HOMEWORK PROBLEMS 7

1) How do instance variables differ from local variables with respect to default values?

2) Can an instance variable be initialized in its declaration?

3) Suppose `x` in a local variable with type `int`. Can the following call affect the value of `x`? Explain.

```
f(x);
```

4) Suppose an object pointed to by `p` contains an object pointed to by `q` (an instance variable in the `p` object). Suppose the `q` object contains an `x` instance variable. Assuming everything is public, what statement in `main` will display the value of `x`. Assume `main` has access to the reference variable `p`.

5) Why does line 14 in Fig. 7.10a not invoke the copy constructor for `String`?

6) What is displayed when the following program is executed. Determine your answer first by inspection. Then check your answer by running the program.

```
class C7h6
{
    public static void main(String[] args)
    {
       int i = 1;
       Counter r = new Counter();
       while (i++ <= 100)
          r = new Counter();
       r.display();
   }
}
//================================================
class Counter
{
   private static int count1;
   private int count2;
   //----------------------------------
   public Counter()
   {
      count1++;   // add 1 to count1
      count2++;   // add 1 to count2
   }
   //----------------------------------
   public display()
   {
      System.out.println("count1 = " + count1);
      System.out.println("count2 = " + count2);
   }
}
```

7) What is wrong with the following constructor:

```
public C7h7()
{
   C7h7 r;
   r = new C7h7();
}
```

Run a test program to check your answer.

8) Is it legal to initialize a static variable in a constructor for a class? Run a test program to check your answer. Why would it generally not be a good idea to initialize static variables in a constructor?

9) Create a `MyRectangle` class according to these specifications (`x` are `y` are the coordinates of the upper left corner of the rectangle):

Fields:

```
private double x;
private double y;
private double width;
private double height;
```

Constructors

```
public MyRectangle(double newX, double newY,
       double newWidth, double newHeight)
```

Assigns `newX`, `newY`, `newWidth`, and `newHeight` to `x`, `y`, `width`, and `height` respectively.

```
public MyRectange(MyRectangle r)
```

Copy constructor

Instances Methods:

```
public void set(double newX, double newY,
                double newWidth, double newHeight)
```

Assigns `newX`, `newY`, `newWidth`, and `newHeight` to `x`, `y`, `width`, and `height` respectively.

```
public String toString()
```

Returns a string with the values of `x`, `y`, `width`, `height` in the following format:

```
x = ___   y = ___   width = ___   height = ___
```

```
public double area()
```

Returns the area of the rectangle.

```
public void move(double xChange, double yChange)
```

Adds `xChange` to `x`, and adds `yChange` to `y`

Also write a class `TestMyRectangle` that tests your `MyRectangle` class. The `main` method of `TestMyRectangle` should

1) Create an object from your `MyRectangle` class in which all the fields are initially zero.
2) Display the initial values of `x`, `y`, `width`, and `heigth`.
3) Set `x` to 2.0, `y` to 3.0, `width` to 4.5, `height` to 5.1.
4) Display `x`, `y`, `width`, `height`.

5) Display the area.
6) Move the rectangle to the right 2.5 units, down 3.0 units.
7) Display `x`, `y`, `width`, `height`.
8) Display the area.
9) Create a second `MyRectangle` object by calling the copy constructor
10) Display `x`, `y`, `width`, `height` of the new object.

10) Write a `Clock` class according to the following specifications:

Fields:

```
private int seconds;
private int minutes;
private int hours;
```

Constructors:

```
public Clock(int s, int m, int h)
```

Initializes the instance variables `seconds`, `minutes` and `hours` to the parameters `s`, `m`, and `h`, respectively.

```
public Clock(Clock r)
```

Copy constructor

Instance Methods:

```
public void setTime(int seconds, int minutes, int hours)
```

Sets the instance variables `seconds`, `minutes` and `hours` to the parameters `s`, `m`, and `h`, respectively.

```
public void tick()
```

Adds 1 to `seconds`. `seconds` wraps around at 60 seconds to 0, in which case `tick` adds 1 to `minutes`. `minutes` similarly wraps around at 60 minutes to 0, in which case `tick` adds 1 to `hours`. `hours` similarly wraps around at 24 hours to 0.

```
public String toString()
```

returns a string containing the values of `seconds`, `minutes`, and `hours` in the following format:

```
seconds = ___ minutes = ___hours   = ___
```

```
private void incrementMinutes()
```

Adds 1 to `minutes`. `minutes` and `hours` wrap around at 60 and 24, respectively.

```
private void incrementHours()
```

Add 1 to `hours`. `hours` wraps around at 24 hours to 0.

Use named constants to hold the wrap-around values for seconds, minutes, and hours (60, 60, and 24). Write a test program that constructs a `Clock` object, sets `seconds`, `minutes`, and `hours` to 58, 59, and 23, respectively. Then call `tick` four times. Display the time after each call of `tick`. Then create a copy with the copy constructor. Display the time in the new clock.

11) Same as homework problem 10 but add a `changeMode` method that changes the mode: 12 hour to 24 hour or 24 hour to 12 hour. In either the 24 or 12-hour mode, the clock should record hours as a number from 0 to 23.

However, in the 12-hour mode, the `toString` method should return the time as a 12-hour time, labeled with `a.m.`, `p.m.`, `noon`, or `midnight`, as required. Initially, a `Clock` object should be in the 24-hour mode. Write a test program that checks all the features of your enhanced clock. For example, make sure it correctly progresses from a.m. to noon to p.m., and from p.m. to midnight to a.m. when in either 12-hour or 24-hour mode.

12) Create classes `C1`, `C2`, and `C3` as follows:

    `C1` has an instance variable `r1` whose type is `C2`.
    `C2` has an instance variable `r2` whose type is `C3`
    `C3` has and instance variable `x` whose type is `int`.

    Each class should have a parameterless constructor. The `C1` constructor should create a `C2` object and assign `r1` its reference. Similarly, the `C2` constructor should create a `C3` object and assign `r2` its reference. The `C3` constructor should initialize `x` to 7. Each class should also have a copy constructor. Your `main` method should create a `C1` object and then create a copy using the copy constructor for `C1`. `main` should then change the value of `x` associated with the first `C1` object from 7 to 11 (add `set` methods to your classes to allow this). It should then display the `x` value associated with both `C1` objects.

13) Create a `MinimalChange` class that has two instance methods: `deposit` and `change`. `deposit` accepts and accumulate deposits. Amounts are passed to the `deposit` method in units of cents. For example, to deposit $5.25, you would pass the `deposit` method 525. `change` returns in string form the current balance using the *minimum* pieces of currency. For example, if the current balance $26.61, then `change` would return one twenty-dollar bill, one five-dollar bill, one one-dollar bill, two quarters, one dime, and one penny. `change` also sets the deposit amount to zero. Assume the `change` method has an unlimited supply of pennies, nickels, dimes, quarters, one, five, ten, and twenty-dollar bills. Test `change` on the following balances: $0.00, $0.01, $0.09, $0.10, $0.57, $4.99, $31.33, and $1234.56.

14) Create an `ExamResults` class that has four instance methods: `record`, `average`, `max`, `min`, and `count`. `record` enters a grade (an integer between 0 and 100). `average`, `max`, `min`, and `count` report on the grades entered so far. `average`, `min`, and `count` return the average, minimum, and maximum grades, respectively. `count` returns the number of grades entered. `record` should perform an integrity check on the grade it is passed (it should check that the grade is between 0 and 100). Write a program to test your class. Assume grades are integers. However, `average` should return the average grade as a `double` value.