Midterm 3

CS 1323, Fall 2016

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 extra space use the blank page at the back of the examination. If you pull the test apart, please put all of the pages in order when you hand your exam in.

Answer all programming questions in Java. Show your work to receive partial credit.

Pay careful attention to what is requested: a code fragment (a few lines of code), a method, or a program.

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. 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.

Relevant parts of the API are given in a separate document.

1. (9 points; 3 points each)

a) What does the String object direction contain after the following code is executed?

```
String direction = "North";
direction.replace('N', 'S'); // API in other document
direction.replace('r', 'u');
```

b) What does the StringBuilder object direction contain after the following code is executed?

```
StringBuilder direction = new StringBuilder("North");
direction.replace(0, 3, "Sou");  // API in  other document
```

c) Suppose that there were a method in the String class that returns a String object that contains n copies of its current contents. For example, three copies of “ab” would be “ababab”. For the method signature below, determine whether the method is a class method (static) or an instance method and ***circle the appropriate answer***

String copies (int n)

class method instance method

2. (10 points; 3 points for algorithm, 7 for sorting) Sort the values below using either insertion sort or selection sort. You must indicate which algorithm you used and show the process exactly as it was in class, with each individual swap or move of data shown on a separate line. **Only data that are moved or changed should be shown.**

I used (circle one): selection sort insertion sort

3	5	2	7	4	6	Aux

Do not write anything on the back of any page

3. (10 points) Trace the code fragment below in the table below.

```
String result = "";
String[] source = {"abra", "cadabra", "ally", "kazam"};
for (int outer = 0; outer<source.length; ++outer)
{
	char outerC = source[outer].charAt(0);

	for(int inner = outer+1; inner < source.length; ++inner)
	{
		char innerC = source[inner].charAt(0);

		if (outerC != innerC)
		{
			result = result + outer;
		}
	}
}
```

outer	outerC	inner	innerC	result

4. (11 points; 2 points each for a) to d), 3 points for e))

Use the documentation for the SecureRandom class (given in a separate document) to answer the following questions and perform the following tasks.

In parts a) to d), if there are no methods of the required type, say “None.”

a) List one constructor in the SecureRandom class

b) List one mutator method in the SecureRandom class.

c) List one accessor method in the SecureRandom class.

d) List one class method in the SecureRandom class.

e) “PKCS11” is a legal name for a SecureRandom number generation algorithm. Use this information to create a SecureRandom object that uses this algorithm.

5. (40 points; 15 points for a), 10 points for b) and 15 points for c))

Oklahoma Electric and Gas (OEG) collects data from the thermostats in customer homes to help them manage their business. They are constantly analyzing the data to determine whether they are generating the right amount of power. If they generate too little power, they risk brown-outs. If they generate too much power, they waste resources and make power more expensive for everyone.

The thermostats return data in the following format: time (hours: minutes), Customer ID # (5 digits), thermostat setting in degrees. Another program puts all of this data into a file every two minutes. A sample file is shown below.

```
2:10 12345 75
2:10 23456 76
2:10 34567 77
2:10 45678 78
2:11 67897 75
2:11 98765 74
2:12 87654 72
```

Write a program to help OEG manage power. Your program should read in a file of customer data to an ArrayList<String> object, create a new ArrayList<Integer> of temperatures that are greater than a given temperature, and report the median temperature from the ArrayList<Integer>.

The median temperature is the middle temperature. The easiest way to get it is to put the elements in the ArrayList<Integer> in ascending order, and find the element(s) in the middle. If the ArrayList<Integer> has an odd length, there is only one middle element. If the ArrayList<Integer> has an even length, then the middle two elements are averaged.

A sample program run is shown below (using the data in the file above).

```
Enter the target temperature
76
Do you want temperatures above or below this temperature?
above
The median temperature is 77.5
```

There is a utility method (written by someone else, but available to you) that will get the temperature from a line in this format.

public static int getThermostat(String line)

Your program will have five methods (including the main and the method above), only three of which need to be written for this examination. The signatures of the non-main methods are below. You may not modify these signatures in any way.

void addCurrentResults(String fileName, ArrayList<String> data) // You won't write this
ArrayList<Integer> getExtremeTemperatures(ArrayList<String> data, int temperature, boolean above)
double getMedianTemperature(ArrayList<Integer> data)

a) Write the getExtremeTemperatures method. This method will take an ArrayList<String> and create a new ArrayList<Integer> object that contains only the temperature values from customers with thermostats set to values that are greater than (if above is true) or less than (if above is false) the given temperature value. For example, if the file were as given in the problem statement, temperature was 76, and above was true (as in the example given in the problem description), the returned ArrayList<Integer> would contain 77 and 78.

This method will use the getThermostat method that takes a String like "2:10 12345 75" and returns 75. The signature is below. Do not write this method.

```
public static int getThermostat(String line) // do not write this method

public static ArrayList<Integer> getExtremeTemperatures(ArrayList<String> data, int temperature,
        boolean above)  //write this method

{
```

b) Write the getMedianTemperature method. This method should take an ArrayList<Integer> and return the median value. This should be accomplished by putting the data in the ArrayList in ascending order and finding the middle element. If the ArrayList has an odd length there is only one middle element. If the ArrayList is of even length, you average the middle two elements. This can be done using a method in the Collections class. This method should be short (probably less than ten lines).

```
public static double getMedianTemperature(ArrayList<Integer> data)

{
```

c) Write the main program. You must use the methods from a) and b), and the other methods exactly as described. The method signatures are given below. The data is stored in a file called “Data.txt”.

void addCurrentResults(String fileName, ArrayList<String> data)
ArrayList<Integer> getExtremeTemperatures(ArrayList<String> data, int temperature, boolean above)
double getMedianTemperature(ArrayList<Integer> data)

Do not write anything on the back of any page