Midterm 3 Solution

CS 1323, 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 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 attention to whether the program requires you to use an ArrayList or an array.

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 on the back of previous pages and in a separate document.

1. (15 points; 3 points each for a) and b); 9 points for c))

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

```
String game = new String("Baseball");
game.substring(0,4); // read the API on back of previous page carefully
```

Baseball

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

```
StringBuilder game = new StringBuilder("Softball");
game.substring(0,4); //Read the API on the back of previous page carefully
```

Softball

c) Write a method that returns true if any String in a perfect size array of Strings starts with a given prefix. So if the String array contained: {"jolly", "green", "giant"} and the prefix is "gr" the method should return true. If the prefix were "ja", the method returns false.

The signature is below:

public static boolean anyStartsWith(String[] array, String prefix)

The API for a useful method in the String class is given on the back of the previous page.

```
public static boolean anyStartsWith(String[] array, String prefix)
{
      for (int index = 0; index < array.length; ++index)
      {
            String element = array[index];
            if (element.startsWith(prefix)
                  return true;
      }

      return false;

}
```

2. (15 points; 5 points for algorithm, 10 for sorting values) 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

Selection sort

3	**9**	**1**	**2**	**6**	**4**	**Aux**
1		3				
	2		9			
		3				
			4		9	
				6		

Insertion sort

3	**9**	**1**	**2**	**6**	**4**	**Aux**
						9
	9					
						1
		9				
	3					
1						
						2
			9			
		3				
	2					
						6
				9		
			6			
						4
					9	
				6		
			4			

3. (15 points) Trace the code fragment below in the table below. Only show changes to the data.

```
int[] data = {4, 6, 1, 9, 3, 8};
for(int start = data.length-1; start > 0; --start)
{
      int temp = start;
      for (int find = start-1; find > 0; --find)
      {
            if (data[find] > data[temp])
            {
                  temp = find;
            }
      }// end for find

      int other = data[temp];
      data[temp] = data[start];
      data[start] = other;
} // end for start
```

start	temp	find	other	data[0]	data[1]	data[2]	data[3]	data[4]	data[5]	data[6]
5	5	4		4	6	1	9	3	8	
		3								
	3									
		2								
		1								
		0								
			9				8		9	
4	4	3								
	3	2								
		1								
		0								
			8				3	8		
3	3	2								
	1	1								
		0								
			6		3		6			
2	2	1								
	1	0	1		1	3				
1	1	0								
	0									

4. (15 points;6 points for a), 3 points for b), 6 points for c)) Use the documentation for the Logger class (given in a separate document) to answer the following questions and perform the following tasks.
   a) Construct a Logger object with reference search, name "SearchLog" and level 500.

   b) Change the level on a Logger object with reference insertLog to 800.

   c) Modify the method below to use the Logger object from the first line of the method body to create entering and leaving messages and to log each loop iteration at level 200.

```
public static boolean search(ArrayList<String> list, String target) {
        Logger search = getLogger("SearchLog");  // returns Logger constructed in a)
// Log entering method at level 200



        for (int index = 0; index< list.size(); ++index) {
// Log each index at level 200



                if (list.get(index).equals(target)) {
// Log leaving method at level 200



                        return true;
                }
        }
// Log leaving method at level 200



        return false;
}
```

5. (40 points; 10 points for a) , 15 points for b), 15 points for c)) A "bucket list" is a list of things you want to do in your life (i.e. before you "kick the bucket"). Many people keep these lists, adding and removing elements as they select, attain and adjust their life priorities. A short bucket list is below (some of these lists have thousands of entries).

Sky dive
Attend a political protest march
Visit Africa

You can do three critical operations on a bucket list. First, you can add new items. Items are added at the end of the list by default. Adding "Ride in hot air balloon" to the list above it will look like this. If an item is already on the list it should not be added again.

Sky dive
Attend a political protest march
Visit Africa
Ride in hot air balloon

You can also remove items from the list. If I went skydiving, the list would be below. If the user asks you to remove an item that is not on the list, do nothing. Do not leave gaps in the list.

