Worksheet W1: Arrays and such

Total points: 10

Out: January 21 (Tuesday evening)

Due: January 23 (Thursday end of day [2359 CDT according to D2L])

*No late submissions will be accepted*

## What to submit?

Upload exactly one file to the designated D2L folder. Type the methods on this sheet. I recommend that you use Notepad or some sort of programming editor to type the methods so that Word doesn’t mess up your capitalization. *The solution submitted must be typed.* The worksheet is available online and is open now.

# Exercise 1: Arrays of Primitives

Consider an array of java **int** primitives such as the following:

**int scores[] = {10, 40, -10, 50, 70, 65, 54, 23, 82 };**

Write Java methods to do the following, using the given names. You may assume that the **scores** array in the parameter is not empty.

**public int[] oddAndEven( int[] scores ):**

This method takes the input array **scores** and returns an **int[]** array of size 2 in which the 0th index contains the number of odd numbers in the array, and the 1st index contains the number of even numbers in the array.

**public static int[] oddAndEven( int[] scores ){**

**int even = 0;**

**int odd = 0;**

**int result[] = new int[2];**

**for( int i=0;i<scores.length;i++){**

**if( scores[i] % 2 == 0 ){**

**even++;**

**}**

**else {**

**odd++;**

**}**

**}**

**result[0] = odd;**

**result[1] = even;**

**return result;**

**}**

**public boolean containsX( int[] scores, int x ):**

This method returns true if the scores array contains the value **x**, and returns false if it does not.

*(1 point per method)*

    public static boolean containsX( int[] scores, int x ){

        for (int score: scores){

            if (score == x){

                return true;

            }

        }

        return  false;

    }

# Exercise 2: Arrays of Objects

Consider the **Point** and **Line** classes that are represented by the following UML.





Assume you have an array that stores references to **Line** objects. You may assume that the array is not empty. These objects are not in any kind of sorted order. Write the following Java methods:

**public Line longestLine( Line[] lineArray ):**

Returns the **Line** with maximum length. If there are multiple **Lines** that tie for the longest length, it may return any one of those **Lines**.

**public Line longestLine( Line[] lineArray ){**

**Line longest = new Line(new Point(0.0,0.0), new Point(0.0,0.0));**

**for (Line line: lineArray){**

**if (line.length() > longest.length()){**

**longest = line;**

**}**

**}**

**return longest;**

**}**

**public void deleteOriginLines( Line[] lineArray ):**

Deletes every **Line** that starts or ends at point (0,0). You do not need to fill in the null values. Just null out the appropriate elements of lineArray.

*(2 points per method)*

        public void deleteOriginLines( Line[] lineArray ){

            start = new Point(0.0,0.0);

            end = new Point(0.0,0.0);

            for (Line line: lineArray){

                if (line.getStart() == start || line.getEnd() == end){

                    line = null;

                }

            }

        }

# Exercise 3: ArrayLists of Objects

Redo exercise 2, but with the **Line** objects stored in a java.util.ArrayList instead of a raw array of **Line** objects. So the methods become:

**public Line longestLine( ArrayList<Line> lineList ):**

Returns the **Line** with maximum length. If there are multiple **Lines** that tie for the longest length, it may return any one of those **Lines**.

**public void deleteOriginLines( ArrayList<Line> lineList ):**

Deletes every **Line** that starts or ends at point (0,0).

*(2 points per method)*