Introduction to Java
Programs for Packet #4: Classes and Objects

Note. All of these programs involve writing and using more than one class file.

1. Copy the Box class and compile it. But you won't be able to run it because it does not have a main method.

Create a second class file named TestBoxes which has a main method. In the main method do the following:
- create a Box object 2 ft wide by 6 ft long
- call the getArea method and display the result
- call the bigger method and double the dimensions of the box
- call the getPerimeter method and display the result

```
public class Box {
    private double len, width;

    public Box( double a, double b ) {
        len = a;
        width = b;
    }

    public double getArea() {
        return len * width;
    }

    public double getPerimeter () {
        double p = 2.0*(len + width);
        return p;
    }

    public void bigger( double f ){
        len = f * len;
        width = f * width;
    }
}
```

2. Copy the DiceRunner class and complete the Dice class. Compile and run.

```
public class DiceRunner {
    public static void main(String[] args) {
        Dice d = new Dice();
        d.roll();
        int n = d.getTotal();
        System.out.println( n );

        d.roll();
        n = d.getTotal();
        System.out.println( n );
    }
}
```

```
public class Dice {
    private int die1, die2;

    public Dice() {
        Give die1 and die2 random values
    }

    public void roll() {
        Assign die1 a random integer value
between 1 and 6.  Do the same thing for die2.
    }

    public int getTotal() {
        return the sum of die1 and die2
    }
}
```

3. Write another class (with a main method) in the same location as the files you created for Problem 2. In this main method, create a Dice object and roll the dice 100 times (in other words, call the roll method followed by the getTotal method 100 times). Count how many times you roll a 7 or 11. Also count how many times you rolled a 2, 3, or 12. Display the results as percents. For example:

You rolled a 7 or 11	21% of the time.
You rolled a 2, 3, or 12	13% of the time.

Note. Take advantage of the fact that you are rolling the dice 100 times to calculate the percentages.

4. Copy the Circle class and complete the two methods to the right. getC should return the circumference of the circle and changeR should assign the value of the parameter to the radius.

Write a second class with a main method. In the main method:
- create a Circle object with a radius of 3.
- call the getC method and display the returned value (it should be 18.84955592153876)
- call the getArea method and display the returned value (it should be 28.274333882308138)
- call the changeR method and change the circle's radius to 1.
- call the getArea method again and display the returned value (it should be 3.141592653589793)

```
public class Circle {
    private double radius;

    public Circle( double r ) {
        radius = r;
    }

    public double getArea() {
        double a = Math.PI * radius * radius;
        return a;
    }

    public double getC () {
        it returns the circle's circumference
    }

    public void changeR( double r ) {
        changes the value of radius to r
    }
}
```

5. Write the Employee class so that when the main method in RunEmployee is executed, the following is displayed.

```
Salary is 42000.0
Salary is 42000.0
Salary is 45000.0
```

The Employee class has one instance variable for storing an Employee's salary.

Notice that the increase method does NOT change an employee's salary if the argument is negative.

```
public class RunEmployee {
    public static void main(String[] args) {
        Employee e = new Employee( 40000.0 );
        e.increase( 2000.0 );
        double p = e.getPay();
        System.out.println( "Salary is " + p );

        e.increase( -500.0 );
        p = e.getPay();
        System.out.println( "Salary is " + p );

        e.increase( 3000.0 );
        p = e.getPay();
        System.out.println( "Salary is " + p );
    }
}
```

6. Finish the Travel class. Then write a second class that has a main method. In the main method:
- Write a loop that iterates 21 times with the counter going from 6 to 40 in steps of two.
- In the body of the loop, create a Travel object and call each method. Print the returned values.
- The output should look like this:

people = 6, vans = 1, canoes = 2, planes = 1
people = 8, vans = 1, canoes = 3, planes = 1
people = 10, vans = 2, canoes = 4, planes = 1
people = 12, vans = 2, canoes = 4, planes = 1
people = 14, vans = 2, canoes = 5, planes = 2
...
people = 32, vans = 4, canoes = 11, planes = 3
people = 34, vans = 5, canoes = 12, planes = 3
people = 36, vans = 5, canoes = 12, planes = 3
people = 38, vans = 5, canoes = 13, planes = 4
people = 40, vans = 5, canoes = 14, planes = 4

```
public class Travel {
        private int people;

        public Travel( int n ) {
                people = n;
        }

        public int goByVan(){
                this returns the number of vans
required to transport everyone (8 per van)
        }

        public int goByCanoe(){
                this returns the number of
canoes required (3 per canoe)
        }

        public int goByPlane() {
                this returns the number of
planes required (12 per plane)
        }
}
```