Attend a political protest march
Visit Africa
Ride in hot air balloon

You also need to be able to display the list to the user (as was done above), and the list needs to be written to a file and read from a file so it can be permanently stored.

The bucket list is stored in an oversize array. You may assume that bucket lists are less than 1,000 items in length.

The user interactions with the program are on the next page (with annotations). The menu of commands is below. I'll abbreviate this as "**Menu Displayed**" to save space.

Choose from the following menu items:
1. Add items to bucket list
2. Remove items from bucket list
3. Show the bucket list
4. Quit

```
                                        // The bucket list is initially empty
Menu displayed                          // The first three operations add items to bucket list
1
What do you want to add to your bucket list?
Trek Inca trail
Menu displayed
1
What do you want to add to your bucket list?
Run marathon
Menu displayed
1
What do you want to add to your bucket list?
Travel to India
Menu displayed                          // Show the bucket list
3
Trek Inca trail
Run marathon
Travel to India
Menu displayed                          // Remove an item from the bucket list
2
What do you want to remove from your bucket list?
Run marathon
Menu displayed                          // Show the bucket list again
3
1rek Inca trail
Travel to India
Menu displayed                          // Now quit
4
```

a) Write the method that adds an item to the end of the bucket list, if the item is not already on the list. The bucket list is stored in an oversized array. **If the item is already in the bucket list it should not be added again.** The parameter list contains the reference to the oversized array, listSize is the number of elements in the list that are currently active, and item is the item to be added. This method should not do any input or output. The method returns the number of items in the bucket list after the addition.

public static int addItem(String[] list, int listSize, String item)

```
public static int addItem(String[] list, int listSize, String item)
{
        // Return if item is already in list
        for (int index = 0; index < listSize; ++index)
        {
                if (list[index].equals(item))
                        return listSize;
        }

        // If the item is not in the list, add it to the end
        list[listSize] = item;
        ++listSize;

        return listSize;
}
```

b) Write a method that removes an item from the bucket list. When an item is removed, the items at bigger positions are shifted so there are no gaps in the list. So if a list contained ["A", "B", "C", "D"] and we remove "B", the list will contain ["A", "C", "D"]. If the item is not in the bucket list, nothing should be done. The method signature is below. The parameter list contains a reference to the oversized array containing the bucket list. The parameter listSize is the number of elements in the bucket list that are currently active. The parameter item contains the item to be removed from the list. The method returns number of items in the list after the item is removed. This method should not do any input or output.

```
public static int removeItem(String[] list, int listSize, String item)
```

```
public static int removeItem(String[] list, int listSize, String item)
{
        //Locate the item in the list
        int removeIndex = -1;
        for (int index = 0; index < list.length && removeIndex < 0; ++index)
        {
                if (list[index].equals(item))
                        removeIndex = index;
        }

        // The item was not in the list, can't remove it, get out of here
        if (removeIndex < 0)
                return listSize;

        // Shift things in the list over to remove the item.
        for (int index = removeIndex; index < list.length-1; ++index)
        {
                list[index] = list[index+1];
        }

        return listSize -1; // If we get here, this item was removed
}
```

c) Complete the main program below by filling in code below the bold, italisized, underlined, comments. Some method calls in the code ar3 replaced by comments that say "Method call to <method name> concealed."

```
public static void main(String[] args) throws FileNotFoundException {
        final int SIZE = 1000;
        final String FILENAME = "BucketList.txt";
        Scanner keyboard = new Scanner(System.in);

// Create an oversize array. (5 points)




        // Method call to read file concealed
        int choice = getMenuChoice(keyboard);
        while (choice != 4) {
                if (choice == 1) {
// Get user input and call the add method from a) (5 points)








                }
                else if (choice == 2) {
// Get user input and call the remove method from b) (5 points)








                }
                else if (choice == 3) {
                        // Method call to print file concealed
                }

                choice = getMenuChoice(keyboard);
        } // end while

        // Method call to write file concealed
        keyboard.close();
```

Do not write anything on the back of any page