Regis University CC&IS
CS210 Introduction to Programming
Java Programming Assignment 6: Objects and Loops

This assignment will require you to implement three types of Java loops, the **while** loop, the **do-while** loop, and the **for** loop. You will again use an **object** to store data. Looping statements will be used to run some of the statements repeatedly.

> WARNING: Again, this assignment will be ***more challenging*** than the previous assignments, so make sure to ***start early*** and ***allocate enough time*** to complete it, including time you might need to seek any necessary help.

Problem Summary and Equations

Money can be invested, in order to earn interest. Compound interest is interest added to the initial deposit, so that the added interest also earns interest from then on. This addition of interest to the principal is called compounding. But different investments compound interest at different time. Some compound interest daily, some monthly, some quarterly and some annually. For example:

Compounding of an initial investment of 1000.00 **yearly** for 3 years at 4% annual interest.

Calculate 4% of the current balance at the end of each year:
End of year 1: 1000.00 + 40.00 = 1040.00
End of year 2: 1040.00 + 41.60 = 1081.60
End of year 3: 1081.60 + 43.26 = 1124.86
Total earnings = 124.86

Compounding of an initial investment of 1000.00 **quarterly** for 3 years at 4% annual interest.

Calculate 1/4th of 4% of the current balance (i.e. 1%) at the end of each quarter:
End of quarter 1 in year 1: 1000.00 + 10.00 = 1010.00
End of quarter 2 in year 1: 1010.00 + 10.10 = 1020.10
End of quarter 3 in year 1: 1020.10 + 10.20 = 1030.30
End of quarter 4 in year 1: 1030.30 + 10.30 = 1040.60
End of quarter 1 in year 2: 1040.60 + 10.41 = 1051.01
End of quarter 2 in year 2: 1051.01 + 10.51 = 1061.52
End of quarter 3 in year 2: 1061.52 + 10.62 = 1072.14
End of quarter 4 in year 2: 1072.14 + 10.72 = 1082.86
End of quarter 1 in year 3: 1082.86 + 10.83 = 1093.69
End of quarter 2 in year 3: 1093.69 + 10.94 = 1104.63
End of quarter 3 in year 3: 1104.63 + 11.05 = 1115.68
End of quarter 4 in year 3: 1115.68 + 11.16 = 1126.84
Total earnings = 126.84

Note that the final values computed by a program may be off by a penny or two, because the calculated interest is not actually rounded to 2 decimal places internally.

You will write a program to **compare** the earnings from annually compounded interest (i.e. figured and added to the balance annually) to earnings from interest that is compounded quarterly, monthly, or daily.

Overview of Program

This program will contain two classes, in two separate files within your project:

- An **Investment** class to define the data fields and methods for an **Investment** object, containing:
  - ***Six*** data fields
  - A **constructor** method to create an Investment object (with parameters).
  - ***Two* getter** methods to get two data field values.
  - A method to calculate the investment results: the end balance and earnings.
  - A method to display the conditions and results for an investment.
- The main **InvestmentComparison** class, minimally containing:
  - ***Four*** static methods to read and validate user input.
  - A **main** method to display the program description, read the inputs from the user, create Investment objects, call other methods for calculating and displaying the results, and finally display a comparison analysis.

Reminder: *CS210 Coding Standards state:*

- The **break** statement may be used ***only*** to terminate a series of statements within a **case** of the **switch** statement. The **break** statement may ***never*** be used to terminate a loop.
- The **continue** statement may ***not*** be used anywhere within your code.
- A **return** statement will never be used to exit a loop.
- An **exit()** call will ***never*** be used to exit a loop. The ***only*** exception to this rule is when an **exit()** statement is used to end a program due to an error that would prevent its execution.

Program Requirements

NOTE: You must implement all three types of loops within this program (**for, do-while,** and **while**).

