1 Short Answer/Method Construction (10 Points Each)

1. Write a method that will bring in a string as a parameter and return the count of the number of vowels. The method needs to count vowels that are uppercase or lowercase and you may assume that 'y" is not a vowel.

   Solution:

```
public static int VowelCount(String str) {
    int count = 0;
    str = str.toUpperCase();
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        if (ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U')
            count++;
    }
    return count;
}
```

2. Write a method called `getInteger` that takes in two integer parameters `low` and `high` and continually asks the user for a number until the number is in the range $low \leq n \leq high$. When the number is inside this range the method will return that value. For example, if the method is called with 5 and 10 as low and high a run would look like the following and the method would return 5. You may assume the user always types in a valid integer.

```
Input a number: 1
Input a number: 12
Input a number: 32
Input a number: 5
```

   Solution:

```
public static int getInteger(int low, int high) {
    Scanner kb = new Scanner(System.in);
    int returnValue = 0;
    do {
        System.out.print("Input a number: ");
        returnValue = kb.nextInt();
    } while (returnValue < low || returnValue > high);

    return returnValue;
}
```

3. Write a method that will take in two decimal number as parameters and return the midpoint between the two numbers. So if 7.23 and 21.76 came in as parameters then the value of 14.495 would be returned.

   Solution:

```
public static double midpoint(double a, double b) {
    return (a + b) / 2.0;
}
```

4. Write a method that will bring in three decimal parameters, $x1$, $y1$, and tol (tolerance). The method is to continually generate random values $x2$ and $y2$ both decimal numbers in the range from -100 to 100 and count the number of trials until the distance between the two points (x_1, y_1) and (x_2, y_2) is less than or equal to the tolerance. Recall that the distance between the two points (x_1, y_1) and (x_2, y_2) is calculated as $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

   Solution:

```
public static int trials(double x, double y, double tol) {
    int count = 0;
    double d = 0;

    do {
        double x2 = Math.random() * 200 - 100;
        double y2 = Math.random() * 200 - 100;
        d = Math.sqrt((x - x2) * (x - x2) + (y - y2) * (y - y2));
        count++;
    } while (d > tol);

    return count;
}
```

5. Say we have constructed a class named Triangle with two constructors, one default and one that brings in the lengths of the three sides. Write declarations for a default triangle named `tri1` and a triangle with side lengths 3, 4, and 5 named `tri2`. Also assume that we have two methods in our triangle class `isRight` that returns true or false as to if the triangle is a right triangle or not and `area` that returns the area of the triangle. Write a call (say from the main) that will store the area of triangle `tri2` in a variable named `area` and write an if statement that will test if `tri2` is a right triangle or not and print out an appropriate statement. Do not write any code for the Triangle class, we assume this has already been done.

   Solution:

```
Triangle tri1 = new Traingle();
Triangle tri2 = new Traingle(3, 4, 5);

double area = tri2.area();

if (tri2.isRight()){
    System.out.print("The triangle is a right triangle.");
} else {
    System.out.print("The triangle is not a right triangle.");
}
```

2 Program Traces (10 Points Each)

1. For the given input, write the output of the program.

```
1  import java.util.Scanner;
2
3  public class Exam2_Trace1 {
4
5      public static int DoSomething(int a, int b) {
6          System.out.println("DoSomething int: " + a + " " + b);
7          if (a > b)
8              return a / b;
9          else
10             return a * b;
11     }
12
13     public static int DoSomething(int a, double b) {
14         System.out.println("DoSomething double: " + a + " " + b);
15         int count = 0;
16         while (a > 5) {
17             a -= (int) b;
18             count++;
19             System.out.print(a + " ");
20         }
21         System.out.println();
22
23         return count;
24     }
25
26     public static int DoSomethingElse(int a, int b) {
27         System.out.println("DoSomethingElse: " + a + " " + b);
28         if (a >= b)
29             return DoSomething(b, a);
30         else
31             return DoSomething(3 * b, a * 2.0);
32     }
33
34     public static void main(String[] args) {
35         Scanner kb = new Scanner(System.in);
36         System.out.print("n = ");
37         int n = kb.nextInt();
38         System.out.print("m = ");
39         int m = kb.nextInt();
40
41         int t = DoSomething(m, n);
42         int q = DoSomethingElse(m, n);
43         int r = DoSomethingElse(t, q);
44
45         System.out.println(n + " " + m + " " + t + " " + q + " " + r);
46     }
47 }
```

```
n = 2
m = 6
```

