CSCI 1581 Laboratory 8
Classes and Objects Part I

Jonathan Redmann

Introduction

In this lab we will learn to work with classes and objects. A class can be thought of as many different things, but one of the most important ways to think of classes is as programmer defined data-types. You have seen several examples of classes and objects already like class Random and class Scanner. To see this in action we will be creating a class Fraction.

Class Declaration

1. Open Gedit and type in the class declaration.

```
public class Fraction
{


}
```

Instance Variables

Instance variables are the data that "make-up" the programmer defined type. Remember that every data-type tells the computer what kind of information is being stored and how to store it. In a class we do this with instance variables.

2. Add instance variables that allow us to model a fraction. Think about what a fraction is made up of: a numerator and a denominator. Those two numbers are usually integers, so what instance variables would we need for this? Create them.

Constructors

The constructor of a class is the method that actually creates or ***constructs*** the object of that class. This is mainly about given the instance variables of the class initial values, but it can do other things.

3. Write the constructor for class Fraction.

```
public Fraction( /*arguments go here*/ )
{
     //method body goes here
}
```

Notice that the constructor does not have a return type.

The this Keyword

When defining a class we will need to be able to clearly distinguish between the object of the class that is calling the method of the class and other objects of that class that the method is using. To do this we use the **this** keyword.

Methods

Next we will need methods associated with the objects. Since a class defines a data-type, we can think of the methods of the class as being the operations for that data-type. If we are making a class for fractions what kind of operations would we want to be able to do with freactions?

4. Write an addition method for class Fraction

```
public Fraction add( Fraction otherFraction )
{
     // Implement fraction addition here

}
```

Notice that the return type of the add method is Fraction. When we add to fractions with pencil and paper we expect to get another fraction, so our add method should return a new fraction that is the sum of the fraction object that is calling the add method and the fraction object that is being passed to the method as an argument.

5. Write a subtract method in exactly the same way as the add method except that you are now subtracting the two fractions.

6. Write a multiplication method. Remember that for fraction multiplication we just multiply the numerators together to get the resulting fraction's

numerator and multiply the denominators together to get the resulting fraction's denominator.

7. Now write a division method. For fraction division we just invert the divisor and multiply the two fractions the same as we did in step 6.

Hint: Use the multiplication method you wrote in step 6 to simplify what you have to write.

8. Write a toString() method that prints the fraction object to the screen. It should look like this:

```
> 5 / 6
```

or

```
> 21 / 107
```

and so on.

9. Add, commit , and push your fraction class.

10. To test our fraction class we now need to write a small progam that uses it.

11. Open a new file in Gedit and type in the "boiler-plate code"

12. In the main method create two references of your class Fraction.

13. Write a simple menu that allows the user to perform one of the four operations you wrote methods for: add, subtract, multiply, and divide.

14. Then ask the user for the numerator and denominator of each of the two fractions. Use the users input to initialize the references you created in step 12 to objects of class Fraction.

15. Perform the operation selected by the user.

16. Use the toString method to print the results

17. Add commit and push your program to test class Fraction.

18. Compile and run the program and test it!