7. Copy and complete the Cube method.
Write a second file with a main method that does the following.
- Create a Cube object that has a side of 3.
- Call the volume and surfaceArea methods and print the results.
- Call the setSideForVolume method and use 125 as the argument.
Call the getSide, and surfaceArea methods and display the results.

When your code compiles and runs, the following should print.

```
The surface area is 54.0 sq. units
The volume is 27.0 cu. units
The side is now 4.999999999999999 units
The surface area is 149.99999999999994 sq. units
```

The last two answers are not exact because of floating point errors.

```
public class Cube{
    private double side;

    public Cube( double s ){
        side = s;
    }

    public double volume(){
        return side*side*side;
    }

    public double surfaceArea(){
        returns the cube's surface area
    }

    public double getSide(){
        return side;
    }

    public void setSideForVolume(double v){
        Change side to a value that
corresponds to a volume of v.  For example, if
v is 64 then side should be changed to 4.
    }
}
```

8. Write a Sphere class (similar to the Cube class) so that the class to the right compiles and runs. (SOP is an abbreviation.)

The output should be something like this:

```
The surface area is 50.26548245743669 sq. units
The volume is 33.510321638291124 cu. units
Half the volume is 16.755160819145562 cu. units

The side is now 1.587 units
The surface area is 31.649271471835966 sq. units
The volume is 16.742464608601225 cu. units
```

Notice that some small inaccuracies creep in when working with doubles.

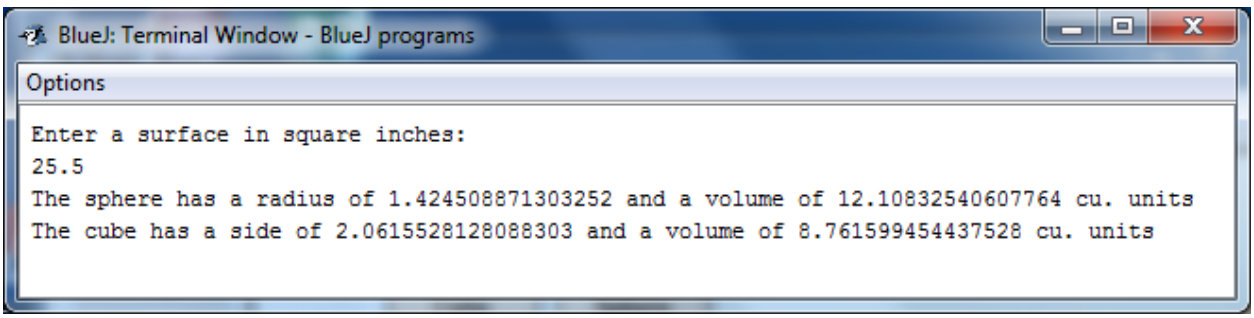
```
public class RunSphere {
    public static void main(String[] args) {
        Sphere x = new Sphere( 2 );
        double vol = x.volume();
        double sa = x.surfaceArea();
        SOP( "The surface area is " + sa + " sq. units");
        SOP( "The volume is " + vol + " cu. units");
        SOP( "Half the volume is " + vol/2 + " cu. units");
        SOP();
        x.setRadiusForVolume( vol/2 );
        double s = x.getRadius();
        vol = x.volume();
        sa = x.surfaceArea();
        SOP( "The side is now " + s + " units" );
        SOP( "The surface area is " + sa + " sq. units");
        SOP( "The volume is " + vol + " cu. units");
    }
}
```

9. Modify the Cube class by adding another method, named setSideForSA, which changes the side based on a given surface area. Modify the Sphere class by adding a method, setRadiusForSA, which changes the radius based on a given surface area. These methods are similar to the setSideForVolume and setRadiusForVolume methods.

