Midterm 2

CS 1323-4, Fall 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. (8 points; 4 points each) Trace the loops below in the tables on the right. If the loop is an infinite loop, trace three iterations and write "Infinite Loop" in the table.

a)
```
int index = 0;
int fib = 1;
int accumulator = 0;
while (index < 3)
{
        index = index + 1;
        fib = fib + fib;
        accumulator = accumulator + fib;
}
```

index	fib	accumulator
0	1	0
1	2	2
2	4	6
3	8	14

b)
```
int count = 11;
int sum = 0;
while (count != 0)
{
        count = count – 2;
        sum = sum + count;
}
```

count	sum
11	0
9	9
7	16
5	21
Infinite	Loop

2. (10 points) Write a loop that will print out the next to the last word entered by the user before "QUIT" is entered. For example, if the user entered: "Try untried circuitry QUIT", "untried" would be printed out. You do not need to prompt the user. Use the Scanner declared below.

Scanner input; // constructed elsewhere

```
String previousPrevious = "";
String previous = "";
String data = input.next();
while (! data.equals("QUIT"))
{
      previousPrevious = previous;
      previous = data;
      data = input.next();
}
System.out.println(previousPrevious);
```

Do not write anything on the back of any page

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.

```
int[] ord = {5, 6, 3, 1, 2, 0, 4, 7};
int target = 0;

for (int index = 1; index < ord.length; ++index)
{
       if (ord[index] < ord[target])
       {
              target = index;
       }
}

int temp = ord[target];
ord[target] = ord[0];
ord[0] = temp;
```

index	target	ord[0]	ord[1]	ord[2]	ord[3]	ord[4]	ord[5]	ord[6]	ord[7]	ord[8]	temp
1	0	5	6	3	1	2	0	4	7		
2	2										
3	3										
4											
5	5										
6											
7											
8		0					5				0

4. (5 points) The paragraph below describes a static method. Write the signature of the method.

The method will take an array of int, and place the contents of the array in random order. The array should not be reconstructed. For example, if the array originally contained {7, 1, 5, 2, 4}, after the method the same array could contain {4, 2, 1, 7, 5}.

public static void randomOrder(int[] source)

5. (5 points) The Java API below describes the copyOfRange() method from the Arrays class

copyOfRange

```
public static int[] copyOfRange(int[] original,
                                int from,
                                int to)
```

Copies the specified range of the specified array into a new array. The initial index of the range (`from`) must lie between zero and `original.length`, inclusive. The value at `original[from]` is placed into the initial element of the copy (unless `from == original.length` or `from == to`). Values from subsequent elements in the original array are placed into subsequent elements in the copy. The final index of the range (`to`), which must be greater than or equal to `from`, may be greater than `original.length`, in which case `0` is placed in all elements of the copy whose index is greater than or equal to `original.length - from`. The length of the returned array will be `to - from`.

Parameters:

`original` - the array from which a range is to be copied

`from` - the initial index of the range to be copied, inclusive

`to` - the final index of the range to be copied, exclusive. (This index may lie outside the array.)

Returns:

a new array containing the specified range from the original array, truncated or padded with zeros to obtain the required length

Call this method to create a new array with a copy of only the positive values from the array declared below stored in the same array reference.

int[] array = { -2, -4, 7, 8, 1, 4, 2, 5, -9, -7};

array = Arrays.copyOfRange(array, 2,8);

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

double[] array = {9.5, 2.4, -3.1, 4.7, 7.2, 8.3, 5.6}
Arrays.sort(array); // See API for details
int index = Arrays.binarySearch(array, 4.7);

After sort, array contains: {-3.1, 2.4, 4.7, 5.6, 7.2, 8.3, 9.5}

Index contains 2 (the index where the value is found)

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 MemoryDiagrams {

	public static void main(String[] args) {
		String[] input = {"Shrek", "Donkey", "Fiona"};
		String[] output = mystery(input);
	} // end of main

	public static String[] mystery(String[] source) {
		String[] result = new String[source.length];
		for(int index = 0; index<source.length; ++index) {
			result[index] = source[source.length-index-1];
		}

		return result;
	} // end of mystery
} // end of class MemoryDiagrams
```

At the end of the program,
The contents of array input (not the reference) are:
The contents of the array output (not the reference) are:

main stack frame

Identifier	Address	Contents
input	100	1000
output	101	1004
	102	
	103	

mystery stack frame

Identifier	Address	Contents
source	200	1000
result	201	1004
	202	
	203	
	204	

heap

Identifier	Address	Contents
0	1000	~~Shrek~~ “”
1	1001	~~Donkey~~ “”
2	1002	~~Fiona~~ “”
length	1003	3
0	1004	Fiona
1	1005	Donkey
2	1006	Shrek
length	1007	3
	1008	
	1009	
	1010	
	1011	
	1012	

6. (40 points; 10 points for a), 15 points for b), 15 points for c)) Write three of the five methods that together form a program to solve the problem below.

WildCare is an organization near Norman. They rehabilitate sick and injured wild animals so they can be returned to the wild to live out their natural lives. They need some software to help them manage their front desk. Their front desk needs to accomplish two things: they need the species of the animals that are admitted to their care and they need to be able to find the number of animals from any given species. The interface is shown below. The original contents of the AdmittedPatients.txt file are in the top box on the right. The final contents of the AdmittedPatients.txt file are in the bottom box on the right. User entries are in italics.

