Assignment 2: Personal Finance

COMP-202, Fall 2016 (all sections)

Due: October 21st, 2016 - 11:55pm

Please read the entire pdf before starting. You must do this assignment individually.

Part 1:	0 points
Part 2,Question 1:	20 points
Part 2,Question 2:	5 points
Part 2,Question 3:	30 points
Part 2,Question 4:	10 points
Part 2,Question 5:	5 points
Part 2,Question 6:	20 points
Part 2,Question 7:	10 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment through automated tests. While these tests will not determine your entire grade, it will speed up the process significantly, which will allow the TAs to provide better feedback and not waste time on administrative details. Marks can be removed if comments are missing, if the code is not well structured, or if your solution does not follow the assignment specifications. Graders have the discretion to deduct up to 15% of the value of this assignment for not adhering to the best practices, as discussed in class and on the syllabus.

Late policy: Assignments are accepted up to two days (48 hours) late with a penalty of 10% off per day (or fraction thereof). After two days, no late assignments will be accepted without explicit permission from the section instructor, granted only for exceptional circumstances.

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions. You are responsible for knowing all of the material in these questions.*

Warm-up Question 1 (0 points)

Create a file called `Counting.java`, and in this file, declare a class called `Counting`. This program takes as input from the user a positive integer and counts up until that number. eg:

```
run Counting 10
I am counting until 10: 1 2 3 4 5 6 7 8 9 10
```

Warm-up Question 2 (0 points)

For this question you have to generalize the last question. The user will give you the number they want the computer to count up to and the step size by which it will do so.

```
run Counting 25 3
I am counting to 25 with a step size of 3:
1 4 7 10 13 16 19 22 24
```

Warm-up Question 3 (0 points)

A perfect number is a number whose factors (numbers that evenly divide the number) sum up to the number itself. For example, the number 28 is a perfect number because its factors are 1,2,4,7 and 14. These numbers added together equal 28. We do not count the number itself as a factor for the purpose of this calculation. Write a method `isPerfectNumber` that returns a boolean representing whether a number is perfect or not.

Warm-up Question 4 (0 points)

Write a method `sumDigits` which returns the sum of the digits of a number. For example `sumDigits(145)` should return 10. Hint: Use the modulo operator. (There is more detail in question 3 of the graded part below.) Next, write a program that uses this program to take a number the user provides as `args[0]` and prints the sum of the digits.

Warm-up Question 5 (0 points)

A prime number is a positive number whose only even divisors are 1 and itself. Write a method `isPrime` that takes as input an integer `n` and returns a `boolean` representing whether `n` is prime or not. Make sure to handle cases where the integer is negative. (Your method should return false in these cases.)

Warm-up Question 6 (0 points)

Write a method `firstPrimeNumbers` which takes as input an `int n` and returns an `int[]` . The `int[]` should contain the first `n` prime numbers.

Warm-up Question 7 (0 points)

x is a factor of y if y is a multiple of x. Write a method `calculateFactors`. The method should take as input an `int n` and return an `int[]` containing all the factors of the number `n`. Hint: You will need two separate loops to do this!

Warm-up Question 8 (0 points)

Write a method `numberToDigits` which takes as input an integer and returns an `int[]` with all of the digits of the original number in the array. For example, if the number is `136` it should return an array with contents {`1,3,6`}

Warm-up Question 9 (0 points)

Write a method that prints the outline of a square made up of # signs. It should take as input the length of the sides in number of #s. Using two loops, you should be able to draw the outline of a square as follows.

```
##########
#        #
#        #
#        #
#        #
#        #
#        #
#        #
#        #
##########
```

N.B. It is normal that the square does not appear to be a perfect square on screen as the width and the length of the characters are not equal.

Part 2: Graded Assignment

The questions in this part of the assignment will be graded.
You are expected to properly indent your code, and to add comments where appropriate. Marks will be deducted for badly formatted and uncommented code.

We don't ask you to write a main method until question 7, but to verify that your methods work, you should test them inside the main method. You can then delete that test code when you get to question 7.

The following should go into a class `YearlyBudget`.