1. Create a new project called ***Lastname*Assn6**.
2. Within the project, create an **Investment** class with the following data fields:
   - initial investment (dollars and cents)
   - annual interest rate (percent, e.g. 3.5)
   - term (amount of time money will be invested) in whole years
   - compounding type (char 'A', 'Q', 'M', or 'D')
   - balance (dollars and cents)
   - earnings (dollars and cents)

   Within the **Investment** class:

   ✓ Add a **constructor** with 4 parameters to instantiate an **Investment** type object and initialize all of the data fields.
   - Set the first 4 data fields directly, using the passed in parameter values.
   - Set the **balance** data field to the same value as the **initial investment**.
   - Set the **earnings** data field to 0.

- ✓ Add a **getter** for the **earnings** data field that returns the **earnings** value.
- ✓ Add a **convertCompoundingTypeToString** instance method. The method will:
  - o Return a String equivalent for the compounding type character data field (i.e. "Annual", "Quarterly", "Monthly", or "Daily").
- ✓ Add an instance method to **calculate the investment results** for an **Investment** object.
  - o Create a new variable to hold the converted annual interest rate -- a rate that can be used in math formulas (e.g. 3.5% would convert to 0.035).
  - o Decide which compounding rate to use by examining the **compounding rate** data field, and set up any necessary variables accordingly.
  - o Use a **for** loop to compute the **end balance**.
    (NOTE: You must use a loop, NOT banking interest formulas).

    Interest earned must be calculated and added to the balance for each compounding period.
    - ▪ Calculate annual interest on the current balance using the annual interest rate.
      Add interest to the balance every year.
    - ▪ Calculate quarterly interest on the current balance using a quarterly interest rate (1/4th of the annual interest rate). Add interest to the balance every quarter for 4 quarters per year.
    - ▪ Calculate monthly interest on the current balance using a monthly interest rate (1/12th of the annual interest rate). Add interest to the balance every month for 12 months per year.
    - ▪ Calculate daily interest on the current balance using a daily interest rate (1/365th of the annual interest rate). Add interest to the balance every day for 365 days per year.

    NOTE: You should **not** have ***separate*** loops for each different compounding period. Instead, figure out how to modify the loop variables to work for whichever period you are compounding.
  - o After calculating the end balance, store the correct value into the **earnings** data field. The earnings for an investment are the difference between the end balance and the initial investment.
- ✓ Add an instance method to **display the investment results** for an **Investment** object, as shown:

```

Results for Annual compounding:
   For an initial investment of $   5555.55
   after 5 years at 5.500%
   the ending balance is        $   7260.88
   for earnings of              $   1705.33
```

  - o Display a blank line before the results.
  - o Display a descriptive word (Annual, Quarterly, Monthly or Daily) in the description on the top line. HINT: Use your convert method.
  - o Display dollar figures to 2 decimal places, and interest rates to 3 decimal places.
  - o Dollar signs and decimal points of the dollar values should line up.

2. Create an **InvestmentComparison** class that will contain a **main** method.

   Within the **InvestmentComparison** class:

   - ✓ Create ***three separate static methods*** to read and return a ***valid*** value for each user input (initial investment, annual interest rate, and term years).
     - o Error check each of the above items is valid, using loops.
       - ▪ All three values must be at least 1. Define this amount as a **class level** constant.
       - ▪ Initial investment cannot be more than $500,000.00.
       - ▪ Annual interest cannot be more than 30%.
       - ▪ Terms can be up to (and including) 75 years.
     - o Valid value ***upper*** limits for all items should be stored as **local** constant values.
     - o For each item, within its method, loop and re-prompt the user until a valid value is entered. All error messages will use the constants to tell the user what values **are** valid.
   - ✓ Create ***another static method*** to read and return a ***valid*** compounding type choice from the user.
     - o Let the user choose from a **menu** which compounding type will be compared to annual compounding (choices are: compounded quarterly, monthly, or daily).
     - o A fourth menu choice, **exit** the program, should be included on the menu.
     - o Loop, displaying the menu and reading the user choice, until the user enters a **valid** choice from the menu.
   - ✓ Create a **main** method that will:

     Display a description of what the program will do, for the user.