```
Enter 1 to enter a new patient
Enter 2 to count patients by species
Enter 3 to quit          // Replaced by MENU below
2
Which species would you like to count?
fox squirrel
We have had 4 fox squirrel(s) admitted this year.
MENU
1
Enter the species of the new patient
fox squirrel
MENU
2
Which species would you like to count?
fox squirrel
We have had 5 fox squirrel(s) admitted this year.
MENU
2
Which species would you like to count?
mole
We have had 1 mole(s) admitted this year.
MENU
2
Which species would you like to count?
cormorant
We have had 0 cormorant(s) admitted this year.
MENU
1
Enter the species of the new patient
red squirrel
MENU
3
```

8
Fox squirrel
mole
red squirrel
Red tailed hawk
fox squirrel
raccoon
fox squirrel
fox squirrel

10
Fox squirrel
mole
red squirrel
Red tailed hawk
fox squirrel
raccoon
fox squirrel
fox squirrel
fox squirrel
red squirrel

The method signatures are given below and will be explained in the individual parts

int countSpecies(String[] admitted, String species)
String[] admitNewPatient(String[] admitted, String species)
String[] readFile(String fileName) // you won't write this
String[] writeFile(String[] admits, String fileName) // you won't write this

a) (10 points) Write the countSpecies method. This method takes a String array that contains the species of all admitted patients and the name of a given species and returns the number of times that species occurred in the array. You may assume that data are entered perfectly (i.e. no extra spaces or hyphens) except that capitalization may vary, as was shown in the introduction to the problem.

```
public static int countSpecies(String[] data, String species)
{
	int count = 0;
	for (int index =0; index<data.length; ++index)
	{
		if (data[index].equalsIgnoreCase(species))
		{
			++count;
		}
	}

	return count;
}
```

b) (15 points) Write the admitNewPatient method. This method takes a given perfect sized array of current patients, and the given species of a new patient and creates a new perfect sized array with the current patient appended to the end. For example, if the original array contained {“fox squirrel”, “red squirrel”, “red tailed hawk”} and a “falcon” was admitted, the returned array would contain {“fox squirrel”, “red squirrel”, “red tailed hawk”, “falcon”}.

```
public static String[] admitNewPatient(String [] patients, String species )
{
      // Construct the array to return--need one more space
      String[] newPatients = new String[patients.length +1];

      // Move the data from the original array to the new array
      int index;
      for (index = 0; index < patients.length; ++index)
      {
            newPatients[index] = patients[index];
      }

      // Put on the last data element
      newPatients[index] = species;

      // Return the new array reference
      return newPatients;
}
```

c) (15 points) Write the main program. You must use the methods from a) and b) and two additional methods that were written by someone else. These methods read and write the file that stores the data between program runs. The method signatures are below. All input and output is done in this method.

int countSpecies(String[] admitted, String species) // from a)
String[] admitNewPatient(String[] admitted, String species) // from b)
String[] readFile(String fileName) // reads in the original perfect sized array from a file of a given name
void writeFile(String[] admits, String fileName) // writes a perfect sized array to a file of a given name

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

	String [] admits = readFile("AdmittedPatients.txt");

	//Priming read
	System.out.println("Enter 1 to enter a new patient");
	System.out.println("Enter 2 to count patients by species");
	System.out.println("Enter 3 to quit");

	int choice = keyboard.nextInt();
	keyboard.nextLine();  // this is necessary
	// Sentinel controlled loop
	while (choice != 3)
	{
		if (choice ==  1)
		{
			System.out.println("Enter the species of the new patient");
			String species = keyboard.nextLine();
			admits = admitNewPatient(admits, species);
		}
		else if (choice == 2)
		{
			System.out.println("Which species would you like to "
				+ "count?");
			String species = keyboard.nextLine();
			System.out.println("We have had "
				+ countSpecies(admits, species) + " " + species
				+ "(s) admitted this year.");
		}

		//Priming read
		System.out.println("Enter 1 to enter a new patient");
		System.out.println("Enter 2 to count patients by species");
		System.out.println("Enter 3 to quit");

		choice = keyboard.nextInt();
		keyboard.nextLine();  // this is necessary
	}

	writeFile(admits, "AdmittedPatients.txt");
	keyboard.close();

}
```