Question 1: Calculating Tax (20 points)

In Canada, taxes are not *flat*. What this means is that portions of your salary after certain amounts are taxed at a higher rate.

For example, the first 10,000 dollars you earn are tax free. Then the amount you earn between 10,000 dollars and 30,000 dollars could be taxed at a rate of 25 percent. The amount you earn from 30,000 to 50,000 could be taxed at a rate of 40 percent. See example below for computations.

Write a method `calculateTax` that takes in seven input arguments (see details below) and returns the amount you pay in taxes.

The seven values are as follow. Your method is required to take them as input in this order.

1. `double yearlyIncome`. This represents the amount of money that you make in a year.
2. `double bracket1Dollars`. This represents a dollar amount. All money earned less than or equal to the value `bracket1Dollars` is not taxed.
3. `double bracket1Rate`. This represents a percentage. All money earned greater than `bracket1Dollars` but less than or equal to `bracket2Dollars` (below) is taxed at this rate. Note that the input is a percentage but can be converted to a proportion by dividing by 100.
4. `double bracket2Dollars`. This is the upper boundary used for `bracket1Rate` and the lower boundary used for `bracket2Rate`.
5. `double bracket2Rate`. This is the same as `bracket1Rate` except it applies to all income earned greater than `bracket2Dollars` and less than or equal to `bracket3Dollars`

6. `double bracket3Dollars`. This is the upper boundary used for `bracket2Rate` and the lower boundary used for `bracket3Rate`.
7. `double bracket3Rate`. This is the tax rate (as a percentage) used for all income earned greater than `bracket3Dollars`

Here is an example. Consider the following inputs to the method:
`calculateTaxes(100000, 10000, 20, 20000, 30, 45000, 50)`

1. The first $10,000 are tax free.
2. The money from $10,000 to $20,000 is taxed at a rate of 20%, meaning a tax so far of $2,000.
3. The money from $20,000 to $45,000 is taxed at a rate of 30%. This is $25,000 taxed at 30% which is $7,500.
4. The money earned from $45,000 to $100,000 is taxed at a rate of 50%. This is $55,000 taxed at 50%, which is $27,500.
5. The total tax paid is thus $\$2000 + 7500 + 27500 = \$37000$, so your method would return the double value `37000`.

You may assume only positive numbers are given as input to your method. You may also assume that `bracket1Dollars` is less than `bracket2Dollars` and that `bracket2Dollars` is less than `bracket3Dollars`.

Question 2: Monthly savings (5 points)

Write a method `monthlySavings` that takes as input two doubles, the first of which represents your total monthly expenses and the second of which represents your monthly income (after tax). This method returns the amount you saved that month.

Question 3: Credit Card Validation (30 points)

For this question you are not allowed to use Strings.

When you purchase a product online, a simple checksum validation is performed to detect simple typos in the number that was input. In this question, you will implement that checksum.

Write a method called `validateCreditCard` which takes as input a `long` and returns a boolean. It determines whether or not the long value stores a valid credit card number based on the checksum (see below) and returns `true` if it is valid and `false` if it is not valid.

To compute the checksum, you must perform the following steps:

1. Counting from the right, sum the digits at odd locations (e.g. 1st digit, 3rd digit, 5th digit, etc).
2. Counting from the right, take all of the digits at even locations (e.g. 2nd digit, 4th digit, 6th digit, etc), double their values, take the result % 9, and then sum those results.
3. Add the results from the previous 2 steps together.
4. If the sum is a multiple of 10, then the credit card number is valid. Otherwise, it is not valid and therefore the card is not valid.

For more details, consider the example below of validating the card with number `1234 5678 1234 5678`

1. First, we sum the digits at even locations. The digits at even locations are 8,6,4,2,8,6,4,2. The sum of these digits is 40.
2. Next, take the sum of all the digits at odd locations doubled % 9:

   The digits at odd locations are 7,5,3,1,7,5,3,1. After doubling these digits, we are left with the digits 14,10,6,2,14,10,6,2. We then take each of these digits modulo 9 and get 5,1,6,2,5,1,6,2. Next, we sum them to get 28.
3. Finally, we add the 2 numbers together. 40 plus 28 is 68, which is not a multiple of 10. Thus the credit card is invalid.

