HOMEWORK PROBLEMS 6

1) What is the scope of a private static variable?

2) What two items are passed to `f` in the following call:

```
r.f(23);
```

3) Under what circumstances, if any, can a static method call an instance method?

4) Why should a named constant that is not inside a method be static?

5) When do class variables come into existence? When do instance variables come into existence?

6) What are the errors in the class below. Compile the class to check your answers.

```
class C6h6
{
   private int x;
   //---------------------------------
   public void f()
   {
      int y;
      System.out.println(x);
      System.out.println(y);
      static int z = 3;
      System.out.println(z);
   }
}
```

7) Write a class that contains

```
private final int x = 10;
```

and the method

```
public void f()
{
   x = 5;
}
```

What happens when you compile the class? Explain.

8) Compute and run the `C6h8` program below. It displays the default values for instance variables of various types. Run the program to determine the default value corresponding to each type.

```
class C6h8
 {
    public static void main(String[] args)
    {
        DefaultValues v = new DefaultValues();
        v.displayVariables();
     }
 }
//=======================================
```

```
class DefaultValues
{
   private byte b;
   private short s;
   private int i;
   private long l;
   private float f;
   private double d;
   private char c;
   private boolean boo;
   //-------------------------
   public void displayVariables()
   {
      System.out.println(b);
      System.out.println(s);
      System.out.println(i);
      System.out.println(l);
      System.out.println(f);
      System.out.println(d);
      System.out.println(c);
      System.out.println(boo);
   }
}
```

9) Add the following constructor to your `DefaultValues` class from homework problem 8:

```
public DefaultValues(int x)
{
   i = x;
}
```

Make no other changes. Why does your new program fail? Fix the problem.

10) Compile the following class. What happens? What is wrong with the program?

```
class C6h10
{
   private int x;
   public static void main(String[] args)
   {
       x = 1;
       g();
   }
   //---------------------------------
   public void g()
   {
      System.out.println(x);
   }
}
```

11) Does the program below work? Compare it with the program in homework problem 10.

```
class C6h11
{
   private int x;
   public static void main(String[] args)
   {
       C6h11 r = new C6h11();
       r.x = 1;
       r.g();
   }
   //---------------------------------
   public void g()
   {
      System.out.println(x);
   }
}
```

12) Does the program below work? Compare its structure with the program in homework problem 11.

```
   class C6h12
   {
      public static void main(String[] args)
      {
          Variation r = new Variation();
          r.x = 1;
          r.g();
      }
   }
   //================================================
   class Variation
   {
      private int x;
      public void g()
      {
         System.out.println(x);
      }
   }
```

13) Write a class that contains an instance variable x. This class should also contain a method that has a parameter x. Does the class compile without error? When the method is executing, two x variables exist: the instance variable and the parameter. Your method should assign 99 to x. Which x—the instance variable or the local variable—is assigned 99? Check your answer by adding a second method that displays the instance variable x. Use it to determine if the instance variable is assigned 99 by the first method.

14) The `set` method in the program below assigns 5 to the local variable `x` declared within `set`. Run this program. The call of `display` will display 1, indicating that the assignment to `x` within `set` was to the local `x`, not to the instance variable `x`.

```
class C6h14
{
   public static void main(String[] args)
   {
      WhichX r = new WhichX(1);
      r.display();
      r.set();
      r.display();
   }
}
//================================================
class WhichX
{
   private int x;
   //---------------------------------
   public WhichX(int a)
   {
      x = a;
   }
   //---------------------------------
   public void set()
   {
      int x;
      x = 5;      // assigns 5 to local x not instance variable x
   }
   //---------------------------------
   public void display()
   {
      System.out.println(x);  // displays instance variable x
   }
}
```

Now change the assignment statement in the `set` method to

```
this.x = 5;
```

Compile and run. Which `x` is assigned 5 now? Whenever we qualify a variable with `this` in an instance method, the variable is treated as an instance variable in the object, even if there is a local variable or parameter with the same name. In the `set` method above,

```
x = 5;
```

assigns 5 to the local `x`. But if we change this statement to

```
this.x = 5;
```

it would then assign 5 to the instance variable `x`. It is convenient to name a parameter used to initialize an instance variable with the same name as the instance variable. If we do this, we have to use `this` to distinguish between the instance variable and the parameter. For example, we can replace the constructor in the program above with the following:

```
public WhichX(int x)   // parameter has same name as variable
{
   this.x = x;         // x is the parameter, this.x is the instance variable
}
```

Here both the parameter and the instance variable are named `x`. Within the constructor, we refer to the instance variable using `this.x`, and we refer to the parameter using `x` alone. Thus, the statement

```
this.x = x;
```

assigns the parameter `x` to the instance variable `x`.

