through even spacing, appropriate variable naming, and descriptive comments.

1. Even spacing means having a consistent amount of blank space between lines of code. Any preference in the length of white space between blocks of code is acceptable as long as it is consistent.

2. Variable names should be simple and self-explanatory. For example, a variable tracking a score in a game should be named something like "gameScore" rather than a vague word like "var2".

3. Comments should be used to explain pieces of code that are not immediately understandable. Good practice is to comment short explanations in complex parts of a program.

5.5 Algorithm Hints

```
Annually -- yearly algorithm
set account value to initial deposit
add 12 times the monthly deposit to the account
compute the interest by multiplying the account rate by interest
add the interest to the account

Monthly -- yearly algorithm
set account value to initial deposit
add one monthly deposit to the account
compute interest by dividing total interest by 12 and multiply
add interest to account, repeat until 12 months pass

Daily -- yearly algorithm
set account value to initial deposit
add one monthly deposit to the account
compute 30 days interest (divide by 360, compound 30 times)
add interest to account, repeat until 12 months pass
```