1 Short Answer (10 Points Each)

1. What is the difference between a compiler and an interpreter? Also, discuss Java's method. In addition, explain why this makes Java a"platform-independent language."

   Solution:

   A compiler will take a program written in a high-level language, translate it into machine language and then save the machine language program to a file that can be run on the computer. An interpreter does essentially the same thing except that it translates the high-level language to machine language one command at a time and does not save the machine language program to a file. Java uses a combination of the two. There is a compile stage that translates the Java code into byte-code that the interpreter (known as the JVM or Java Virtual Machine) runs.

   Java is compiled into byte-code, this byte code is then interpreted by the Java Virtual Machine (JVM). There is a JVM built for every common operating system, so Java byte-code can be run on any operating system.

2. Answer the following questions about numeric data types in Java.

   (a) What happens when you overload an int?
   Solution: The value cycles around to the minimum value of an int.

   (b) What happens when you overload a double?
   Solution: The value turns into Infinity.

   (c) What happens when you underload an int?
   Solution: The value cycles around to the maximum value of an int.

   (d) What happens when you underload a double?
   Solution: The value turns into 0.

   (e) What happens when you input an integer when the Scanner is doing a nextDouble?
   Solution: The integer gets cast to a double.

3. Write a block of code that will take integers from the keyboard until a negative or 0 is input. Once a negative or 0 has been input the block should print out the sum of the positive number inputs.

   Solution:

```
Scanner kb = new Scanner(System.in);
int num = 0;
int sum = 0;
do {
    sum += num;
    num = kb.nextInt();
} while (num > 0);
System.out.println(sum);
```

4. Write declarations for the following,

   (a) A one-dimensional array of 1,234 doubles.
   Solution:

```
double [] A = new double[1234];
```

   (b) A two-dimensional array of integers that has 76 rows and 52 columns.
   Solution:

```
int [][] A = new int[76][52];
```

   (c) An object named `thing1` with type `Thing` that has one constructor that takes no parameters.
   Solution:

```
Thing thing1 = new Thing();
```

5. Write a method called `fib` that takes in as its only parameter a single integer n and prints out the first n terms of the Fibonacci sequence. The Fibonacci sequence starts out with two 1's and every entry after that is the sum of the last two entries. For example, if the user inputs a 10 the output would be, `1 1 2 3 5 8 13 21 34 55 89 144`.

   Solution:

```
1  public static void fib(int n) {
2      int last = 1;
3      int last2 = 1;
4      System.out.print(last2 + " " + last + " ");
5      for (int i = 0; i < n; i++) {
6          int temp = last;
7          last = last + last2;
8          last2 = temp;
9          System.out.print(last + " ");
10     }
11 }
```

6. Write a method that will simulate the rolling of two die and counts the number of rolls it takes to get a total roll of 7, five times in a row. The method should not take in any parameters and should return the number of rolls needed.

   Solution:

```
1  public static int rolls() {
2      int rolls = 0;
3      int count = 0;
4      while (count < 5) {
5          int die1 = (int) (Math.random() * 6) + 1;
6          int die2 = (int) (Math.random() * 6) + 1;
7
8          if (die1 + die2 == 7)
9              count++;
10         else
11             count = 0;
12
13         rolls++;
14     }
15     return rolls;
16 }
```

7. Write a method that will take in a two-dimensional array of doubles as its only parameter and return a one-dimensional array of doubles that holds the column averages of the input array. An example of an input array and its column averages array are below.

```
-2.00   1.00  -9.00
 2.00  -8.00  -4.00
-1.00   1.00  -3.00
 5.00  -1.00   5.00
-7.00   6.00   0.00

-0.60  -0.20  -2.20
```

   Solution:

```
1  public static double[] ColAvg(double A[][]) {
2      int dim1 = A.length;
3      int dim2 = A[0].length;
4
5      double[] avgs = new double[dim2];
6
7      for (int j = 0; j < dim2; j++) {
8          double sum = 0;
9          for (int i = 0; i < dim1; i++) {
10             sum += A[i][j];
11         }
12         avgs[j] = sum / dim1;
13     }
14
15     return avgs;
16 }
```

8. Write a method that will take in a single string as a parameter and return the number of vowels in the string. In this method you should consider only `a`, `e`, `i`, `o`, and `u` as vowels and the count must be case-insensitive. .

Solution:

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

9. Write a segment of code that takes an integer value of `t`, which you are to assume has a positive value before the segment of code, and then produces the following image.