15) Move the `main` method in the `TestOOP1` class in Fig. 6.1 into the `OOP1` class. Delete the `TestOOP1` class. Now the complete program is in the `OOP1` class. Rename `OOP1.java`. Enter

```
javac OOP1.java
java OOP1
```

Does the new program work the same way the original program works? In the new program, only `main` and `x` are initially available because they are static. `main` creates the `n` and `m` objects as in the original program and then proceeds as in the original program. The only difference is that `main` is now in the `OOP1` class.

16) Create a class named `Point`. It should have instance variables `x` and `y` of type `double` whose values define the location of a point on the x-y plane. Include a constructor with no parameters that assigns 0.0 to `x` and `y`. Include a second constructor with two parameters that sets the values of `x` and `y` to whatever values you pass the constructor. Your class should have a method named `distanceFromOrigin` that returns the distance of the point defined by `x` and `y` from the origin (i.e., the point (0.0, 0.0)). Your class should also have a second method `toString` that returns a string representing the values of `x` and `y`. For example, if `x` and `y` are both 1.0, then `toString` should return the string `"(1.0, 1.0)"`. Test your program with the following main method:

```
public static void main(String[] args)
{
   Point p = new Point();
   System.out.println(p.toString + " is this far from origin: " +
                      p.distanceFromOrigin());
   p = new Point(3, 4);
   System.out.println(p.toString() + " is this far from origin: " +
                      p.distanceFromOrigin());
}
```

Hint: the distance of the point (x, y) from (0.0, 0.0) is the square root of the sum of the squares of x and y. To determine the square root of a double value, call the method `Math.sqrt`, passing it the value for which you need the square root. This method will then return the square root as a `double` value. For example, to find the square root of the value in `sum` and assign it to `d`, use

```
d = Math.sqrt(sum);
```

`sqrt` is a static method in the `Math` class. Thus, it is called via the name of its class.

17) Add a method `distance` to your program for homework problem 16. The `distance` method should have one parameter of type `Point`. It should return the distance between the point in the object containing the called method and the point it is passed. For example, in the following code, `distance` returns the distance between the `p1` and `p2` points:

```
Point p1 = new Point(1.0, -2.0);
Point p2 = new Point(5.0, 6,0);
System.out.println(p1.distance(p2));
```

Write a program that tests your new `Point` class. Use the `distance` method to determine the distance between the point (1.0, -2.0) and (5.0, 6.0). Hint: To compute the distance between two points, square the difference of the two `x` values, square the difference of the two `y` values, add these two squares, and take the square root of the result.

18) Draw a picture of the structure the following program creates. What will eventually happen when it is executed?

```
class C6h18
{
   public static void main(String[] args)
   {
      long i = 1;
      Node first = null, p;
      while (true)
      {
          p = new Node();
          p.count = i++;
          p.link = first;
          first = p;
      }
   }
}
//===============================================
class Node
{
    public Node link;
    public long count;
}
```

Change the `while` loop in the `C6h18` class to

```
while(true)
{
   p = new Node();
   p.count = i++;
}
```

What impact does this change have on the ultimate behavior of the program?

19) A **rational number** is any number that can be expressed as the ratio of two integers. A rational number can be represented by two integers: the numerator and the dominator. Create a `RationalNumber` class according the following specifications:

Data fields: `int numerator`, `int denominator`

Constructor: `public RationalNumber(int n, int d)`

Instance Methods:

`public RationalNumber add(RationalNumber r)`
Adds the rational number in the object containing the called method and the rational number in the `r` object. Returns a new `RationalNumber` object that contains the sum.

`public RationalNumber multiply(RationalNumber r)`
Multiplies the rational number in the object containing the called method and the rational number in the `r` object. Returns a new `RationalNumber` object that contains the product.

`public void reduce()`
Reduces the rational number to lowest terms. Does not create a new object.

`public String toString()`
Returns a string represesenting the value of the rational number. The string returned should include the slash character between the numerator and denominator. For example, if `numerator` and `denominator` are 5 and 7, respectively, then `toString` returns the string `"5/7".`

Write a program that uses your `RationalNumber` class. Your program should add 3/7 and 7/3 and display the result. It should also multiply 3/7 and 7/3, put the result in reduced form, and display the result.