Midterm 2

CS 1324, Spring 2018

Name (printed legibly):

Student number:

Integrity Pledge: On my honor, I affirm that I have neither given nor received inappropriate aid in the completion of this exercise.

Signature:

Do not write anything on the back of any page.

If you need additional space, use the blank page at the end of the examination. If you pull your examination apart, put ***all of your examination pages in order*** when you turn them in and turn in all pages.

Answer all programming questions in Java.

Pay careful attention to what is requested: a code fragment (a few lines of code), a method, or a program. Students who write a whole program when a code fragment is requested will waste so much time that they may not complete the examination.

Unless otherwise indicated, each part of a problem is worth the same number of points. Show your work to receive partial credit.

When a variable has a comment like "value assigned elsewhere," it means that the value of the variable has already been created somewhere out of sight (like in Turingscraft). You do not need to prompt the user or get values from the user in this situation.

Since you do not have the whole API available during the examination, it is acceptable to guess at method names, parameters and return values. If your guesses are reasonable—even if not perfect—you will receive credit. For example, if you forget that the String class has a length() method and call it size(), that is fine. If, however, you make up new methods that are not reasonable for the class or that magically solve problems the class cannot solve, you will not get credit.

You do not need to import packages or throw exceptions in any methods.

1. (12 points; 4 points each) Find the logical value (true or false) of each of the statements below. If the code is not legal, say so. Assume the variables are declared and initialized as follows:

```
double candy = 3.9 ;
int bunny = 14;
int[] eggs = {5, 1, 4, 2, 7, 5};
```

a) candy >= 0.0 || candy < 5.0
3.9 >= 0.0 || 3.9 < 5.0
true || true
true

b) 12 <= bunny < 15
12 <= 14 < 15
true < 15
This expression is illegal

c) eggs[0] != eggs[5] && eggs[1]==eggs[4] || eggs[2] == eggs[3] || eggs[0] == eggs[5]
5 != 5 && 1 == 7 || 4 == 2 || 5 == 5
false && false || false || true
false || false || true
false || true
true

2. (6 points) Write a single if/else statement that uses a logical operator that will perform the operation described below. Use the variables declared below.

I like to buy paper towels at Fred's Club, but I don't want to have so many paper towels that I can't store them in my cabinet. My cabinet has enough space for 12 rolls of paper towels, but I already have some rolls stored there. I also only want to buy the towels if they cost less than $0.75 per roll. Write a code fragment that prints out whether I should buy paper towels or not.

```
int rollsInCabinet;         // The number of rolls stored in my cabinet.  Value given elsewhere.
double pricePerPackage; // The price of a package of paper towels, which contains multiple rolls. Value
                            // given elsewhere.
int rollsPerPackage;        // The number of rolls of paper towels in one package. Value given elsewhere.
if(rollsInCabinet + rollsPerPackage < 12 && pricePerPackage/(double) rollsPerPackage < 0.75) {
        System.out.println("Buy paper towels");
}
else {
        System.out.println("Do not buy paper towels");
}
```

3. (15 points; 10 points for index, count and temp, 5 points for the contents of the rain array) Trace the loop below in the tables. Remember to show initial and final values.

```
double[] rain = {4.2, 3.1, 0.4, 0.0, 0.0, 1.4, 3.1};


// READ CAREFULLY—not the usual for loop
for(int index= 0; index<rain.length; index = index + 2)
{
      if (rain[index] < 2.0)
      {
            double temp = rain[index];
            rain[index] = rain[0];
            rain[0] = temp;
      }
}
```

index	temp	rain[0]	rain[1]	rain[2]	rain[3]	rain[4]	rain[5]	rain[6]
0		4.2	3.1	0.4	0.0	0.0	1.4	3.1
2	0.4	0.4		4.2				
4	0.0	0.0				0.4		
6								
8								

4. (5 points) The paragraph below describes a static method.

The method will take a given positive integer and create an array that is initialized with integers (starting at 1) and increasing by one with each index.

For example, if the given positive integer is 5, an array containing {1, 2, 3, 4, 5} would be returned.