Program Run, when t is 7

```
*
 **
  ***
   ****
    *****
     ******
      *******
     ******
    *****
   ****
  ***
 **
*
```

Same Run, showing spaces

```
␣*
␣␣**
␣␣␣***
␣␣␣␣****
␣␣␣␣␣*****
␣␣␣␣␣␣******
␣␣␣␣␣␣␣*******
␣␣␣␣␣␣␣******
␣␣␣␣␣␣*****
␣␣␣␣␣****
␣␣␣***
␣␣**
␣*
```

In the image the number of spaces before the $\star$ are the same as the number of $\star$'s. These start at one and go up to the value of t, and then back down to 1. In addition, you may only use the following print statements in the code.

```
System.out.print(" ");
System.out.print("*");
System.out.println();
```

Solution:

```
for (int i = 1; i < t; i++) {
    for (int j = 1; j <= i; j++) {
        System.out.print(" ");
    }

    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}

for (int i = t; i >= 0; i--) {
    for (int j = 1; j <= i; j++) {
        System.out.print(" ");
    }

    for (int j = 1; j <= i; j++) {
        System.out.print("*");
    }
    System.out.println();
}
```

10. Write a linear search method for an integer array that takes in an array and target value as parameters and returns the first position of the target in the array. If the target is not in the array then the method should return -1.

    Solution:

```java
public static int LinearSearch(int[] A, int N) {
    for (int index = 0; index < A.length; index++) {
        if (A[index] == N)
            return index;
    }
    return -1;
}
```

11. Write a method that does either the bubble sort, insertion sort or selection sort for an array of integers. You must also state which sort you are writing.

    Solution:

```java
public static void BubbleSort(int A[]) {
    for (int i = A.length - 1; i > 0; i--) {
        for (int j = 0; j < i; j++) {
            if (A[j] > A[j + 1]) {
                int temp = A[j];
                A[j] = A[j + 1];
                A[j + 1] = temp;
            }
        }
    }
}

public static void InsertionSort(int[] A) {
    for (int itemsSorted = 1; itemsSorted < A.length; itemsSorted++) {
        int temp = A[itemsSorted];
        int loc = itemsSorted - 1;
        while (loc >= 0 && A[loc] > temp) {
            A[loc + 1] = A[loc];
            loc = loc - 1;
        }
        A[loc + 1] = temp;
    }
}

public static void SelectionSort(int[] A) {
    for (int lastPlace = A.length - 1; lastPlace > 0; lastPlace--) {
        int maxLoc = 0;
        for (int j = 1; j <= lastPlace; j++)
            if (A[j] > A[maxLoc])
                maxLoc = j;

        int temp = A[maxLoc];
        A[maxLoc] = A[lastPlace];
        A[lastPlace] = temp;
    }
}
```

12. (*2 points*) Never take a Computer Science course with a Mathematician.

2 Program Traces (20 Points Each)

1. Write the output of the program in the Program Output box. The user's input was 7.

```
1  import java.util.Scanner;
2
3  public class Trace1 {
4
5      public static void PrintArray(int A[]) {
6          for (int i = 0; i < A.length; i++) {
7              System.out.printf("%4d", A[i]);
8          }
9          System.out.println();
10     }
11
12     public static void Make(int A[]) {
13         for (int i = 0; i < A.length; i++) {
14             A[i] = (3 * i + 2) % A.length;
15         }
16     }
17
18     public static void Mix(int A[]) {
19         for (int i = 0; i < A.length; i++) {
20             A[i] = A[A[i]];
21         }
22     }
23
24     public static void Mix2(int A[]) {
25         for (int i = 0; i < A.length; i++) {
26             A[i] = A[(i + 1) % A.length];
27         }
28     }
29
30     public static void main(String[] args) {
31         Scanner kb = new Scanner(System.in);
32         System.out.print("n = ");
33         int num = kb.nextInt();
34
35         int[] Arr = new int[num];
36         Make(Arr);
37         PrintArray(Arr);
38         Mix(Arr);
39         PrintArray(Arr);
40         Make(Arr);
41         PrintArray(Arr);
42         Mix2(Arr);
43         PrintArray(Arr);
44         Mix2(Arr);
45         Mix2(Arr);
46         PrintArray(Arr);
47     }
48 }
```

Solution:

```
n = 7
   2   5   1   4   0   3   6
   1   3   3   0   1   0   6
   2   5   1   4   0   3   6
   5   1   4   0   3   6   5
   4   0   3   6   5   1   4
```

