COP 3330 Suggested Exercises for Week 6

Classes, Method Overloading

1) The add method in the fraction class is overloaded, but the subtraction, multiplication and division methods are not. Write second versions of these methods that each take in a single integer to subtract, multiply and divide the current integer by, respectively. Test your methods to make sure they work!

2) Create a Money class (very similar to a GiftCard) but have it store two integers, dollars and cents. Make the class so that a Money object is immutable, like Java's String class. Create suitable methods to add and subtract two money objects.

3) Overload the add method of the money class to take in an integer (which is assumed to be a number of dollars). Add some tests of this method.

4) Overload the add method a second time with a method that takes in a double. Parse out the integer portion of the double (you can add 1e-9 to it and cast to an int to do this) and treat this as the number of dollars, then subtract this integer from the original double, again add 1e-9, take that and multiply by 100, and cast that to an integer to extract the number of cents. (This is doing truncation instead of rounding. To round, you would add .5 after multiplying by 100.) Test your overloaded method.

More Classes

5) Rewrite the matrix code from the last set of practice problems in an object oriented fashion. Create a SquareMatrix class which stores n by n matrices and has addition, subtraction and multiplication defined. When defining these methods, return null if the two objects (this and the formal parameter) don't have equal dimentions. Return the appropriate result otherwise.

6) Create a Die class that manages a single n-sided die with labels 1 through n. This class should have three instance variables: the value of n, a Random object and the current value showing on the die. The only two methods this class needs are a roll() method and a method called currentValue() that returns the current value on the Die object. Then, create a Dice class that has as its instance variables the number of Die objects and an array of those Die objects. In this class, create a roll method which returns an array of integers showing the numbers of all of the dice on a simulated roll. Write another method which returns true if each Die object shows the same value currently. Add any other methods you see fit.

7) Write a Complex Number class with two doubles storing the real and imaginary components of a Complex Number. Write the usual arithmetic methods (addition, subtraction, multiplication, division) and any other methods you feel would be useful for a user.

8) Create your own MathVector2D class to store a two-dimensional vector (also two doubles, one for the x direction and one for the y direction). This class should have standard vector methods such as returning the magnitude of a vector, computing the dot product of two vectors and computing the cross product of two vectors.

9) Write a class to store a standard playing card. There are many ways to store the instance variables for this, choose a technique you feel is appropriate. Feel free to add arrays that belong to the class (static) that help various parts of the implementation and display.

10) Use the class written in exercise 9 to write a simple card game such as Blackjack. Feel free to make up your own game or pick another well-known game to implement.