Write the signature of the method. The method signature contains a return type, the method name and any necessary parameter(s).

int[] consecutiveArray(int value)

5. (5 points) Suppose there is a method called nCopies. This method is given two integer parameters (size and init) and an array reference and changes the first size values in the array to init. For example, if the array contained {1, 2, 3, 4, 5}, size is 3, and init is 7, the method would return an array that contains {7, 7, 7, 4, 5}. The signature of the method is below.

void initializeArray(int size, int init, int[] data)

Write a few lines of code that call this method as shown in the example above.

int[] data = {1, 2, 3, 4, 5};

initializeArray(3, 7, data);

6. (5 points) What will the value of index be after the code below is run?

```
int[] array = {2, 4, 5, 1, 8, 9, 7};
Arrays.sort(array);
int index = Arrays.binarySearch(array, 3);
```

After sort, array contains: 1, 2, 4, 5, 7, 8, 9

3 would be inserted where 4 is at index 2.

The method therefore returns -2-1 = -3

5. (12 points) Answer the questions in the box below about program execution. You may use a memory diagram to trace the code if you wish to, or may use your knowledge of passing by value and/or sharing.

```
public class SecondMidterm
{
	public static void main(String[] args)
	{
		int[] start = {9, 1, 7, 2};
		int factor = 2;
		mystery(start, factor);
	}

	public static void mystery(int[] data, int mult)
	{
		mult = mult + 1;

		for (int index=0; index < data.length; ++index)
		{
			data[index] = mult + index;
		}
		data = new int[mult];
		for (int index = 0; index < data.length; ++index)
		{
			data[index] = mult;
		}
	}
```

At the end of the program,
The contents of array start (not the reference) are (8 points): {3, 4, 5, 6}
factor contains (4 points): 2

main stack frame

Identifier	Address	Contents
start	100	1000
factor	101	2
	102	
	103	

mystery stack frame

Identifier	Address	Contents
data	200	~~1000~~ 1004
mult	201	~~2~~ 3
	202	
	203	
	204	

heap

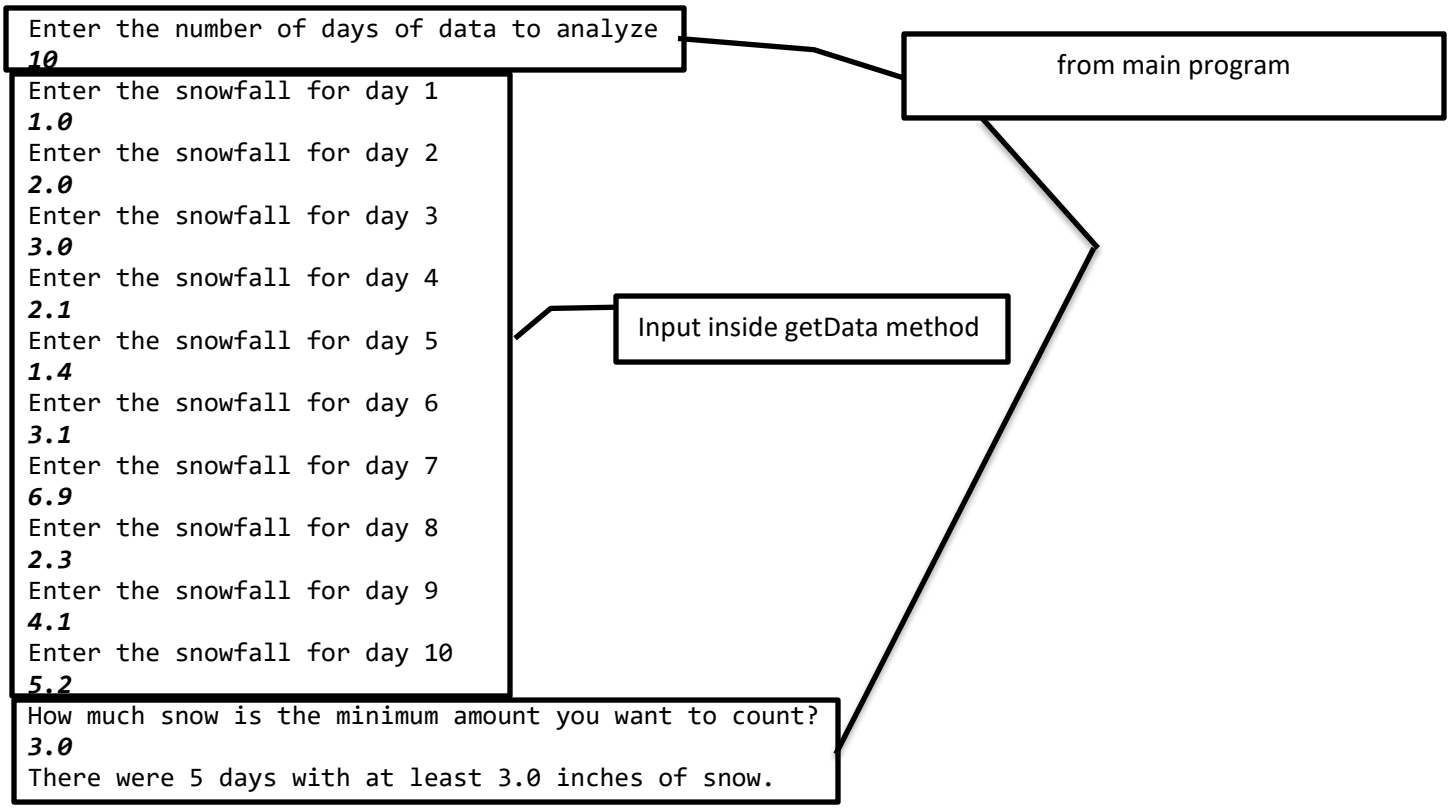
Identifier	Address	Contents
0	1000	~~9~~ 3
1	1001	~~1~~ 4
2	1002	~~7~~ 5
3	1003	~~2~~ 6
length	1004	4
0	1005	~~0~~ 3
1	1006	~~0~~ 3
2	1007	~~0~~ 3
length	1008	3
	1009	
	1010	
	1011	
	1012	

