

# Midterm 3

---

*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 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) or a method.

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 imperfect—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. (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 carefully
```

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 carefully
```

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 in a separate document.

- I used (circle one):      selection sort                  insertion sort

[illegible]

```
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
```

[illegible]

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 that has been declared and constructed in the first line of the method body to create entering and leaving messages at level 500 and to log each loop index at level 200.

```
public static boolean search(ArrayList<String> list, String target) {  
    Logger search = Logger.getLogger("SearchLog"); // returns Logger constructed in a)  
    // Log entering method at level 500  
  
    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 500  
  
            return true;  
        }  
    }  
    // Log leaving method at level 500  
  
    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 perform three operations on a bucket list. First, you can add new items. Items are added at the end of the list. If an item is already on the list it should not be added again. If you add "Ride in hot air balloon" to the list above, it will look like this.

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, which removes it from the bucket list, the list would be as 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

```
Menu displayed                                // The bucket list is initially empty
1                                                    // The first three operations add items to bucket list
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 method signature is below. 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. The method returns the number of items in the bucket list after the addition. **This method should not do any input or output.**

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



b) Write a method that removes an item from the bucket list. When an item is removed, the items at bigger positions can be shifted or swapped so there are no gaps in the list. Suppose we have a list that contains ["A", "B", "C", "D"] and we remove "B". If we use shifts the list will contain ["A", "C", "D"]. If we want to use a swap, the last element in the list will swap with the removed one, so the list will contain ["A", "D", "C"]. Either method can work. **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)
```

c) Complete the main program below by filling in code below the bold, italicized, underlined, comments. Some method calls in the code are replaced by comments that say "Call to <method name> concealed." The getMenuChoice method prints out the menu and returns a legal menu choice.

```
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)  
  
    // 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) {  
            // Call to print list concealed  
        }  
  
        choice = getMenuChoice(keyboard);  
    } // end while  
  
    // Call to write file concealed  
    keyboard.close();  
}
```