If you have a Visa or Mastercard, you should be able to test it this way. **Please do not leave your personal credit card number anywhere in your code when you submit!**

Hint 1: To extract the ones digit of a number, you can take the number % 10. In math, when you divide a number by 10, you move the decimal point one to the left. This means that in Java, dividing an int or long by 10 has the effect of stripping the number of its ones digit. Use these facts to extract individual digits.

Hint 2 : It is not necessary to use an array on this question.

Hint 3: Recall that to type a `long` literal, you must add `L` to your literal. For example `123` is interpreted as an `int`, `123.0` is interpreted as a `double`, and `123L` is interpreted as a `long`.

You may assume the long given as input is a positive 16 digit number, and that the first digit is not zero.

Question 4: Build Expenses (10 points)

Write a method `buildExpenses` which takes as input a double representing the monthly rent and returns a `double[]` of all of the expenses of a person in a year. It does this by creating an array of size 12 (with index 0 representing January, index 1 representing February, etc) and then filling it with values according to the following rules. Every month you must pay rent, and in addition there are the following expenses:

1. You spend 600 dollars every month on food, cell phone, internet, and other miscellaneous expenses.
2. In January and June, you go to the dentist and spend $200.
3. In September, you spend $300 on textbooks, none of which are from COMP 202, which has a free textbook.
4. In April, July, and September, you spend $100 on a family member's birthday.
5. In December, you spend $200 for the holidays.

In order to get any marks on this question, you *must* use a loop to fill out your array.

Question 5: Build Payments (5 points)

Write a method `buildPayments` which takes as input a double representing a yearly post-tax income, and retuns a `double[]` representing monthly credit card payments. The payments are made according to the following rules:

- As a minimum, every month you spend 10% of your income that month towards paying off the card balance.
- In December, you get a large monetary gift from your family that you don't declare on your taxes :) Because of this, you spend an addition $150 that month on paying off your credit card.
- In September, your family also gives you a large gift as you head off to university. You spend the entire $200 on paying off your card.

In order to get any marks on this question, you *must* use a loop to fill out your array.

Question 6: Printing expenses (20 points)

Write a method `printBalance` that takes as input your credit card balance, the yearly interest rate as a percent, an array of expenses by month, and an array of payments by month. Each array has 12 elements in it, one for each month. You may assume index `0` is January, index `1` is February, etc.

This method prints the balance owed on the card at the end of each month. To do this, it should start with initial balance and add to it the expenses in the first month. Next subtract the payments for that month. Finally, calculate the interest on the total and add it to the existing total. The monthly interest rate is the yearly rate divided by 12. Note that in addition you need to convert the percentage to a proportion by dividing by 100. Repeat this for the entire year.

Your method should print a table as follows (for example - these are made up numbers):

```
Month 1 balance 100.0
Month 2 balance 50.0
Month 3 balance 30.0
Month 4 balance 200.0
Month 5 balance 40.0
Month 6 balance 80.0
Month 7 balance 3000.0
Month 8 balance 2.0
Month 9 balance 10.0
Month 10 balance 50.0
Month 11 balance 0.0
Month 12 balance 0.0
```

Note: If a payment is more than the balance, then it is possible to have a negative balance. In this case, the balance can be negative but there is no interest charged.

Question 7: Putting it all together (10 points)

Now, write a `main` method and add it to your program. Your program should take as input via command line arguments, your pre-tax income, your present credit card balance, the annual interest rate of the card (as a percent), the credit card number, and your monthly rent.

It first checks if the credit card number is valid. If it is invalid, your program should print "Invalid card" and then exit. Otherwise, it should use the above methods to display your monthly credit card balances for the year, as well as your monthly additions to your savings account. At the end it should also display your total cumulative savings for the year.

Note that you can use `Long.parseLong` to convert a `String` to a `long`

What To Submit

You have to submit one zip file called `Assignment2_YourName.zip` with all your files in it to MyCourses under Assignment 2. If you do not know how to zip files, please ask any search engine or friends. Google might be your best friend with this, and a lot of different little problems as well.
`YearlyBudget.java`