Exam #3 — Solutions

1 Short Answer/Method Construction (10 Points Each)

1. Write a method that will bring in a one-dimensional array of integers as a parameter and shift all the values to the right one cell and the last cell value should cycle back to the first. For example, if the first line is the original array then the second line is what it is holding after doing the shift.

3	17	20	5	13	18	16	1	4	20	16	5	4	3	10

10	3	17	20	5	13	18	16	1	4	20	16	5	4	3

Solution:

```java
public static void shift(int [] A) {
    int temp = A[A.length-1];
    for (int i = A.length - 1; i > 0; i--)
        A[i] = A[i-1];

    A[0] = temp;
}
```

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 7. You may not assume the user always types in a valid integer, this method should accept any type of input without crashing the program.

```
Input a number: 23
Input a number: 54
Input a number: dirt
Input a number: 3.54
Input a number: 7
```

Solution:

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

    return returnValue;
}
```

3. For the following one-dimensional array, show the final array state after each pass of the three sorting algorithms. That is, after each iteration of the outside loop for each method.

Solution:

Bubble Sort:

Original Array:

9	21	17	15	3	2	10

```
Pass 1:   9 17 15  3  2 10 21
Pass 2:   9 15  3  2 10 17 21
Pass 3:   9  3  2 10 15 17 21
Pass 4:   3  2  9 10 15 17 21
Pass 5:   2  3  9 10 15 17 21
Pass 6:   2  3  9 10 15 17 21
```

Insertion Sort:

Original Array:

9	21	17	15	3	2	10

```
Pass 1:   9 21 17 15  3  2 10
Pass 2:   9 17 21 15  3  2 10
Pass 3:   9 15 17 21  3  2 10
Pass 4:   3  9 15 17 21  2 10
Pass 5:   2  3  9 15 17 21 10
Pass 6:   2  3  9 10 15 17 21
```

Selection Sort:

Original Array:

9	21	17	15	3	2	10

```
Pass 1:   9 10 17 15  3  2 21
Pass 2:   9 10  2 15  3 17 21
Pass 3:   9 10  2  3 15 17 21
Pass 4:   9  3  2 10 15 17 21
Pass 5:   2  3  9 10 15 17 21
Pass 6:   2  3  9 10 15 17 21
```

4. Write a method that takes in a one-dimensional array of integers as its only parameter and sorts it using either the bubble sort, insertion sort, or selection sort.

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;
    }
}
```

5. Write a method that will take in a two-dimensional array of integers as its only parameter. The method is to find the maximum value for each column and swap the maximum value with the element in the top row of that column. An example is below, the left array is the original and the right array is what it is converted to.

```
  3  3  2 14 16  1  6                  16 18 19 19 20 19 11
  7  9 19 15 20 19 11                   7  9  2 15 16  1  6
  8  1 18 10 12  4  9                   8  1 18 10 12  4  9
 16 18  1 18  7  9  9                   3  3  1 18  7  9  9
  8  9 10 19  8 11 10                   8  9 10 14  8 11 10
```

Solution:

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

6. Write a program that will take in two parameters, the first is a two-dimensional array of integers and the second is a single integer that represents the index of the column of the array that will be returned by the method. For example, if the array that is input is

```
  6 14 12 15 11 10  2
 17 19 13 13 20 19  8
 16  1 14 16 18  6 13
 10  2  5 17  7  7  5
 13  5  1  7  4  4 11
```

and the input column index is 3 the method would return the following one-dimensional array.

```
 15 13 16 17  7
```

If the column index parameter is either negative or too large have the method return `null`.

Solution:

```java
public static int[] ExtractColumn(int A[][], int col) {
    if (col < 0 || col >= A[0].length)
        return null;

    int[] column = new int[A.length];

    for (int i = 0; i < A.length; i++)
        column[i] = A[i][col];

    return column;
}
```

