CC3 – Object Oriented Programming

Laboratory Exercise #5
Inheritance and Polymorphism

Name: _____ Date: _____

Code/Schedule: _____ Terminal #: _____

Topic(s) Covered: Inheritance, Polymorphism

Estimated Completion Time: 2 meetings

Objectives:

1. To write a class that extends a super class and implement inheritance.
2. To override certain methods and apply polymorphism.
3. To instantiate objects from the subclasses that will utilize instance variables and methods declared in the super class.

Activity:

1. In the RPG project from the previous lab exercises, subclasses will be created such as Wizard and your own character. Another test class, CharacterDuel, will be written to implement objects of the subclasses.

2. Complete the Wizard.java template (see page 2), which extends Character to add magic points and spellcasting. You should have two new instance variables: maxmagic, currentmagic, and two new methods: castLightningBolt() and castHeal().
Some implementation is up to you to decide, although the castLightningBolt() should be modified by the character's intelligence in some way, and should decrease currentmagic by 5 points.

3. Using Wizard.java template (see page 2) as a guide, create a new .java file which extends Character, and contains at least 1 new method, which uses dexterity as an attack modifier similar to attack() in Character and castLightningBolt() in Wizard. Be creative!

4. Save as your TestCharacter.java and name it CharacterDuel.java. Modify the file to see your new characters fight to death using the subclass objects and behaviors.

Wizard.java

```
/**
 * The Wizard class extends Character, adding two new instance
 * variables: maxmagic, and currentmagic, as well as the
 * castLightningBolt() and castHeal() methods, which represent
 * casting magic spells.
 */

public class Wizard extends Character {
      /** instance variables */

      /**
       * The Wizard(String,int,int,int) constructor overrides the
       * constructor with the same signature in Character. It first
       * calls that constructor using the super keyword, then
       * initializes maxmagic to a value, and sets currentmagic to
       * the same value, similar to lifepoints.
       */
      public Wizard(String n, int s, int d, int i) {
      }

      /**
       * castLightningBolt() represents casting a magic spell. The
       * method first checks that currentmagic is greater than/equal
       * to 5. If it is, currentmagic is reduced by 5 and a random
       * number is returned using the dice inherited from Character,
       * modified by the character's intelligence.
       * Otherwise, returns 0.
       */
      public int castLightningBolt() {
      }

      /**
      * castHeal() represents casting a magic spell. The method first
      * checks that currentmagic is greater than/equal to 8. If it
      * is, currentmagic is reduced by 8 and the character's heal()
      * method is called with a random number, modified by the
      * character's intelligence score. The amount healed is also
      * returned by the method.
      */
      public int castHeal() {
      }

      /**
       * Returns maxmagic
       */
      public int getMaxMagic() {
      }

      /**
       * Returns currentmagic
       */
      public int getCurrentMagic() {
      }
}
```

Laboratory Exercise Score Sheet

Trait	Exceptional (4)	Acceptable (3)	Amateur (2)	Unsatisfactory (1)
Specifications	The program works and meets all of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
Readability	The code is exceptionally well organized and very easy to follow.	The code is fairly easy to read.	The code is readable only by someone who knows what it is supposed to be doing.	The code is poorly organized and very difficult to read.
Reusability	The code could be reused as a whole or each routine could be reused.	Most of the code could be reused in other programs.	Some parts of the code could be reused in other programs.	The code is not organized for reusability.
Documentation	The documentation is well written and clearly explains what the code is accomplishing and how.	The documentation consists of embedded comment and some simple header documentation that is somewhat useful in understanding the code.	The documentation is simply comments embedded in the code with some simple header comments separating routines.	The documentation is simply comments embedded in the code and does not help the reader understand the code.
Delivery	The program was delivered on time.	The program was delivered within a week of the due date.	The code was within 2 weeks of the due date.	The code was more than 2 weeks overdue.
Efficiency	The code is extremely efficient without sacrificing readability and understanding.	The code is fairly efficient without sacrificing readability and understanding.	The code is brute force and unnecessarily long.	The code is huge and appears to be patched together.