EECS 1022 Labtest 3 Version A

Friday November 30, 2018, 10:00-11:30

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 `Labtest3A`.
- 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 `labtest3a`, navigate to and choose the file `Utilities.java` (it should be in folder `AndroidStudioProjects/Labtest3A/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 11:30; 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>` may contain duplicate elements.
b. `List<String>` may contain duplicate elements but `Set<String>` may not.
c. `Set<String>` may contain duplicate elements but `List<String>` may not.
d. Neither `List<String>` nor `Set<String>` may contain duplicate elements.
e. None of the above.

Question 2

Which of the following is false?

a. Both `HashSet<String>` and `TreeSet<String>` implement all the methods in the `Set<String>` interface.
b. `HashSet<String>` and `TreeSet<String>` use different representations of sets of strings.
c. If we create an empty `HashSet<String>` and an empty `TreeSet<String>` and add the same 3 strings to them the resulting sets will be `equal`.
d. If we create an empty `HashSet<String>` and an empty `TreeSet<String>` and add the same 3 strings to them, `toString()` will return the same result for both sets.
e. None of the above.

Question 3

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

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

Question 4

The `Collections.sort` method sorts a list based on natural order. Which of the following statements about natural order is false?

a. For numeric types of elements, the order is based on magnitude.
b. For string elements, the order is lexicographic.
c. For times or date elements, the order is chronological.
d. For other types, the order is arbitrary.
e. None of the above.

Question 5

Which of the following strings does *not* contain a match for the regular expression `^a+b`

a. `"ab"`
b. `"aab"`
c. `"abbb"`
d. `"aabbb"`
e. `"acde"`

Programming Questions

Question 6

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

```
void moveToFront(String s, List<String> l)
```

Given a string `s` and a list of strings `l`, remove the first occurrence of `s` in `l` and add `s` at the beginning of `l`; if `s` does not occur in `l`, the method leaves `l` unchanged.

For example, if initially `s` has the value `abc` and `l` has the value `[123, abc, 456, abc]`, then after executing `moveToFront(s, l)`, `l` will have the value `[abc, 123, 456, abc]`.

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 7

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

```
int getPos(int k, List<Integer> l)
```

Given an integer `k` and a list of integers `l`, return the position of the last occurrence of `k` in `l`; if `k` does not occur in `l`, return `-1`.

For example, if `k` has the value `13` and `l` has the value `[21, 23, 13, 23, 76]`, then the method returns the value `2`.

You can assume that the argument `l` 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 8

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

```
int findPlateNo(String s)
```

Given a string `s`, return the position of the first occurrence of a license plate number in `s`; if there is no license plate number in `s`, return `-1`.

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

For example, when `s` has the value `"The MHD233 and KLTH978."`, the returned value should be `4`.

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

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

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

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

For example, when `s` has the value `"Some possible products IDs are BMX1, BMX02, BM12, BMXX012, BMX123, XMX12, and Abc01. Which are valid?"`, the returned value should be `[BMX02, BMXX012, BMX123]`.

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> countOccurrences(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 number of times that the key occurs 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=1, abc=3, 456=1, cde=1}`.

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`.
