Financial Calculators

Student Workshop 1w

Version 7.0 Y

What Are Workshops?

At the end of each week of instruction, you will code a workshop that resembles a 1 to $1^1/_2$ day mini-capstone. Its goal is to reinforce everything you learned about Java during the week in a larger project.

You will create a new GitHub repository for **each** of your workshop projects.

If you haven't created a workshops directory, create a new directory in `C:/pluralsight/`

Mac/Linux: `~/pluralsight/workshops`

Windows: `C:/pluralsight/workshops`

The `C:/pluralsight/workshops` directory will be where all workshops will be stored.

These projects will all be console (CLI) projects. In other words, you will not be creating a fancy UI (user interface) such as a website.

The primary concern of each workshop is for you to complete the requirements in Java. Even though the UI is not the primary concern, you should still consider the formatting of your screen, and try to make the application clean and intuitive for the user.

Remember to `commit` and `push` your code frequently (don't just wait until the project is complete).

Project Description

Create a new GitHub project named `financial-calculators`. As well as a new Java project called financial-calculators, and ensure you create a git repository during creation. After you start your new project, connect your repo to your Github.com's URL.

Read all project requirements before you begin to code. Use your notebook to plan your project. Use the links provided in the Hints section to help research how each calculator should work.

The Project

You will build an application for a financial organization that wants to provide a set of financial calculators for their clients. The screen will prompt the user to select which calculator they would like to use.

They are interested in having you implement as many of the following calculators you can in the time allotted. Expectations: Getting two done would be good, getting three done would be great.

- **Calculator 1: A mortgage calculator** - it is used to calculate out how much a monthly payment for a loan would be (minus any insurance or taxes), as well as how much interest you would pay over the life of the loan.

    a. It would accept the principal, interest rate, and loan length from the user

    b. It would display the expected monthly payment and total interest paid

    Example: A $53,000 loan at 7.625% interest for 15 years would have a $495.09/mo payment with a total interest of $36,115.99

    This calculator would use a compounded interest formula.

    ```
    M = P × (i * (1 + i )^n / ( (1 + i)^n ) - 1)
    ```

- **Monthly Payment (M) =**
    - **Principal (P)**:
        - This is the total amount of the loan.
    - **Annual Interest Rate (r)**:
        - The nominal annual interest rate in decimal form. (e.g. 7.625% = 0.07625)
    - **Loan Term in Years (y):**
        - How many years the loan lasts.
    - **Number of Monthly Payments (n)**:
        - This is 12 × y (Because there are 12 monthly payments per year.)
    - **Monthly Interest Rate (i)**:
        - This is the annual interest rate divided by 12, i.e. r/12

- **Total Interest** = (M×n)−P

- **Calculator 2:** A calculator that determines the **future value** of a one-time deposit assuming compound interest - it is used to help you decide how much a CD will be worth when it matures

    a. It would accept the deposit, interest rate, and number of years from the user
    b. It would display the future value and the total interest earned

    Example: If you deposit $1,000 in a CD that earns 1.75% interest and matures in 5 years, your CD's ending balance will be $1,091.44 and you would have earned $91.44 in interest

    Note: The numbers above assume *daily* compounding

```
FV = P × (1 + (r / 365) )^(365 × t)
```

- **Future Value (FV) =**
    - **Principal (P):**
        - This is the initial deposit amount.
    - **Annual Interest Rate (r):**
        - The nominal annual interest rate in decimal form. (e.g., 1.75% = 0.0175).
    - **Number of Years (t):**
        - The total number of years the deposit will earn interest.
    - **Days Per Year:**
        - Daily compounding assumes 365 days per year.
    - **Total Number of Days:**
        - This is 365 × t (because there are 365 days per year).
- **Total Interest Earned =** FV - P

- **Calculator 3:** A calculator that determines the **present value** of an ordinary annuity. **(Note: this is difficult)**

    a. It would accept the monthly payout, expected interest rate, and years to pay out from the user

    b. It would display the present value of that annuity

    Example: To fund an annuity that pays $3,000 *monthly* for 20 years and earns an expected 2.5% interest, you would need to invest $566,141.46 today.

NOTE: If your results on any of these calculators are off by a few pennies (not dollars!), don't worry. The difference is likely attributable to rounding and we aren't that concerned about it in this academy.

Hints

Although you are not creating a website, these links will give you an idea about what each calculator does:

https://www.bankrate.com/calculators/managing-debt/annual-percentage-rate-calculator.aspx

https://www.nerdwallet.com/article/banking/cd-calculator

https://financialmentor.com/calculator/present-value-of-annuity-calculator

A quick web search will reveal the formulas needed to make each calculator work or you can read up on some of the formulas at:

Mortgage Payment: https://www.quora.com/How-is-the-division-of-principal-vs-interest-calculated-on-mortgage-payments

Future Value: https://www.gobankingrates.com/banking/cd-rates/how-calculate-cd-accounts-value

Note: the *n* in the formula stands for how often the compounding occurs; use 365x per year

Present Value of Annuity: http://www.1728.org/annuity-presval-formulas.htm

What Makes a Good Workshop Project?

- **You should:**
  - Have a clean and intuitive user interface (give the user clear instructions)
  - Implement at least the first two calculators
- **You should adhere to best practices such as:**
  - Create a Java Project that follows the Maven folder structure
  - Create a Java package for your classes
  - Use good variable naming conventions (camelCasing, meaningful variable names)
  - Have clean formatted code that is easy to understand
  - use Java comments effectively
- **Make sure that:**
  - Your code is free of errors and that it compiles
- **Include a README for the project**
  - A README.md file describes your project and should include screen shots of
    * your home screen
    * EACH of the calculator screen you build that shows user input prompts and correct outputs
    * one calculator page that shows erroneous inputs and an error message.
  - ALSO make sure to include one interesting piece of code and a description of WHY it is interesting.