1 Short Answer (5 Points Each)

1. What is the difference between a compiler and an interpreter? Also, discuss Java's method.

   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.

2. Java is a "platform-independent language." What is a *platform*, what does *platform-independent* mean, and how does Java attain its platform independence?

   Solution: A platform is an operating system, so platform-independent means that the same program can be run on any operating system. 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.

3. What are the three types of programming errors? Briefly describe each of them.

   Solution:

   Syntax Error: An error in the program code due to misuse of the programming language.

   Run-time Error: An error that occurs during a run of the program which usually causes the program to terminate prematurely.

   Logic Error: This error occurs when the program is syntactically correct and there are no runtime errors but the program does not do what it was intended to do.

4. What are reserved words? Give four examples of Java reserved words.

   Solution: A reserved word is a word that is used for a particular use in the programming language and cannot be redefined. Hence the programmer cannot use a reserved word as a variable name. There are many reserved words in Java, some we have seen thus far are public, void, class, if, else, while, int, double, long, float, new, import, and static.

5. 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 into position 10 of an integer array of size 10?
   Solution: You get an out of bounds exception.

2 Short Coding Answer (10 Points Each)

1. Write a segment of code that will simulate the rolling of two die and counts the number of rolls it takes to get snake-eyes 5 times in a row. Snake-eyes is when both die have a 1 on the upper face.

   Solution:

```
int rolls = 0;
int snakeEyesCount = 0;
while (snakeEyesCount < 5){
    int die1 = (int)(Math.random()*6) + 1;
    int die2 = (int)(Math.random()*6) + 1;

    if (die1 + die2 == 2)
        snakeEyesCount++;
    else
        snakeEyesCount = 0;

    rolls++;
}

System.out.println(rolls);
```

2. Write a function called `Factorial` that takes a long as a parameter and returns the factorial of that long, the return type is also a long. Recall that the definition of a factorial is $n! = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1$ and we define $0! = 1$.

   Solution:

```
public static long Factorial(long n) {
    long result = 1;
    for (long i = 1; i <= n; i++)
        result *= i;

    return result;
}
```

3. 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();
}
```

4. 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:

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

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

   Solution:

```
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;
    }
}
```

6. Write a segment of code that will add up all of the entries on the diagonal of a two-dimensional integer array, `A`. The diagonal of an array are the entries in the $(0,0)$, $(1,1)$, $(2,2)$, $\ldots$, (n,n) positions, where the (n,n) position is the largest possible position that is still in the array. So for a 4×9 array $n = 4$ and for a 5×3 array $n = 3$. You do not know the size of the array before this block of code is executed.

   Solution:

```
int n = A.length;
if (A[0].length < n)
    n = A[0].length;

int sum = 0;
for (int i = 0; i < n; i++)
    sum += A[i][i];
```

7. Write a segment of code that will declare an ArrayList of integers, populate the list with the numbers from 1 to 1000000, then continually remove the last entry until only 10 entries remain in the list, then finally print out the list.

   Solution:

```
ArrayList<Integer> A = new ArrayList<Integer>();
for (int i = 1; i < 1000000; i++)
    A.add(i);

while (A.size() > 10)
    A.remove(A.size() - 1);

System.out.println(A);
```

8. Write a segment of code that will take an integer array `A` and sum up all of the even indexed entries and subtract off all of the odd indexed entries.

   Solution:

```
int val = 0;
for (int i = 0; i < A.length; i++)
    if (i % 2 == 0)
        val += A[i];
    else
        val -= A[i];
```

3 Program Traces (20 Points Each)

1. For each of the three runs, give the program output.