2. For the given input, write the output of the program in the Program Output box. Make sure that you represent spaces by our special symbol ␣.

```
1  public class Thing {
2
3      private int a;
4      private int b;
5      private String str;
6
7      public Thing() {
8          a = 3;
9          b = 7;
10         str = "This is the final exam.";
11     }
12
13     public Thing(int x, int y, String z) {
14         a = y;
15         b = x;
16         str = z;
17     }
18
19     public void set(boolean t, int val) {
20         b = val;
21         if (t)
22             a = val;
23     }
24
25     public void trimThing() {
26         clamp();
27         str = str.substring(a, b);
28     }
29
30     public void slimThing() {
31         clamp();
32         str = str.substring(a);
33     }
34
35     public void change() {
36         int temp = a;
37         a = b;
38         b = temp;
39     }
40
41     public void clamp() {
42         if (a > b)
43             change();
44
45         if (b > str.length())
46             b = str.length();
47         if (a < 0)
48             a = 0;
49     }
50
51     public void nifty() {
52         if (a % 2 == 0)
53             b = b / 2;
54         else
55             b = 3 * b + 1;
56     }
57
58     public void realnifty() {
59         if (b % 2 == 0)
60             b = b / 2;
61         else
62             b = 3 * b + 1;
63     }
64
65     public void nifty(int i) {
66         a = b * i - 4;
67         b = a++;
68     }
69
70     public int almostnifty() {
71         clamp();
72         return b - a;
73     }
74
75     public String toString() {
76         return a + "/" + b + "/" + str + "|";
77     }
78 }
```

```
1  public class Trace2 {
2
3      public static void main(String[] args) {
4          Thing thing1 = new Thing();
5          Thing thing2 = new Thing(5, 23,
6                  "This is a piece of cake.");
7          System.out.println(thing1.toString());
8          System.out.println(thing2.toString());
9          thing1.trimThing();
10         System.out.println(thing1.toString());
11         thing2.nifty();
12         System.out.println(thing2.toString());
13         thing2.realnifty();
14         System.out.println(thing2.toString());
15         thing2.trimThing();
16         System.out.println(thing2.toString());
17         System.out.println(thing1.almostnifty());
18         System.out.println(thing2.almostnifty());
19         thing2.nifty(thing1.almostnifty());
20         System.out.println(thing2.toString());
21         thing2.slimThing();
22         System.out.println(thing2.toString());
23     }
24 }
```

Solution:

```
3/7/This is the final exam.|
23/5/This is a piece of cake.|
3/7/s is|
23/16/This is a piece of cake.|
23/8/This is a piece of cake.|
8/23/a piece of cake|
1
7
12/11/a piece of cake|
11/12/cake|
```

3 Coding (20 Points Each)

1. Below is the start of a program that creates a random one-dimensional array of a size that is input by the user and calculates some statistics on the numbers in the array. The main program is below with a sample run.

```
1  import java.util.Scanner;
2
3  public class Prog1 {
4
5  <<<  Your Methods  >>>
6
7          public static void main(String[] args) {
8          Scanner kb = new Scanner(System.in);
9          System.out.print("n = ");
10         int n = kb.nextInt();
11
12         double[] A = new double[n];
13         populate(A);
14
15         System.out.println("Statistics");
16         System.out.println("Sum = " + Sum(A));
17         System.out.println("Product = " + Product(A));
18         System.out.println("Mean = " + Mean(A));
19         System.out.println("Variance = " + Variance(A));
20         System.out.println("Standard Deviation = " + StandardDeviation(A));
21         System.out.println("Min = " + Min(A));
22         System.out.println("Max = " + Max(A));
23         System.out.println("Number of Positive Values = " + Pos(A));
24         System.out.println("Number of Negative Values = " + Neg(A));
25     }
26 }
```

Program Run:

```
n = 100
Statistics
Sum = -87.71991515761849
Product = 4.85599966888048E91
Mean = -0.8771991515761849
Variance = 186.06052059589248
Standard Deviation = 13.640400309224523
Min = -24.42303463490748
Max = 24.644272787260306
Number of Positive Values = 48
Number of Negative Values = 52
```

The functions/methods you are to create and what they do are described below.

(a) `populate` — This function populates the array with random decimal numbers between -25 and 25.

(b) `Sum` — This function calculates and returns the sum of the array entries. If the size of the array is less than one return 0.

(c) `Product` — This function calculates and returns the product of the array entries. If the size of the array is less than one return 0.

(d) `Mean` — This function calculates and returns the average of the array entries. If the size of the array is less than one return 0. Mean is another word for average.