     Loop:
     - o Call the method to read the compounding choice from the user (sample shown).

```
Compare Annual Compounding to:
    Q - Quarterly Compounding
    M - Monthly Compounding
    D - Daily Compounding
    E - Exit Program
Enter choice from Menu above: m
```

       Unless the user chooses to Exit, continue:
       - ▪ Call the methods to ask the user for the initial investment, annual interest rate, and term years, and store the returned results.
       - ▪ Call the constructor to create **two** different objects of the **Investment** class type.

         Both objects will have the ***same*** initial investment, annual interest rate, and term years. But they will have ***different*** compounding periods.
         - ❖ The ***first*** **Investment** object will be created to use **annual** compounding.
         - ❖ The ***second*** **Investment** object will be created to use the compounding type chosen by the user from the menu.

        - For each object, call the instance method that will calculate its investment results.
        - For each object, call the instance method that will display its investment results.
        - Display a blank line.
        - Using whatever code is necessary, display the difference in the investment results, and then pause, as follows:

```
Compounding Analysis:
   You would earn $ 48.58 more with Monthly compounding
   vs. Annual compounding.

Press ENTER to continue...
```

          When the user presses ENTER to continue, the program will **loop** and re-display the menu.

    - Continue looping and re-running the program, until the user chooses EXIT from the menu.
    - The program should ONLY exit when the user chooses EXIT from the menu. Do NOT ask the user if s/he wishes to run the program again – just redisplay the menu each time until EXIT is chosen.

3. The program must follow the **CS210 Coding Standards** from Content section 6.10.
   Be sure to ***include*** the following comments:
    - Comments at the ***top of each code file*** describing what the class does
        - Include **tags** with the author's name (i.e. your full name) and the version of the code (e.g. version 1.0, Java Assn 4)
    - Comments at the ***top of each method***, describing what the method does
        - Include **tags** with names and descriptions of ***each*** parameter and return value.

   Delete any default comments supplied by the IDE that you did not use.

Run, debug, and test your Java program with different inputs, until you are sure that all control structures within your program work correctly.

Submission

This programming assignment is due by midnight of the date listed in the **Course Assignments by Week**.

From this point in the course forward, programs that **do not compile** without errors **will not be accepted**. So be sure your program compiles without errors, before submitting it.

Again, you will submit a single **zip** file containing all of the files in your project.

- First export your project from NetBeans:
    - Highlight the project name.
    - Click on **File** from the top menu, and select **Export Project**.
    - Select **To ZIP**
    - Name your export file in the following format:
      <lastname>Assn<x>.zip
      For example:
      SmithAssn6.zip

NOTE: Save this zip file to some other directory, not your project directory.

- Then submit your **.zip** file to the **Java Prog Assn 6** assignment submission folder (located under the **Assignments/Dropbox** tab in the online course).
  - Warning: Only NetBeans export files will be accepted.
    Do not use any other kind of archive or zip utility.

Grading

Your program will be graded using the **rubric** that is linked on the same assignment page from which this program requirements file was downloaded.

WARNING:
*Programs submitted more than 5 days past the due date will **not** be accepted,*
and will receive a grade of 0.

Sample Full Program Run

```
Program to demonstrate the benefits of compounding interest more often.

Compare Annual Compounding to:
    Q - Quarterly Compounding
    M - Monthly Compounding
    D - Daily Compounding
    E - Exit Program
Enter choice from Menu above: m

Enter your initial investment:
5555.55

Enter the annual interest rate:
5.5

Enter the term in years:
5

Results for Annual compounding:
   For an initial investment of  $    5555.55
   after 5 years at 5.500%
   the ending balance is         $    7260.88
   for earnings of               $    1705.33

Results for Monthly compounding:
   For an initial investment of  $    5555.55
   after 5 years at 5.500%
   the ending balance is         $    7309.46
   for earnings of               $    1753.91

Compounding Analysis:
You would earn $ 48.58 more with Monthly compounding vs. annual
compounding.

Press ENTER to continue...


Compare Annual Compounding to:
    Q - Quarterly Compounding
    M - Monthly Compounding
    D - Daily Compounding
    E - Exit Program
Enter choice from Menu above: e
BUILD SUCCESSFUL (total time: 24 seconds)
```