```
1  import java.util.Scanner;
2
3  public class FinalTrace1 {
4
5      public static void main(String[] args) {
6          Scanner keyboard = new Scanner(System.in);
7          System.out.print("n = ");
8          int n = keyboard.nextInt();
9          System.out.print("m = ");
10         int m = keyboard.nextInt();
11
12         int j = 1;
13         int i = 10;
14
15         do {
16             System.out.println("n is " + n);
17             if (n % 3 == 0) {
18                 j++;
19             } else if (n % 3 == 1) {
20                 i = i - 3 * j;
21             } else {
22                 i++;
23                 j++;
24             }
25             n++;
26             m -= 2;
27         } while (n < m);
28
29         System.out.println("i is " + i);
30         System.out.println("j is " + j);
31     }
32 }
```

Run #1 Solution

```
n = 3
m = 16
n is 3
n is 4
n is 5
n is 6
n is 7
i is -7
j is 4
```

Run #2 Solution

```
n = 6
m = 6
n is 6
i is 10
j is 2
```

Run #3 Solution

```
n = 5
m = 21
n is 5
n is 6
n is 7
n is 8
n is 9
n is 10
i is -12
j is 5
```

2. Give the program output.

```
1  import java.util.Scanner;
2
3  public class FinalTrace2 {
4
5      public static int doSomething(int a) {
6          return (int) Math.pow(a, 2);
7      }
8
9      public static int doSomething(int a, int b) {
10         return a / b;
11     }
12
13     public static String doSomething(int a, int b,
           String c) {
14         if (a > b) {
15             int t = a;
16             a = b;
17             b = t;
18         }
19         return c.substring(a, b);
20     }
21
22     public static void main(String[] args) {
23         Scanner keyboard = new Scanner(System.in);
24         System.out.print("s = ");
25         String s = keyboard.nextLine();
26         System.out.print("n = ");
27         int n = keyboard.nextInt();
28         System.out.print("m = ");
29         int m = keyboard.nextInt();
30         System.out.println("-----");
31
32         System.out.println(s.length());
33         System.out.println(doSomething(n, m));
34         System.out.println(doSomething(n, m, s));
35
36         System.out.println("-----");
37
38         int t = doSomething(n) + s.length() / 3;
39         int r = m / 5;
40         System.out.println(t);
41         System.out.println(r);
42         System.out.println(doSomething(t, r, s));
43
44         System.out.println("-----");
45
46         System.out.println(doSomething(n));
47         System.out.println(doSomething(doSomething(
               n)));
48         System.out.println(doSomething(doSomething(
               doSomething(n)), m));
49
50         System.out.println("-----");
51
52         System.out.println(doSomething(m, n));
53         System.out.println(2 * (doSomething(n) - 1)
               );
54         System.out.println(doSomething(doSomething(
               m, n), 2 * (doSomething(n) - 1), s));
55     }
56 }
```

Program Run

```
s = This is a test of the program.
n = 3
m = 11
-----
30
0
s is a t
-----
19
2
is is a test of t
-----
9
81
7
-----
3
16
s is a test o
```

3. For each of the three runs, give the program output.

```
1  import java.util.ArrayList;
2  import java.util.Scanner;
3
4  public class FinalTrace3 {
5
6      public static void print(int M[]) {
7          for (int i = 0; i < M.length; i++)
8              System.out.printf("%3d", M[i]);
9          System.out.println();
10     }
11
12     public static void move(int M[]) {
13         for (int i = 0; i < M.length; i++)
14             M[i] = M[2 * i % M.length];
15     }
16
17     public static void main(String[] args) {
18         Scanner keyboard = new Scanner(System.in);
19         System.out.print("n = ");
20         int n = keyboard.nextInt();
21
22         if (n < 0)
23             n = -n;
24
25         if (n == 0)
26             n = 10;
27
28         int[] A = new int[n];
29         int[] B = new int[n / 3];
30
31         for (int i = 0; i < n; i++) {
32             A[i] = i + 1;
33         }
34
35         print(A);
36         print(B);
37         System.out.println("-----");
38
39         for (int i = 0; i < B.length; i++) {
40             B[i] = A[2 * i % A.length];
41         }
42
43         print(A);
44         print(B);
45         System.out.println("-----");
46
47         move(A);
48         print(A);
49         System.out.println("-----");
50
51         A = B;
52         B[0] = 25;
53         A[A.length - 1] = -7;
54
55         print(A);
56         print(B);
57     }
58 }
```

Run #1 Solution

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

Run #2 Solution