6. (40 points; 13 points for a), 13 points for b), 14 points for c)) Write three methods that together form a program to solve the problem below.

The east coast of the United States has recently experienced three strong storms (called Nor'easters) right in a row. This has piqued your interest in analyzing weather data.

You will write a program that allows the user to enter the amount of snow that fell on a given number of days. Your program should then allow the user to enter an amount of snow and have it count how many days had more snow than the entered amount.

A sample run of the program is below (the data in bold italics was entered by the user)



The program will have three methods:

public static double[] getData(Scanner keyboard, int days)

The data in the box above is entered inside this method.

public static int countMinimumSnowfall(double[] snow, double min)

This method has no input or output statements.

and a **main program**.

The number of days and minimum amount of snowfall are entered in the main program.

a) Write the public static double[] getData(Scanner keyboard, int days) method. This method allows the user to enter snowfall data for each day, storing the result in an array with days elements. The array reference should be returned.

```
public static double[] getData(Scanner keyboard, int days)
{
   double[] snowfall = new double[days];

   for (int index=0; index<snowfall.length; ++index)
   {
         System.out.println("Enter the snowfall for day " + (index+1));
         double snow = keyboard.nextDouble();
         snowfall[index] = snow;
   }

   return snowfall;
}
```

b) Write the countMinimumSnowfall method below. The method should take an array that contains daily snowfall and a minimum value. The method should return the number of days of snowfall that were greater than that minimum value.

```
public static int countMinimumSnowfall(double[] snow, double min)
    {
          int count = 0;

          for (int index = 0; index<snow.length; ++index)
          {
                if (snow[index] >= min)
                {
                      ++count;
                }
          }

          return count;
    }
```

c) **Write the main program.** The main program should call the methods you wrote in a) and b). Their signatures are below:

public static int countMinimumSnowfall(double[] snow, double min)

public static double[] getData(Scanner keyboard, int days)

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

	System.out.println("Enter the  number of days of data to analyze");
	int days = keyboard.nextInt();

	double [] snow = getData(keyboard, days);

	System.out.println("How much snow is the minimum amount you want "
		+"to count?");
	double min = keyboard.nextDouble();

	int count = countMinimumSnowfall(snow, min);

	System.out.println("There were " + count + " days with at least " + min

		+ " inches of snow.");

	keyboard.close();
	}
```

Do not write anything on the back of any page