EECS 1022 Labtest 3 Version B

Friday November 30, 2018, 17:00-18:30

Instructions

This labtest is closed book, no aids allowed. You must work individually. You are not allowed to browse any documents on the web other than those provided in this labtest page.

There is a set of multiple-choice questions and a set of programming questions. You will fill your answers to all these questions into a Java class called `Utilities.java` which you will then submit. Here is how to proceed:

- Launch Android Studio and open the **existing project** called `Labtest3B`.
- Select the Java class `Utilities.java` in the editor window; it contains the starter code.
- Then go over the questions below and enter your answers into the `Utilities.java` file and save your changes.
- Test your solutions by running the `UtilitiesTester.main()` method (right click on the `UtilitiesTester.java` class in the project window and select to run `UtilitiesTester.main()`). Your output should be as shown here.
- When you are ready, use Web Submit to submit your modified `Utilities.java`. To do this, first open the link https://webapp.eecs.yorku.ca/submit in your browser, login using your Passport York account, select your course `1022`, select the assignment `labtest3b`, navigate to and choose the file `Utilities.java` (it should be in folder `AndroidStudioProjects/Labtest3B/app/src/`), and then click the "Submit files" button at the bottom of the page. If you have successfully completed this task, you will see a message saying that you have submitted the file `Utilities.java` with the submission time.
- You can submit as many times as wou want and only the last version submitted will be graded.
- **You must submit your answers before the labtest period ends at 18:30; submission will be disabled after that time.**
- **Check to make sure that your submitted file appears in the list of submitted files in WebSubmit. Also check that you have submitted the right version (you can download it and open it to check).**
- **Make sure that your `Utilities.java` class compiles without errors. If there are compilation errors, you may get a grade of 0.**
- Here, you can find a table of common Java regular expressions constructs.
- The Java API is here .

Multiple Choice Questions

For these questions, enter your answers into the `Utilities.java` file by editing the method corresponding to the question, replacing the `?` in the printed string by your answer, a, b, c, d, or e.

Question 1

Which of the following is true?

a. Both `List<String>` and `Set<String>` are ordered collections.
b. `List<String>` represents ordered collections but `Set<String>` represents unordered collections.
c. `Set<String>` represents ordered collections but `List<String>` represents unordered collections.
d. Neither `List<String>` nor `Set<String>` represent ordered collections.
e. None of the above.

Question 2

Which of the following is false?

a. Both `ArrayList<String>` and `LinkedList<String>` implement all the methods in the `List<String>` interface.
b. `ArrayList<String>` and `LinkedList<String>` use different representations in their implementations of lists of strings.
c. If we create an empty `ArrayList<String>` and an empty `LinkedList<String>` and add at the end of each list the same 3 strings in the same order, the resulting lists will be `equal`.
d. If we create an empty `ArrayList<String>` and an empty `LinkedList<String>` and add at the end of each list the same 3 strings in the same order, `toString()` will return the same result for both lists.
e. None of the above.

Question 3

Which of the following statements about a `Map<Integer,String>` is false?

a. Such a map contains a set of pairs *(i,s)* where *i* is an integer and *s* is a string.
b. For any given key in the map *i*, there is a unique value *s*.
c. There may be integers for which the map does not assign a value.
d. For any given value in the map *s*, there is a unique key *i* such that the map associates *s* to *1*.
e. None of the above.

Question 4

Suppose that we want to implement a method to check whether a `List<String>` contains duplicates. Which of the following approaches could *not* be used to implement this method?

a. Copy the elements of the list into a `Set<String>` and check if `add` returns true each time.
b. Sort the list using `Collections.sort` and then traverse the list checking if adjacent elements are the same.
c. For each element of the list, traverse the entire list to see if another element at a different position is equal.
d. Copy all the elements of the list into a `Set<String>` and check if the size of the list and set are the same at the end.
e. None of the above.

Question 5

Which of the following strings does *not* contain a match for the regular expression `xy*$`

a. `"xy"`

b. `"xxy"`
c. `"xyy"`
d. `"x"`
e. `"y"`

Programming Questions

Question 6

Go to the `Utilities.java` class and implement the method

```
void increment(int k, List<Integer> l)
```

Given an integer `k` and a set of integers `s`, remove `k` from the set `s` and add `k + 1` to `s`; if `k` is not an element of `s`, the method leaves `s` unchanged.

For example, if initially `k` has the value `56` and `s` has the value `[23, 37, 56, 89]`, then after executing `increment(k,s)`, `s` will have the value `[23, 37, 57, 89]`.

You can assume that the `s` argument is not `null`. There is no need to check for it.

You can see some examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.

Question 7

In the `Utilities.java` class, implement the method

```
int getPos(String s, List<String> l)
```

Given a string `s` and a list of strings `l`, return the position of the first occurrence of `s` in `l`; if `s` does not occur in `l`, return `-1`.

For example, if `s` has the value `Mary` and `l` has the value `[John, Mary, Paul, Mary, Helen]`, then the method returns the value `1`.

You can assume that the arguments are not `null`. There is no need to check for it.

You can see some examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.

Question 8

In the `Utilities.java` class, implement the method

```
String findFlightNo(String s)
```

Given a string `s`, return the first occurrence of a flight number in `s`; if there is no flight number in `s`, return the empty string.

A flight number is a string of 2 upper case letters followed by 3 or 4 digits.

For example, when `s` has the value `"Flight AC233 arrives at 16:15 and KL0978 arrives at 17:55."`, the returned value should be `AC233`.

You can assume that the argument `s` is not `null`. There is no need to check for it.

You can see some examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.

Question 9

In the `Utilities.java` class, implement the method

```
Set<String> findAllProductIDs(String s)
```

Given a string `s`, return the set of all the product ID strings in `s`; if there is no product ID in `s`, return the empty set.

A product ID is a string of 2 or more digits followed by 3 or more upper case letters, where the first digit can neither be 4 nor 8.

For example, when `s` has the value `"Some possible products IDs are 1ABC, 24ABC, 24AB, 24ABCD, 248ABC, 248ABCD, 44ABC, 88ABC, and 24ABc. Which are valid?"`, the returned value should be `[248ABC, 248ABCD, 24ABC, 24ABCD]` (the order of the elements of the set does not matter).

You can assume that the argument `s` is not `null`. There is no need to check for it.

You can see more examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.

Question 10

In the `Utilities.java` class, implement the method

```
Map<String,Integer> firstOccurrences(List<String> l)
```

Given a list of strings `l`, return a map whose keys are all the distinct strings in `l` and such that the value of the map for each key is the position of the first occurence of the key in the list `l`. If `l` is the empty list, return an empty map.

For example, when `l` has the value `[123, abc, 456, abc, abc, cde]`, the returned value should be `{123=0, abc=1, 456=2, cde=5}` (the order of the pairs in the map does not matter).

You can assume that the argument `l` is not `null`. There is no need to check for it.

You can see more examples in the `UtilitiesTester.java` class and expected output `UtilitiesTesterOutput.txt`.