```
n = 10
  1  2  3  4  5  6  7  8  9 10
  0  0  0
-----
  1  2  3  4  5  6  7  8  9 10
  1  3  5
-----
  1  3  5  7  9  1  5  9  5  5
-----
 25  3 -7
 25  3 -7
```

Run #3 Solution

```
n = 12
 1 2 3 4 5 6 7 8 9 10 11 12
 0 0 0 0
-----
 1 2 3 4 5 6 7 8 9 10 11 12
 1 3 5 7
-----
 1 3 5 7 9 11 1 5 9 1 9 9
-----
 25  3  5 -7
 25  3  5 -7
```

4 Coding (20 Points Each)

1. Write a program that will take integer input from the user until the user types in a negative number. You may assume that the user does not type in a number larger than 1,000,000,000. For each input the program is to print out if the number is even or odd. Once the user types in a negative number the program is to display the minimum input and the maximum input. You may not use an array or ArrayList for this program. A sample run is on the right.

Solution:

```
1  import java.util.Scanner;
2
3  public class FinalCode001 {
4
5      public static void main(String[] args) {
6          Scanner keyboard = new Scanner(System.in);
7          int val = 0;
8          int min = 1000000000;
9          int max = -1;
10
11         while (val > -1) {
12             System.out.print("Input: ");
13             val = keyboard.nextInt();
14             if (val % 2 == 0)
15                 System.out.println("Input was even.
                       ");
16             else
17                 System.out.println("Input was odd."
                       );
18
19             if (val > -1) {
20                 if (val > max)
21                     max = val;
22
23                 if (val < min)
24                     min = val;
25             }
26         }
27
28         System.out.println();
29         System.out.println("Minimum = " + min);
30         System.out.println("Maximum = " + max);
31     }
32 }
```

Sample Run

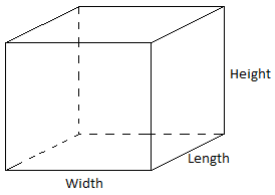
```
Input: 5
Input was odd.
Input: 24
Input was even.
Input: 9
Input was odd.
Input: 2
Input was even.
Input: 7
Input was odd.
Input: 15
Input was odd.
Input: -6
Input was even.

Minimum = 2
Maximum = 24
```

2. Create a `Box` class that stores three double values for the length, width, and height of a box. The class is to have a single constructor that loads in the length, width, and height into the object. The class is to have three more methods,

   - `volume` — This returns the volume of the box.
   - `surfaceArea` — This returns the surface area of the box. Which is the sum of the areas of the 6 faces.
   - `edgeLength` — This returns the edge length of the box. Which is the sum of the lengths of the 12 edges of the box.

   Then write a main program that creates a box with length 5, width 3 and height 7, then prints out the volume, surface area, and edge length.



Solution:

```java
public class Box {
    private double l;
    private double w;
    private double h;

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

    public double volume() {
        return l * w * h;
    }

    public double surfaceArea() {
        return 2 * l * w + 2 * l * h + 2 * w * h;
    }

