CS 1301 - Homework 03

Due before September 12, 2pm on Vista (WebCT). You may only submit your work once.

1. **CashReg.java** – There are three types of customers: regular, government, and special. Government customers get a 10% discount and are not charged sales tax while special customers get a 7% discount. There are also two age groups: senior citizens and not seniors. Seniors also get a 10% discount. However, in no case, for any person, can the total discount be more than 15%. Tax (7%) is charged on all orders except government orders.

   Write a program that reads a customer type: (r,g, or s), an age type (s or n), and an order amount (*e.g. 123.45*) and computes the order amount (the value inputted), the discount amount, the subtotal (order amount – discount), the tax amount, and the total order cost (subtotal + tax).

   Hint: Do some analysis. Draw out, in some notation, all the cases.

   The Scanner class will not read *char*, but it will read *String*. To allow the user to enter one of the codes above, you will read it as a string, and then convert it to a character.

```
String str = s.next();
char choice = str.charAt(0);
```

2. **Sort3Nums.java** – Write a program that reads three decimal numbers from the user and displays them in ascending order.

   Hints:

   There are a number of ways you can write this and most of them will be acceptable. The real key to any of the approaches is to think about how many possible orderings of 3 numbers that there are. The answer is 6=3*2*1. This is called a *permutation of 3 objects*. If we think about picking the position of the smallest number, middle, then largest number we can see that there are 3 positions the smallest number can be in. Once you have picked a position for the smallest number, then there are two ways to pick the position of the middle number, finally, there is one way to pick the position of the largest number (the only one remaining) once the smallest and middle have already been picked. Thus, there are 3*2*1 = 6 possible orderings.

   Write these 6 possibilities down. You could use notation like this: 2,1,3 to represent the smallest is in position 2, the middle in position 1, and the largest in position 3. Now, what are the rest of them?

   Now, the key to writing the most efficient algorithm is to think about how you could write a *nested if* for these possibilities, without *else if*'s nor && nor ||. However, as stated above, there are numerous other ways to do it.

3. **Lottery.java** – The program will prompt the user to enter three digits (0-9) which we will call the *lottery*. Then, the user will be prompted to enter 3 more digits which we will call a *guess*. Evaluate the guess and print the appropriate message:

   match all 3 digits in correct order – "win $1000"
   match all 3 digits – "win $500"
   match 2 of 3 digits – "win $200"

4. **Calc.java** –Write a program that reads two decimal numbers and an operator (+,-,*,/,%) and computes the correct result of the operation. Round all numbers to 3 decimals. For example, if the user enters: 2.5, then 2, then "*", then the output should look like this:

   2.500 * 2.000 = 5.000

   See the hint from problem 2 to read in a *char*.

5. **Validate.java** –Write a program that prompts the user for number (can be decimal or integer). However, the number is required to be an odd integer (treat 3.0 as an integer), bigger than 0 and less than 1000, and divisible by either 3, 5, or 11. If the number is valid, then print a message: "Number supplied is valid". Otherwise, print a message that details **a** problem with the input. For example: "864.3 is not an integer", or "22 is not odd". In other words, you don't have to detail every problem with the number, just one).