CS 1063 Lab 3: Computing Future Values

The `FutureValues` Class

Objectives

- Call methods with parameters and return values.
- Use the Math class.
- Use Scanner to input values.

Hand-in Requirements

All projects and laboratories will be submitted electronically through Blackboard. Zip up your entire lab directory to submit as the source. (Right click on the lab folder and follow **Send To > Compressed (zipped) Folder** or **7-Zip > Add to "lab3.zip"**.) The lab folder should include the following:

- `FutureValues.java`
- `FutureValuesOutput.txt`

Tasks

Write a program that prints

```
Lab 3 written by YOURNAME
```

and calls two methods.

1. Compute and return the future value of an account based on the present value of the account, the interest rate, and the number of years.
2. Compute and return the future value of an annuity based on the payment per year, the interest rate, and the number of years.

For each method, the main method needs to obtain input from the user, call the method with the input values, save the result of the method in a local variable, and print the inputs and the result.

Details

Future Value Using Compound Interest

If the *present value* of an account is \$1000 and the interest rate is 5%, then after one year, the account will increase by \$50 (5% of \$1000). In the second year, the interest applies to all \$1050, so the account will increase by \$52.50 (5% of \$1050). Getting future interest on past interest is called *compound interest*. A general formula for future value assuming p is the present value, r is the interest rate, and y is the number of years is:

$$\text{future value} = p * (1 + r / 100)^y$$

Your method should have the following characteristics:

- It should have three `double` parameters:
  - present value
  - interest rate
  - number of years
- It should return a `double`, the future value.
- It should use `Math.pow` in the calculation.
- It should *not* have any print statements. The main method should do all the printing.

For examples with similar characteristics, see the `hypotenuse` method in the book and the `max3` method in the lecture notes.

Future Value of an Annuity

For a typical annuity, you pay a certain amount every year (or some other period of time) for so many years, and an interest rate is applied to your payments. It's like a bank account where you deposit money regularly and wait several years to withdraw anything. In return, you are guaranteed a certain interest rate.

For example, suppose the payment is \$100 by the end of each year and the interest rate is 5%. In the first year, your annuity will be worth \$100. In the second year, you get \$5 interest (5% of \$100), and you make a payment of \$100, so the annuity will be worth \$205 after two years. In the third year, you get \$10.25 interest (5% of \$205), and you make another payment of \$100, so the annuity will be worth \$315.25 after three years.

A general formula for the future value of an annuity assuming p is the yearly payment, r is the interest rate, and y is the number of years is:

$$\text{future value} = p * \frac{(1 + r / 100)^y - 1}{r / 100}$$

Your method should have the following characteristics:

- It should have three `double` parameters:
  - yearly payment
  - interest rate
  - number of years
- It should return a `double`, the future value.
- It should use `Math.pow` in the calculation.
- It should *not* have any print statements. The main method should do all the printing.

For examples with similar characteristics, see the `hypotenuse` method in the book and the `max3` method in the lecture notes.

Printing Money

When you are printing `doubles`, you will find out that Java often prints a lot of decimal places. To print out your monetary amounts as reasonable looking `Strings`, try the following method in your lab.

```
// Returns a String $dollars.cents rounded to the nearest cent.
// For example, moneyString(12.3456) returns "$12.35".
public static String moneyString(double amount) {
    DecimalFormat dollarsAndCents = new DecimalFormat("$#0.00");
    return dollarsAndCents.format(amount);
}
```

If you use this method, you'll need the following import statement at the top of your class.

```
import java.text.*;
```

Rubric

Your program should compile without any errors. A program with more than one or two compile errors will likely get a zero for the whole assignment.

The following criteria will also be used to determine the grade for this assignment:

- [5 points] If your submission includes the following:
  - The main method prints "Lab 3 written by [...]".
  - Your submission was a Zip file named `lab3.zip` containing a folder named `lab3`, which contains the other files.
  - If your Java program was in a file named `FutureValues.java`.

    - If the output of your Java program was in a file named `FutureValuesOutput.txt`.
    - If your program contains a comment that describes what the program does and contains a comment for each method.
    - If your program is indented properly.
- [5 Points] If the main method does all of the user input and does all of the printing.
- [5 Points] If the main method calls a method to compute future value with compound interest. This method should have three parameters, should perform the correct calculation, should return the correct value, and should NOT print anything.
- [5 Points] If the main method calls a method to compute future value of an annuity. This method should have three parameters, should perform the correct calculation, should return the correct value, and should NOT print anything.

Revision Date Tue Feb 25 2014 09:08:43 GMT-0600 (Central Standard Time)