Solution:

```
DoSomething int: 6 2
DoSomethingElse: 6 2
DoSomething int: 2 6
DoSomethingElse: 3 12
DoSomething double: 36 6.0
30 24 18 12 6 0
2 6 3 12 6
```

2. For the given input, write the output of the program.

```java
public class Thing {
    private int x;
    private int y;
    private int z;

    public Thing() {
        x = 5;
        y = 1;
        z = 3;
    }

    public Thing(int a, int b, int c) {
        x = a;
        y = b;
        z = c;
    }

    public void PrintXYZ() {
        System.out.println(x + " " + y + " " + z);
    }

    public int DoSomething(Thing thing1) {
        int a = x - thing1.x;
        int b = y + thing1.y;
        int c = z * thing1.z;
        return a + b - c;
    }

    public int DoSomething(Thing thing1, int t) {
        if (t > 0) {
            while (t > 0) {
                x += t;
                y--;
                z++;
                t--;
                thing1.z += x;
                thing1.x -= y;
                thing1.y++;
                PrintXYZ();
                thing1.PrintXYZ();
            }
        } else {
            for (int i = 0; i < -t; i++) {
                thing1.y++;
                z--;
                x = thing1.x + z;
                PrintXYZ();
                thing1.PrintXYZ();
            }
        }
        return DoSomething(thing1);
    }
}
```

```java
import java.util.Scanner;

public class Exam2_Trace2 {

    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        System.out.print("Input: ");
        int p = kb.nextInt();
        int q = kb.nextInt();
        int r = kb.nextInt();
        int s = kb.nextInt();

        Thing thing1 = new Thing();
        Thing thing2 = new Thing(p, q, r);

        thing1.PrintXYZ();
        thing2.PrintXYZ();

        System.out.println();
        System.out.println(thing2.DoSomething(
            thing1));
        System.out.println();
        System.out.println(thing1.DoSomething(
            thing2, s));

        System.out.println();
        thing1.PrintXYZ();
        thing2.PrintXYZ();
    }
}
```

```
Input: 1 2 3 2
```

Solution:

```
5 1 3
1 2 3

-10

7 0 4
1 3 10
8 -1 5
2 4 18
-81

8 -1 5
2 4 18
```

3 Coding (20 Points Each)

1. Write a program that will ask the user for the number of darts they would like to throw. Have the program simulate the throwing of that many darts on a square ranging from -1 to 1 in both the x and y directions. Have the program find the ratio of the number of darts that within the unit circle (circle radius 1 centered at the origin), and then multiply this ratio by 4 to obtain an approximation to π. Finally have the program output the approximation to π. A couple sample runs are below.

```
Number of darts: 100000
Pi is approximately = 3.14464
```

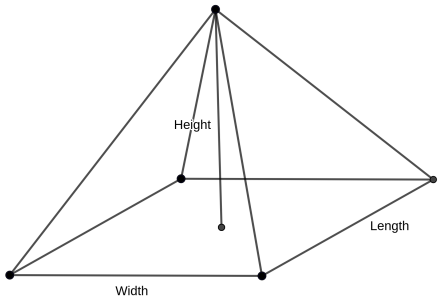
```
Number of darts: 100000000
Pi is approximately = 3.14134584
```

In this program you must write a method `dartToss` that returns a random double between -1 and 1. The main will then use this method to get the values of x and y on each toss. Also write a method called `inside` that will take the x and y values as parameters and return true if the (x, y) point is inside the unit circle and false if not. The main will use this method for determining if the dart hit the dartboard.

