CS1073 - Assignment #6 - Fall 2020

Submission Deadline: Friday, October 23rd before 12:00 NOON (Atlantic Daylight Time Zone) in the Assignment 6 dropbox in Desire2Learn. (Read the submission instructions at the end of this document carefully).

The purpose of this assignment is:

- to gain practice with arithmetic in Java,
- to review decision statements, and
- to introduce the **`while`** loop.

This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during a help session in the "Faculty of Computer Science Student Success Centre" team or to your course instructor.

As always, begin by creating a new folder to hold your work for this assignment.

I. Checking for Leap Year

Write a Java application named **`LeapYearCheck`** that prompts the user for a year and then prints out whether or not that year is a leap year.

Sample output (note: user input is shown here in italics and green):

```
Please enter a year: 2020
2020 is a leap year!
```

Continued on the next page…

A leap year is a year in which February has 29 days. A year is a leap year if it is divisible by 4, unless it is also divisible by 100 but not 400. For example, the year 2019 is not a leap year, but 2016 is. The year 1900 is not a leap year because it is divisible by 100, but not divisible by 400. The year 2000 is a leap year because even though it is divisible by 100, it is also divisible by 400.

If the user enters a year before 1582 (the year the Gregorian calendar was adopted), you program should not try to determine whether or not it is a leap year. Instead, you should print out an error message and prompt the user for a new value. For example:

```
Invalid year. You may not enter a date prior to 1582.
Please enter a new year:
```

Do this repeatedly until a valid input value is provided.

NOTE: For this question you only need to write one class. You may place all of your code in the main method of that class.

Be sure to write a javadoc comment for your **`LeapYearCheck`** class. This comment should include a one-line description of the class and **`@author`** information.

After you have tested your application and you're sure that it works properly, save sample output. Include both invalid input (prior to 1582) and valid input. For this question, take a snapshot of the terminal (or Command Prompt) window and save that as your sample output (like you did for Assignment #1). That way, the markers will be able to see the input values as well. Adjust the size of your terminal window before capturing it to make sure that it will be legible when copied into your report document.

I. Making Change

Write a Java application named **`MakingChange`** that prompts the user for the total price of a customer's purchases, and then prompts the user for the amount of money that the customer gave to the cashier. (Use the nextDouble() method from the Scanner class to read in the numbers.) The program should then calculate and display the change that the cashier should give to the customer (i.e. the quantity of each type of bill and coin that they should receive). For the purpose of this exercise, let's assume that pennies still exist (as this will simplify the problem a bit). Please make your output resemble the example shown below.

Continued on the next page…

Sample output (note: sample user input is shown here in green and italics):

```
Please enter the total price: $56.20
Please enter the amount paid: $100.00

Here is the change that they are due:
$20 bills: 2
$10 bills: 0
$5 bills: 0
Toonies: 1
Loonies: 1
Quarters: 3
Dimes: 0
Nickels: 1
Pennies: 0
```

Hint: You may find this question easier if you convert monetary values from dollars to cents before performing calculations (e.g. $43.80 is 4380 cents.)

NOTE 1: The total price that the user enters should be positive. If the user enters a non-positive number (i.e. 0 or a negative number), your program should print out an error message and prompt the user for a new input value for the total price. For example:

```
Please enter the total price: $0
Invalid input.  You must enter a positive number.
Please enter a positive total price now: $
```

Do this repeatedly until a positive input value is provided.

NOTE 2: You should also make sure that the amount that the customer paid is greater than or equal to the total price. If it is not, your program should print out an error message and prompt the user for a new input value for the amount paid. For example:

```
Please enter the total price: $27.50
Please enter the amount paid: $25.00
Invalid input.  The amount paid must be greater than or equal to
the price.
Please enter a higher amount paid now: $
```

Do this repeatedly until a valid input value is provided.

NOTE 3: For this question, you only need to write one class. You may place all of your code in the main method of that class.

Continued on the next page…

Be sure to write a javadoc comment for your **`MakingChange`** class. This comment should include a one-line description of the class and **`@author`** information.

After you have tested your application and you're sure that it works properly, save sample output. Include both invalid input (see NOTES 1 & 2 above) and valid input. For this question, take a snapshot of the terminal (or Command Prompt) window and save that as your sample output (like you did for Assignment #1). That way, the markers will be able to see the input values as well. Adjust the size of your terminal window before capturing it to make sure that it will be legible when copied into your report document.

Submission instructions are on the next page…

Your electronic assignment submission (submitted via Desire2Learn) will consist of two files:

i. a written report. This should begin with a title page; just as we described in Assignment #1, your title page should include: the course (CS 1073), your section (FR01A, FR02A, FR03A or FR04A), the assignment number, your full name, and your UNB student number. That should be followed by four sections, with each part clearly identified with a section heading. Include:

   a. the source code for Part I (LeapYearCheck.java),

   b. the sample output for Part I,

   c. the source code for Part II (MakingChange.java), and

   d. the sample output from Part II.

   This written report should be prepared using a word processor; we recommend using Microsoft Word (i.e. create a .docx file for your report). Copy & paste your java source code & required output into the report document. Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code to maintain proper indentation.) Once the report is complete and you've checked it all over, save the .docx file for your own records, and then **save a second copy in pdf format for submission**. (Note: Be sure to open that file in a pdf viewer to verify that the pdf was generated correctly.) The **SINGLE pdf file** containing your report will be submitted to the appropriate assignment drop box on Desire2Learn. (It is important that you submit a pdf file and NOT the original Word document. This pdf will allow the marker to write comments directly on your work to give you better feedback.)

   Note: Please name this report as follows: **YourName_As6_Report.pdf**

ii. an archive file **(.zip)** that contains your Java source code and output for this assignment. Make sure that your archive includes all .java files (in case the marker wishes to compile & run your code to test it). You should not include the report document or the .class files in your archive. This archive should be submitted as a **single file** to the appropriate drop box on Desire2Learn.

   Note: Please name this archive file as follows:
   YourName_As6_Archive.zip

End of Assignment 6

Maintained by Natalie Webber