Project #2: Change Maker

Overview

Write a program that accepts the price for an item, amount of payment, and computes the change required to repay a customer.

Sample Sessions

The logic for this program is best communicated by example sessions that must be mimicked *exactly*. In the sessions below, user input is in green.

Session 1: Normal interaction

```
Cost of transaction (enter 0 or negative to exit; max is $500.00): 10
Amount due (with 5.00% tax): $10.50
Please enter payment amount: 10.50
Exact Change! Amazing!

Cost of transaction (enter 0 or negative to exit; max is $500.00): 10
Amount due (with 5.00% tax): $10.50
Please enter payment amount: 11
Change back $0.50
$100: 0   $50: 0   $20: 0   $10: 0   $5: 0   $1: 0   $0.25: 2   $0.10: 0   $0.05: 0   $0.01: 0

Cost of transaction (enter 0 or negative to exit; max is $500.00): 0

Done.
```

Session 2: Quitting Immediately

```
Cost of transaction (enter 0 or negative to exit; max is $500.00): -1

Done.
```

Session 3: Excessive Price

```
Cost of transaction (enter 0 or negative to exit; max is $500.00): 1000
Cost of transaction (enter 0 or negative to exit; max is $500.00): 501
Cost of transaction (enter 0 or negative to exit; max is $500.00): 500
Amount due (with 5.00% tax): $525.00
Please enter payment amount: 1000.57
Change back $475.57
$100: 4   $50: 1   $20: 1   $10: 0   $5: 1   $1: 0   $0.25: 2   $0.10: 0   $0.05: 1   $0.01: 2

Cost of transaction (enter 0 or negative to exit; max is $500.00): 0

Done.
```

Requirements

- The name of the class that contains the main method must be `ChangeMachine`.
- Constants must be defined and used for the tax rate (5%) and the maximum price ($500).
- Output of all dollar amounts (other than dollar-bill change denominations) shall be of the form `$X.XX` where only two decimal points are printed; use a `DecimalFormat` object.
- The program will quit when the user enters a non-positive number.
- The program will loop until a valid price is entered ($0 < price \leq 500$).
- The program will compute and output the most appropriate change given the amount of change that must be returned to the customer. That is, $5 should be returned as a single $5 bill and not five $1 bills; similarly, for coins.

Recommendations

- Before coding, work out on paper the logic for making change.
- Write the program first to make change (no loops).
- Once change is made properly, add loops and other error-handling code.
- Be thorough in your testing because it is an instructor's job to try and break your code.

Submitting

Header Comments

Your program must use the following standard comment at the top of *each source code file*. Copy and paste this comment and modify the parenthesized values accordingly.

```
/*
 *  @author (Student Name)
 * <p>      (File Name)
 * <p>      (Assignment)
 * <p>      (Describe, in general, the code contained.)
 */
```

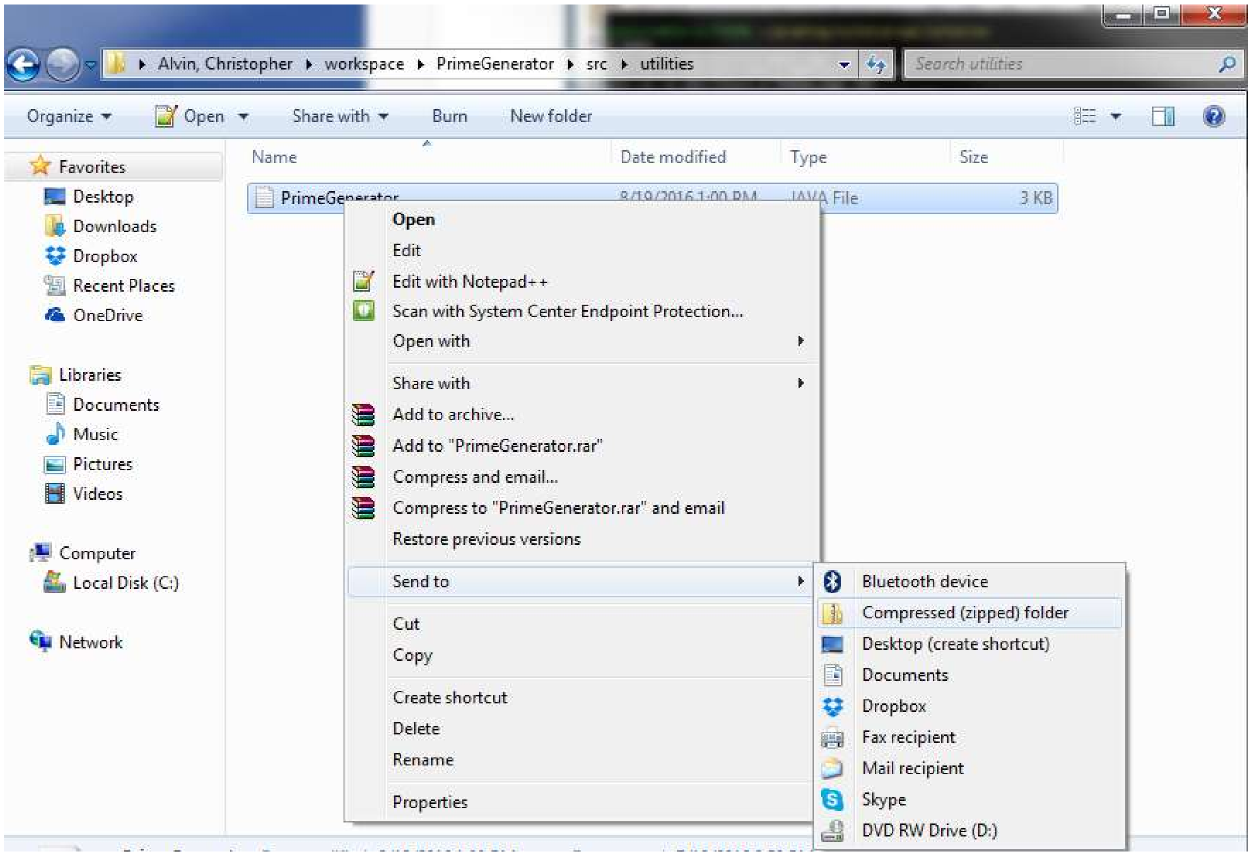
Inline Comments

Please comment your code with a *reasonable amount of comments* throughout the program. Each *block* of code (3-4 or more lines in sequence) in a function should be commented.

Although it is an issue of style and preference, please avoid *long* comments to the right of lines of source code. Long, ubiquitous comments to the right of code will result in a deduction.

Final Submission File

Create a zip file (`proj2.zip`) containing *only* the source code files (ChangeMachine.java). Please note that the zip must not contain any subfolders or other extraneous files. In Windows, (1) select all the source files in a folder, (2) right-click, and (3) Send to > Compressed (zipped) folder:



Submit your zip file via Sakai under `Assignments > Project 2`. Be sure to review the university policy on academic dishonesty: this is an individual project.