Write a another class that has a main method. The first three lines of the main method are:

```
Sphere ball = new Sphere( 1 );
Cube box = new Cube( 1 );
Scanner s = new Scanner( System.in );
```

The rest should ask the user to enter a surface area then change the radius of the Sphere to match that surface area and change the side of the cube to match the surface area. Then call the volume methods and print the results. When you run the program, if the user enters 25.5, the results should look like this:



10. Copy the Car class. In another class's main method, create two car objects.

Write a while loop where the drive method is called for each car and their locations are printed. When a car has gone over 200 miles, the loop stops and that car is the winner. Here is one sample output.

```
Car 1: 4 miles, Car 2: 36 miles.
Car 1: 14 miles, Car 2: 71 miles.
Car 1: 17 miles, Car 2: 75 miles.
Car 1: 21 miles, Car 2: 80 miles.
Car 1: 35 miles, Car 2: 104 miles.
Car 1: 63 miles, Car 2: 115 miles.
Car 1: 91 miles, Car 2: 122 miles.
Car 1: 123 miles, Car 2: 154 miles.
Car 1: 156 miles, Car 2: 178 miles.
Car 1: 167 miles, Car 2: 217 miles.

Car 2 wins
```

But obviously sometimes car 1 will win and there could even be a tie.

```
public class Car {
    private int x;      // location of the car

    public Car( ) {
        x = 0;

    public int getX(){
        return x;
    }

    public void drive() {
        x= x + (int)( 40 * Math.random());
    }
}
```

11. First complete the Number class. Then we play the guessing game. Each turn the user gets to ask if the secret number is a multiple of some number and then they get to guess the number. Here's a rough outline.

```
public class Runner {
    public static void main(String[] args) {
        create a Scanner object
        create a Number object
        boolean keepPlaying = true;
        int count = 0;
        while ( keepPlaying  ) {
            count++;
            Ask the user to enter a positive integer.
Get response and call the multipleOf method.
Print a message that indicates if the secret number
is a multiple of that input or not
            Ask the user to guess what the secret
number is.  Call the guess method.  If the user's
right then print Congratulations, print count, and
set keepPlaying to false (to get out of the loop).  If
the user is wrong, then indicate that.
    }
}
```

```
public class Number {
   private int num;

   public Number( ) {
       num = (int)(100*Math.random() )+1;
   }

   public boolean multipleOf( int x ){
       returns true if num is a multiple of x
   }

   public boolean guess( int x ) {
       returns true if num equals x;
otherwise it returns false
   }
}
```

The following program uses the String class to hold a person's name. We will do a lot of work with the String class in the next packet. This is just a preview.

12. First, complete the play method.

In another class's main method, create two Game objects which each start with 25 tokens. Then keep calling the play method until one of the Game objects run out of tokens.

After the loop call the status method for each Game object.

The main method should only be about 8 lines long.

Note. Over time a player is just as likely to win tokens as to lose tokens so it may take a while before one player loses all their tokens.

Sample output.

```
Paul has a 6 and a 2  Lose 1
Mr. Sawyer has a 4 and a 4  WIN 1
Paul has a 1 and a 6  WIN 2
Mr. Sawyer has a 3 and a 3  WIN 1
....
Paul has a 2 and a 4  Lose 1
Mr. Sawyer has a 2 and a 4  Lose 1
Paul has a 3 and a 1  Lose 1
Mr. Sawyer has a 1 and a 2  Lose 1

Paul has 0 tokens
Mr. Sawyer has 22 tokens
```

```
public class Game{
    private int tokens;
    private String name;

    public Game( int t, String s ){
        tokens = t;
        name = s;
    }

    public void play(){
        If the number of tokens is zero or less, then exit the method immediately

        Generate two random numbers, n1 and n2, between 1 and 6.  Then display them like this:
        SOP*(name+" has a "+n1+" and a "+n2);

If the numbers add up to 7, increase the number of tokens by 2 and print WIN 2.
If the numbers are the same, increase the number of tokens by 1 and print WIN 1.
Otherwise decrease the number of tokens by 1 and print LOSE 1.
    }

    public int tokens() {
        return tokens;
    }

    public void status(){
        SOP*(name+" has "+tokens+" tokens");
    }
}
```

* you need to type System.out.print or println