COP 3330 Suggested Exercises for Week 7

Inheritance

1) Add a distance method in the Coordinate class. Define the distance between two Coordinates to be the sum of the absolute value of the difference in the numbers and the absolute value of the of the difference of the Ascii values of the characters of the two coordinates. The method signature should be:

```
public int distance(Coordinate other);
```

Test this method from a different class by calling it with both Coordinate objects and ColorCoordinate objects. For example, the distance between (7, 'J') and (3, 'L') should be |7 – 3| + |74 – 76| = 6. (The Ascii value of 'J' is 74 and the Ascii value of 'L' is 76.)

2) Add a method to the ColorCoordinate class called distance that takes in a ColorCoordinate object. In this method, call the corresponding super class method and add to it the absolute value of the difference of lengths between the two colors. For example, the distance between (7, 'J', "green") and (3, 'L', "purple") should be 7 (6 from the previous example in question 1 plus the value of |5 – 6| = 1, the difference of the lengths of "green" and "purple".) Write test code to see differences in how the same exact main calls work differently when this method is added to your code (assuming you did exercise #1).

3) Create a Point2D class to maintain a point/vector in the Cartesian plane. (This class should have two instance variables: x and y.) Create a distSq method that returns the distance squared between this Point2D and a formal parameter Point2D. Add any other methods you would like to add. Then, create a Point3D (added instance variable z). Write a new distSq method to this class that takes in a Point3D and calls super to aid in its task.

4) Add to the MixedFraction class so that it has methods to subtract, multiply and divide mixed fractions. Test these methods out as you see fit to make sure that you've written them correctly.

Pac Man Edits

5) Add functionality to the PacMan class so that it's possible to set the direction of movement to an absolute direction instead of modifying this direction of movement relatively. (Play around with the code a bit and you'll see what this means.)

6) Add functionality so that the all of the fruit move in a predictable pattern. For example, if you are generating 10 fruits, generate them at (0, 10), (1, -10), (2, 10), (3, -10), ..., (9, -10) and then have the fruits go up and down one step at a time, bouncing in between y=-10 and y=10, turning back up when hitting y=-10 and turning back down when hitting y=10.

7) Add a Ghost class to the Pac Man example where the ghost eats Pacman if a Ghost object occupies the same square as Pacman at the same time.

Polymorphism

8) Edit either of the examples with classes A, B and C to add or delete methods called f or g and see if you can predict what the program (with the same main but different methods) will print out!

9) Classic textbook examples of polymorphism use examples of animals and a speak() method. Create your own example from this idea testing out different combinations of speak methods (more interesting if these methods take in parameters) in a set of classes that has inheritance structure.

10) Come up with an example where removing a method prevents the code from compiling even though there is technically a method left your code that matches the method signature of a method call. See if it's possible to come up with a similar example when adding a method.