(e) `Variance` — This function calculates and returns the variance of the array entries. The variance of a list of numbers is defined to be

$$\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}$$

where x_i represents the i^{th} entry in the array and μ is the average (or mean) of the array entries. The sum is taken over all array entries and n is the size of the array. If the size of the array is less than two the function should return 0.

(f) `StandardDeviation` — This function calculates and returns the standard deviation of the array entries. The standard deviation of a list of numbers is defined to be the square root of the variance. As with the variance, if the size of the array is less than two return 0.

(g) `Max` — This function returns the maximum value on the array.

(h) `Min` — This function returns the minimum value on the array.

(i) `Pos` — This function returns the number of array entries that are strictly greater than 0.

(j) `Neg` — This function returns the number of array entries that are strictly less than 0.

Solution:

```
1  import java.util.Scanner;
2
3  public class Prog1 {
4
5      public static void populate(double[] A) {
6          for (int i = 0; i < A.length; i++)
7              A[i] = Math.random() * 50 - 25;
8      }
9
10     public static double Product(double[] A) {
11         if (A.length == 0)
12             return 0;
13
14         double prod = 1;
15         for (int i = 0; i < A.length; i++)
16             prod *= A[i];
17         return prod;
18     }
19
20     public static double Sum(double[] A) {
21         double sum = 0;
22         for (int i = 0; i < A.length; i++)
23             sum += A[i];
24         return sum;
25     }
26
27     public static double Mean(double[] A) {
28         if (A.length == 0)
29             return 0;
30         else
31             return Sum(A) / A.length;
32     }
33
34     public static double Variance(double[] A) {
35         if (A.length <= 1)
36             return 0;
37
38         double m = Mean(A);
39         double sum = 0;
40         for (int i = 0; i < A.length; i++)
41             sum += (A[i] - m) * (A[i] - m);
42
43         return sum / (A.length - 1);
44     }
45
46     public static double StandardDeviation(double[]
           A) {
47         return Math.sqrt(Variance(A));
48     }
49
50     public static double Min(double[] A) {
51         if (A.length == 0)
52             return 0;
53
54         double min = A[0];
55         for (int i = 0; i < A.length; i++)
56             if (min > A[i])
57                 min = A[i];
58
59         return min;
60     }
61
62     public static double Max(double[] A) {
63         if (A.length == 0)
64             return 0;
65
66         double max = A[0];
67         for (int i = 0; i < A.length; i++)
68             if (max < A[i])
69                 max = A[i];
70
71         return max;
72     }
73
74     public static int Pos(double[] A) {
75         int count = 0;
76
77         for (int i = 0; i < A.length; i++)
78             if (A[i] > 0)
79                 count++;
80
81         return count;
82     }
83
84     public static int Neg(double[] A) {
85         int count = 0;
86
87         for (int i = 0; i < A.length; i++)
88             if (A[i] < 0)
89                 count++;
90
91         return count;
92     }
93
94     public static void main(String[] args) {
95         Scanner kb = new Scanner(System.in);
96         System.out.print("n = ");
97         int n = kb.nextInt();
98
99         double[] A = new double[n];
100        populate(A);
101
102        System.out.println("Statistics");
103        System.out.println("Sum = " + Sum(A));
104        System.out.println("Product = " + Product(A
               ));
105        System.out.println("Mean = " + Mean(A));
106        System.out.println("Variance = " + Variance
               (A));
107        System.out.println("Standard Deviation = "
               + StandardDeviation(A));
108        System.out.println("Min = " + Min(A));
109        System.out.println("Max = " + Max(A));
110        System.out.println("Number of Positive
               Values = " + Pos(A));
111        System.out.println("Number of Negative
               Values = " + Neg(A));
112    }
113 }
```

2. First create a new class called Point. Point is to store two private data members x and y, both doubles. It is to have two constructors, one default that sets both x and y to 0, and one that takes the values of x and y in as parameters. There are to be four more methods, their names and function is listed below.

   (a) `distance` — This method takes in a single Point as a parameter and returns the distance between the parameter point and the calling point. Recall that the distance between two points (x_1, y_1) and (x_2, y_2) in two dimensions is

   $$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

   (b) `toString` — This function returns a string representation of the calling point in order paired form. For example, if the point is $(-1, 5)$ then the string would look like `(-1, 5)`.

   (c) `setRandom` — This function takes in two parameters, a minimum value and a maximum value, both doubles, and sets both x and y to random decimal numbers between the minimum and maximum values.

   (d) `clone` — This function returns a point object that has the same values as the calling point.

