Final Exam Review
COSC 117: Programming Fundamentals
Dr. Joseph Anderson

1. All labs, exam reviews, and projects.

2. Modify the following class so that the two instance variables are **private** and there is a getter method and a setter method for each instance variable:

   ```java
   class Player{
       String name;
       int score;
   }
   ```

3. Explain what is meant by the terms subclass and superclass.

4. What is the meaning of the keyword **this**?

5. What is the meaning of the phrase “reference to an object”? How does this affect programming in Java, particularly when you are passing variables as function parameters.

6. Explain the terms “getter” and “setter” in object-oriented programming, with an example of each.

7. Write a class that has both a **default** and **non-default** constructor, then show how to use each one from another part of the program.

8. Write the definition of a class called Pet with a public attribute called name. Write the definition for a second class called Dog that is a subclass of Pet and has its own public attribute called breed.

9. Explain the term **polymorphism**.

10. What is the meaning of the **super** keyword? When might it be used?

11. Write a class called Time with two private integer attributes: hours and minutes. An object of type Time represents a time of day in a 24-hour system. The hours attribute should take values from 0 to 23 and the minutes attribute should take values from 0 to 59. Write the following methods for the class:

    (a) A default constructor which sets both attributes to zero.

    (b) A non-default constructor to take two arguments: one for each attribute.

    (c) A **void** method called addHours that takes a single integer argument to increase the number of hours. Take into account that the number of hours should not end up larger than 23.

    (d) A **void** method called addMinutes that takes a single integer argument to increase the number of minutes. Take into account that the number of minutes should not be larger than 59 and that, if it were to end up more than 59, the number of hours should increase accordingly. E.g. if hours is 3 and minutes is 45 and you call addMinutes(20), the Time object should have hours of 4 and minutes of 5.
(e) A `void` method called `addTime` that takes a single `Time` object as an argument and adds the two `Time` objects together, taking into account the restrictions above.

(f) A method called `toString` that returns a `String` object representing the time in the form of "Hours:Minutes". E.g. if a `Time` object has `hours` of 2 and `minutes` of 45, the `toString` method should return the string "2:45".