2 Program Traces (10 Points Each)

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

```java
import java.util.Scanner;

public class Exam3_Trace1 {

    public static int[] copyArray(int A[]) {
        int B[] = new int[A.length];

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

        return B;
    }

    public static void printArray(int A[]) {
        for (int i = 0; i < A.length; i++)
            System.out.print(A[i] + " ");

        System.out.println();
    }

    public static void manipulate(int A[], int B[]) {
        for (int i = 0; i < A.length; i++)
            if (i < B.length)
                A[i] = A[A.length - 1 - i] + B[i];
    }

    public static void manipulate2(int A[]) {
        for (int i = 0; i < A.length; i++)
            if (A[i] % 2 == 0)
                A[i] /= 2;
            else
                A[i]++;
    }

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Input Array Size: ");
        int n = keyboard.nextInt();
        System.out.print("Input Start: ");
        int st = keyboard.nextInt();
        System.out.print("Input Stride: ");
        int str = keyboard.nextInt();

        int A[] = new int[n];
        for (int i = 0; i < A.length; i++)
            A[i] = i * str + st;

        int B[] = A;
        int C[] = { 3, 5, -2, 1, 4, -7 };

        printArray(A);
        printArray(B);
        printArray(C);
        System.out.println();
        manipulate(A, C);
        printArray(A);
        printArray(B);
        printArray(C);
        System.out.println();
        B = copyArray(A);
        B[B.length / 2] = C[C.length / 2];
        manipulate2(B);
        printArray(A);
        printArray(B);
        printArray(C);
    }
}
```

```
Input Array Size: 5
Input Start: 2
Input Stride: 3

2 5 8 11 14
2 5 8 11 14
3 5 -2 1 4 -7

17 16 6 17 21
17 16 6 17 21
3 5 -2 1 4 -7

17 16 6 17 21
18 8 2 18 22
3 5 -2 1 4 -7
```

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

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class Exam3_Trace2 {

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

            System.out.println();
        }
    }

    public static void populate(int A[][]) {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Input Vals = ");
        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                A[i][j] = keyboard.nextInt();
    }

    public static int HelpDoSomething(int A[][],
                    int a, int b, int c, int d) {
        return A[a][b] * A[c][d] - A[a][d] * A[c][b];
    }

    public static int DoSomething(int A[][]) {
        if (A.length != 3 || A[0].length != 3)
            return -1;

        return A[0][0] * HelpDoSomething(A, 1, 1, 2, 2)
                - A[1][0] * HelpDoSomething(A, 0, 1, 2, 2)
                + A[2][0] * HelpDoSomething(A, 0, 1, 1, 2);
    }

    public static int DoSomethingElse(int A[][]) {
        int t = 0;
        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                t += A[i][j];

        return t;
    }

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Input n: ");
        int n = keyboard.nextInt();
        int A[][] = new int[n][n];
        populate(A);
        printArray(A, 3);
        int d = DoSomething(A);
        System.out.println(d);

        ArrayList<Integer> B = new ArrayList<Integer>();
        for (int j = 0; j < A[0].length; j++)
            for (int i = 0; i < A.length; i++)
                B.add(A[i][j]);

        System.out.println(B);
        Collections.sort(B, Collections.reverseOrder());
        System.out.println(B);
        System.out.println(B.get(B.size() / 2));
        System.out.println(DoSomethingElse(A));
    }
}
```

```
Input n: 3
Input Vals = 2 5 7 4 3 8 2 1 3

  2  5  7
  4  3  8
  2  1  3
8
[2, 4, 2, 5, 3, 1, 7, 8, 3]
[8, 7, 5, 4, 3, 3, 2, 2, 1]
3
35
```

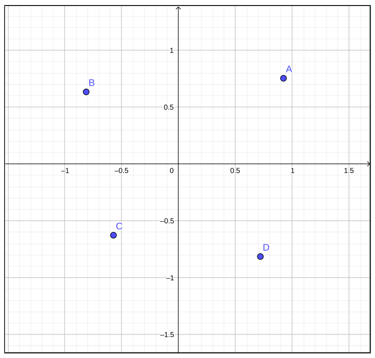
3 Coding (30 Points)

In this exercise you will write a Point class and two separate programs that use the point class.

3.1 The Point Class

The point class is to store two doubles that hold the x and y values of the point (x, y). It is to have two constructors, one default constructor that sets the point to $(0, 0)$, and another that takes in the values of the x and y coordinates as parameters. You do not need to create accessor methods but you need to create four other methods.

- `Length` that returns the length of the line from $(0, 0)$ to the point (x, y). Recall that this calculation is $\sqrt{x^2 + y^2}$.
- `Distance` that brings in a single parameter of type Point and returns the distance between the parameter point and the calling point. Recall that this calculation is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ where the two points are (x_1, y_1) and (x_2, y_2).
- `Quadrant` that returns which quadrant of the plane the point is in. In the picture on the right point A is in quadrant 1, point B is in quadrant 2, point C is in quadrant 3, and point D is in quadrant 4.
- `toString` that returns a string of the point values written like (x, y), for example, `(3.77, -4,92).`



3.2 Program #1

For the first program let the user input a number n. Create a two dimensional array of Points with n rows and 2 columns. Populate the array with random points where the x and y values are between -10 and 10. We will consider each row of this array as a pair of points. Print the array of points out to the screen were each pair is side by side and there is `---` between them. Then print out the quadrant the first point of each pair is in. Have the program call the method `getDistance` on the array of points. The method takes in a two-dimensional array of Points (assumed to be of size $n \times 2$) as its only parameter and returns a one-dimensional array of distances between each pair of points. Have the program print this array to the screen as well. Finally, have the program call the method `maxPos` on the array of distances. This method takes in a one-dimensional array of doubles and returns the position in the array that holds the maximum value. Print this maximum value to the screen. An example run of the program is below.