Solution:

```
1  import java.util.Scanner;
2
3  public class Exam2Prog1 {
4
5      public static double dartToss() {
6          return Math.random() * 2 - 1;
7      }
8
9      public static boolean inside(double x, double y) {
10         return x * x + y * y < 1;
11     }
12
13     public static void main(String[] args) {
14         Scanner kb = new Scanner(System.in);
15
16         System.out.print("Number of darts: ");
17         int n = kb.nextInt();
18
19         int count = 0;
20         for (int i = 0; i < n; i++) {
21             double x = dartToss();
22             double y = dartToss();
23             if (inside(x, y))
24                 count++;
25         }
26
27         double PiApprox = 4.0 * count / n;
28         System.out.println("Pi is approximately = " + PiApprox);
29     }
30 }
```

2. A Pyramid is an object that has a rectangular base with dimensions Width and Length and triangular sides that all meet in a single point that is Height distance to the center of the base. Please see the illustration below.



Write a class called Pyramid that stores the length, width, and height of the pyramid. The class is to have two constructors, a default constructor that sets the length, width, and height all to 1 and another constructor that brings in the length, width, and height as parameters. Also in the class are the following methods, for the formulas presented we use L, W, and H for the values of the length, width, and height respectively.

(a) Create a method called `volume` that calculates and returns the volume of the pyramid. Recall that this is, $\frac{1}{3}$ the area of the base times the height.

(b) Create a method called `baseArea` that calculates and returns the area of the base.

(c) Create a method called `basePerimeter` that calculates and returns the perimeter of just the base.

(d) Create a method called `Perimeter` that calculates and returns the perimeter of the pyramid. This is the sum of the 8 line lengths on the edges of the object, 4 from the base and the other 4 from the triangles. Each of the four triangle sides have length

$$\sqrt{H^2 + \left(\frac{L}{2}\right)^2 + \left(\frac{W}{2}\right)^2}$$

(e) Create a method called `surfaceArea` that calculates and returns the surface area of the pyramid. This is the sum of the area of the base and the the areas of the 4 triangles. Two of those triangles have area

$$\frac{W}{2}\sqrt{H^2 + \left(\frac{L}{2}\right)^2}$$

and the other two triangles have area

$$\frac{L}{2}\sqrt{H^2 + \left(\frac{W}{2}\right)^2}$$

(f) Create a method called `isSquareBase` that returns true if the base is a square and false if it is not.

Once the Pyramid class is constructed, create a main that declares two pyramids `pyr1` and `pyr2`. `pyr1` is to be a default pyramid and `pyr2` is to have length 2, width 4.5 and height 3.7. Have the main then call the method `PrintInfo` that takes in a Pyramid as its only parameter and print to the screen the volume, base area, base perimeter, perimeter, and surface area of the pyramid. Do two calls, one for each of the two pyramids.

Solution:

```java
public class Pyramid {
    private double length;
    private double width;
    private double height;

    public Pyramid() {
        length = 1;
        height = 1;
        width = 1;
    }

    public Pyramid(double l, double w, double h) {
        length = l;
        height = h;
        width = w;
    }

    public double volume() {
        return 1.0 / 3.0 * height * width * length;
    }

    public double baseArea() {
        return width * length;
    }

    public double basePerimeter() {
        return 2 * width + 2 * length;
    }

    public double Perimeter() {
        return 2 * width + 2 * length
                + 4 * Math.sqrt(height * height + (length / 2) * (length / 2)
                        + (width / 2) * (width / 2));
    }

    public double surfaceArea() {
        return width * length
                + width * Math.sqrt(height * height + (length / 2) * (length / 2))
                + length * Math.sqrt(height * height + (width / 2) * (width / 2));
    }

    public boolean isSquareBase() {
        return width == length;
    }
}
```

```java
public class Exam2Prog2 {

    public static void PrintInfo(Pyramid p) {
        System.out.println(p.volume());
        System.out.println(p.baseArea());
        System.out.println(p.basePerimeter());
        System.out.println(p.Perimeter());
        System.out.println(p.surfaceArea());
    }

    public static void main(String[] args) {
        Pyramid pyr1 = new Pyramid();
        Pyramid pyr2 = new Pyramid(2, 4.5, 3.7);
        PrintInfo(pyr1);
        PrintInfo(pyr2);
    }
}
```