    public double edgeLength() {
        return 4 * (l + w + h);
    }
}
```

```java
public class FinalCode002 {
    public static void main(String[] args) {
        Box b = new Box(5, 3, 7);
        System.out.println("Volume = " + b.volume());
        System.out.println("Surface Area = " + b.surfaceArea());
        System.out.println("Edge Length = " + b.edgeLength());
    }
}
```

3. The main program and output from the program are below. Write the five methods for the program.

   - `print` — The print method is to take in as a parameter only a single two-dimensional array of integers, and print the contents of the array in columns that line up and use 6 spaces for each entry.
   - `populate` — The populate method is to take in as a parameter only a single two-dimensional array of integers, and fill the array with random integers between -50 and 50.
   - `Add` — The Add method is to take as parameters two, two-dimensional arrays and return a two-dimensional array that is the same size as the two input arrays and is the sum of the two arrays. The sum of two arrays is the array where each corresponding cell is added. So the $(0,0)$ positions of the two are added, the $(0,1)$ positions of the two are added, the $(1,1)$ positions of the two are added, and so on. If the sizes of the two input arrays are not the same then they cannot be added and in this case the method should return `null`.
   - `Subtract` — The Subtract method is to take as parameters two, two-dimensional arrays and return a two-dimensional array that is the same size as the two input arrays and is the difference of the two arrays. The difference of two arrays is the array where each corresponding cell is subtracted. So the $(0,0)$ positions of the two are subtracted, the $(0,1)$ positions of the two are subtracted, the $(1,1)$ positions of the two are subtracted, and so on. If the sizes of the two input arrays are not the same then they cannot be subtracted and in this case the method should return `null`.
   - `transpose` — The transpose method is to take in as a parameter only a single two-dimensional array of integers, and return another two-dimensional array that transposes the array. The transpose of an array is where each row becomes a column and each column a row. That is, the first row of the original array is the first column of the transpose, and so on.

   The populate method is the only method that is allowed to alter the input array.

```
1  public static void main(String[] args) {
2      Scanner keyboard = new Scanner(System.in);
3      System.out.print("Rows: ");
4      int rows = keyboard.nextInt();
5      System.out.print("Columns: ");
6      int cols = keyboard.nextInt();
7
8      int[][] A = new int[rows][cols];
9      int[][] B = new int[rows][cols];
10     populate(A);
11     populate(B);
12
13     print(A);
14     System.out.println();
15     print(B);
16
17     int[][] sum = Add(A, B);
18     if (sum != null) {
19         System.out.println();
20         print(sum);
21     }
22
23     int[][] difference = Subtract(A, B);
24     if (difference != null) {
25         System.out.println();
26         print(difference);
27     }
28
29     System.out.println();
30     print(transpose(A));
31 }
```

Sample Run

```
Rows: 3
Columns: 4
   -29    11    48    44
   -35    15    20   -37
   -15     6    18    43

    11     1    45    39
    12    32    50   -29
   -24   -41   -38    17

   -18    12    93    83
   -23    47    70   -66
   -39   -35   -20    60

   -40    10     3     5
   -47   -17   -30    -8
     9    47    56    26

   -29   -35   -15
    11    15     6
    48    20    18
    44   -37    43
```

Solution:

```
import java.util.Scanner;

public class FinalCode003 {

    public static void print(int[][] A) {
        for (int i = 0; i < A.length; i++) {
            for (int j = 0; j < A[0].length; j++)
                System.out.printf("%6d", A[i][j]);
            System.out.println();
        }
    }

    public static void populate(int[][] A) {
        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                A[i][j] = (int) (Math.random() *
                     101 - 50);
    }

    public static int[][] Add(int[][] A, int[][] B
         ) {
        if (A.length != B.length)
            return null;

        if (A[0].length != B[0].length)
            return null;

        int[][] Sum = new int[A.length][A[0].
             length];

        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                Sum[i][j] += A[i][j] + B[i][j];

        return Sum;
    }

    public static int[][] Subtract(int[][] A, int
         [][] B) {
        if (A.length != B.length)
            return null;

        if (A[0].length != B[0].length)
            return null;

        int[][] Dif = new int[A.length][A[0].
             length];

        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                Dif[i][j] += A[i][j] - B[i][j];

        return Dif;
    }

    public static int[][] transpose(int[][] A) {
        int[][] retarr = new int[A[0].length][A.
             length];

        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                retarr[j][i] = A[i][j];

        return retarr;
    }

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Rows: ");
        int rows = keyboard.nextInt();
        System.out.print("Columns: ");
        int cols = keyboard.nextInt();

        int[][] A = new int[rows][cols];
        int[][] B = new int[rows][cols];
        populate(A);
        populate(B);

        print(A);
        System.out.println();
        print(B);

        int[][] sum = Add(A, B);
        if (sum != null) {
            System.out.println();
            print(sum);
        }

        int[][] difference = Subtract(A, B);
        if (difference != null) {
            System.out.println();
            print(difference);
        }

        System.out.println();
        print(transpose(A));
    }
}
```