```
Input n: 5

(4.621087937152064, 9.373610272387523) --- (2.996606919070455, 1.2604253653184152)
(-4.642865964419132, 3.6009712923222903) --- (-6.352128683096274, 6.634165361453121)
(-7.761410618709441, 2.7884670298450214) --- (-0.002813299368952471, 9.8780451356808)
(8.671397530059377, -0.13331859377881017) --- (-1.357844374768515, -2.5110290457164197)
(3.7649972303000325, -7.143568541623184) --- (5.360362052776681, -9.737560916898724)

(4.621087937152064, 9.373610272387523) is in Quadrant 1
(-4.642865964419132, 3.6009712923222903) is in Quadrant 2
(-7.761410618709441, 2.7884670298450214) is in Quadrant 2
(8.671397530059377, -0.13331859377881017) is in Quadrant 4
(3.7649972303000325, -7.143568541623184) is in Quadrant 4

8.274219474633329
3.481644051661517
10.509897719884025
10.307240182454716
3.0453218811455036

Max distance at position 2
```

3.3 Program #2

This is another version of the dartboard Monte-Carlo method to approximate π.

Write a program that will ask the user for the number of darts they would like to throw. Create either an array or an ArrayList (your choice) to hold that many Points (each dart will be represented as a point). Populate the array or ArrayList with random points whose x and y values are between -1 and 1. Have the program call the method `countDarts` on this array or ArrayList of points. This method is to return the number of points that are inside the unit circle. Finally have the program print out the approximation of π from the experiment. Remember that you take the count of the points inside the circle divided by the number thrown times 4.

Solution:

```java
public class Point {
    private double x;
    private double y;

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

    public Point(double a, double b) {
        x = a;
        y = b;
    }

    public double Length() {
        return Math.sqrt(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 int Quadrant() {
        if (x >= 0) {
            if (y >= 0)
                return 1;
            else
                return 4;
        } else {
            if (y > 0)
                return 2;
            else
                return 3;
        }
    }

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

```java
import java.util.Scanner;

public class Exam3Prog_F19 {

    public static double[] getDistances(Point[][] A) {
        double[] D = new double[A.length];
        for (int i = 0; i < A.length; i++)
            D[i] = A[i][0].Distance(A[i][1]);

        return D;
    }

    public static int maxPos(double [] A) {
        int max = 0;
        for (int i = 0; i < A.length; i++)
            if (A[i] > A[max])
                max = i;

        return max;
    }

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);

        System.out.print("Input n: ");
        int n = keyboard.nextInt();
        Point A[][] = new Point[n][2];

        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < 2; j++)
                A[i][j] = new Point(Math.random() * 20 - 10, Math.random() * 20 - 10);

        for (int i = 0; i < A.length; i++)
            System.out.println(A[i][0] + " --- " + A[i][1]);

        System.out.println();

        for (int i = 0; i < A.length; i++)
            System.out.println(A[i][0] + " is in Quadrant " + A[i][0].Quadrant());

        System.out.println();

        double [] D = getDistances(A);

        for (int i = 0; i < D.length; i++)
            System.out.println(D[i]);

        System.out.println();
        int max = maxPos(D);
        System.out.println("Max distance at position " + max);
    }
}
```

```
import java.util.ArrayList;
import java.util.Scanner;

public class Exam3Prog2_F19 {

    public static int countDarts(Point[] A) {
        int count = 0;
        for (int i = 0; i < A.length; i++)
            if (A[i].Length() <= 1)
                count++;

        return count;
    }

    public static int countDarts(ArrayList<Point> A) {
        int count = 0;
        for (int i = 0; i < A.size(); i++)
            if (A.get(i).Length() <= 1)
                count++;

        return count;
    }

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);

        System.out.print("Input n: ");
        int n = keyboard.nextInt();

        // Array version

        Point A[] = new Point[n];
        for (int i = 0; i < A.length; i++)
            A[i] = new Point(Math.random() * 2 - 1, Math.random() * 2 - 1);

        System.out.println("Pi = " + 1.0 * countDarts(A) / n * 4);

        // ArrayList version

        ArrayList<Point> AL = new ArrayList<Point>();

        for (int i = 0; i < n; i++)
            AL.add(new Point(Math.random() * 2 - 1, Math.random() * 2 - 1));

        System.out.println("Pi = " + 1.0 * countDarts(AL) / n * 4);
    }
}
```