**CSC 243 Spring 2015**

**Java Programming**

**Programming Project 2**

**Inheritance**

**Coin Counter Simulation**

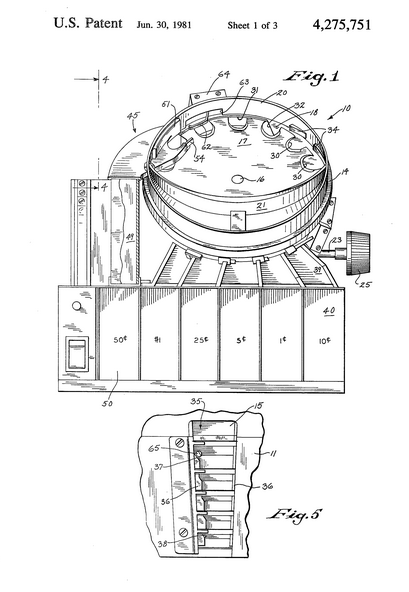
**THIS ASSIGNMENT IS DUE ON MARCH 24th, 2015 at 5:00 PM.**

For this assignment you will write a Java application that simulates a coin counter. A coin counter is a device that receives coins in a hopper and sorts the coins by value. As it sorts the coins it also counts them, coming up with a dollar total for the coins that were processed through the hopper. For this assignment you will simulate the hopper and also simulate the counter.

The hopper holds a bunch of coins that have been loaded by placing the coins in the hopper in what amounts to a pile of coins. The coins fall randomly onto a spinning platter that sends each coin individually down a ramp. The ramp has holes in it so that only a correct sized coin can fall through the hole of the ramp. The holes are ordered from the smallest coin to the largest coin in terms of their diameter. When the coin falls through the hole it passes through a short column that drops the coin into a bin for that denomination of coin. As the coin passes through the column the count of that particular denomination of coin is incremented. When all of the coins in the hopper has been emptied, the coin counter produces as output the total count of each denomination of coin and the total dollar value of all coins that were counted. The output looks something like the sample shown below.

|  |  |
| --- | --- |
| Coin Counter Totals | |
| 1$ | 30 |
| .50 | 10 |
| .25 | 43 |
| .10 | 98 |
| .05 | 22 |
| .01 | 74 |
| **Total** | **$57.39** |

The diagram below depicts a coin sorting mechanism similar to the one I’ve described, albeit somewhat simpler.



You are going to implement the simulation as two distinct processes. One of these processes will produce a random denomination coin the other process will consume the random denomination coin. We will use an advanced Java class for this called a synchronous queue.

The process that produces the random coin is called a producer.

The process that consumes(uses) the random coin is called a consumer.

The producer and the consumer will use a common GUI-type interface. The interface will be provided along with the code for consuming and producing. The GUI is shown below.



As the coins are produced the counts of each type of coin is updated. This occurs on a coin-by-coin basis as each coin “drops” into the coin counter. In addition to the coin counts, the total dollar amount of the coins is updated as coins are counted.

There are six different kinds of coin, but with respect to the program the behavior of each coin is the same. Therefore, this is a perfect application of inheritance.

Assignment Part 1

First outline a general method that performs the necessary functions for coin processing. Each coin type requires a certain similar set of functions. Define the functions that are needed for processing coins in the coin counter.

Write 6 individual methods, one for each coin type. Each of the methods should perform the functions needed for the coins being processed by the coin counter.

Look at each of the methods you have defined. Identify the aspects of all 6 methods that are the same among all 6 methods. For example there may be a multiplication that is the same in all of the methods.

Create a superclass named Coin. Coin contains the common aspects of all of the individual coin methods.

Next define each of the coin methods as a subclass of the superclass Coin. There should be 6 subclasses each of which inherits the common aspects from Coin but implements/overrides those aspects that differ.

Write a test class that tests the Coin superclass and the coin subclass methods.

Show that your Coin class/subclasses work as necessary.

Assignment Part 2

Once you have the Coin class/subclasses finished, the next step will be to connect the classes you have written to the GUI and the consumer/producer classes. The necessary classes have been posted to D2L. The consumer/producer is set to produce random coins when you press the start button and will stop when you press the stop button. The coins are generated as instances of the class Integer. The numbers generated are associated to individual coins as follows.

1 = penny

5 = nickel

10 = dime

25 = quarter

50 = half dollar

100 = dollar

**THIS ASSIGNMENT IS DUE ON MARCH 24th, 2015 at 5:00 PM.**