   Now create a main that will take the number of points to use from the user. For each point the program should create a point with random x and y values between -1 and 1. It should count the number of points that are in the unit circle. If c is the number of points that are in the circle and n is the number of points total, the program should output $4 \cdot \frac{c}{n}$, which is an approximation to π. In addition, the program should output the point that was the furtherest from the origin. You may only use the above methods in the Point class, you may not add any other methods. A program run is below.

   Program Run:

```
n = 1000000
Pi Approx = 3.139436
Max. Point = (-0.9992556896098608, -0.9998335631227706)
```

   Solution:

```java
public class Point {

    private double x;
    private double y;

    public Point() {
        x = 0;
        y = 0;
    }

    public Point(double X, double Y) {
        x = X;
        y = Y;
    }

    public double distance(Point p) {
        return Math.sqrt((x - p.x) * (x - p.x) + (y
              - p.y) * (y - p.y));
    }

    public String toString() {
        return "(" + x + ", " + y + ")";
    }

    public void setRandom(double min, double max) {
        x = Math.random() * (max - min) + min;
        y = Math.random() * (max - min) + min;
    }

    public Point clone() {
        Point p = new Point(x, y);
        return p;
    }
}
```

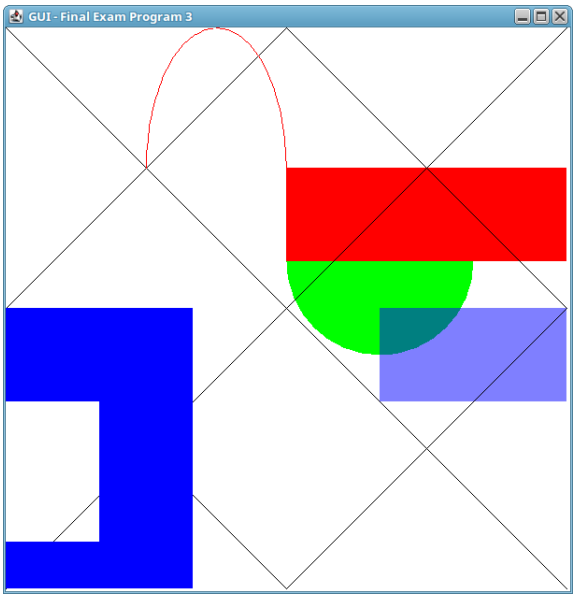
```java
import java.util.Scanner;

public class Prog2 {

    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        System.out.print("n = ");
        int n = kb.nextInt();

        Point o = new Point();
        Point p = new Point();
        Point maxpt = new Point();
        double max = 0;
        int count = 0;
        for (int i = 0; i < n; i++) {
            p.setRandom(-1, 1);
            if (p.distance(o) < 1)
                count++;
            if (p.distance(o) > max){
                max = p.distance(o);
                maxpt = p.clone();
            }
        }
        System.out.println("Pi Approx = " + 1.0 *
             count / n * 4.0);
        System.out.println("Max. Point = " + maxpt.
             toString());
    }
}
```

3. Write the `paint` method for the graphics panel that produced this image. The graphics panel is 600×600 pixels in size. Note that all of the straight black lines are drawn together in one block of code. All other objects are drawn either before the black lines or after the black lines.



Solution:

```
1  import java.awt.*;
2  import javax.swing.*;
3
4  public class GraphicsJPanel extends JPanel {
5
6      public GraphicsJPanel() {
7          setBackground(Color.white);
8      }
9
10     public void paint(Graphics g) {
11         super.paint(g);
12
13         g.setColor(Color.green);
14         g.fillOval(300, 150, 200, 200);
15
16         g.setColor(Color.red);
17         g.fillRect(300, 150, 300, 100);
18
19         g.setColor(new Color(0, 0, 1.0f, 0.5f));
20         g.fillRect(400, 300, 200, 100);
21
22         g.setColor(Color.black);
23         g.drawLine(0, 300, 300, 0);
24         g.drawLine(0, 600, 600, 0);
25         g.drawLine(300, 600, 600, 300);
26         g.drawLine(0, 0, 600, 600);
27         g.drawLine(300, 0, 600, 300);
28         g.drawLine(0, 300, 300, 600);
29
30         g.setColor(Color.red);
31         g.drawArc(150, 0, 150, 300, 0, 180);
32
33         g.setColor(Color.blue);
34         g.fillRect(0, 300, 100, 100);
35
36         g.setColor(Color.blue);
37         g.fillRect(100, 300, 100, 300);
38
39         g.setColor(Color.blue);
40         g.fillRect(0, 550, 100, 50);
41     }
42 }
```