**import** **java.io.IOException**;

**import** **java.util.ArrayList**;

**import** **java.util.HashMap**;

**import** **java.util.HashSet**;

**import** **java.util.LinkedHashMap**;

**import** **java.util.List**;

**import** **java.util.Map**;

**import** **java.util.Map.Entry**;

**import** **java.util.Set**;

/\*\*

\* Java Program to find first duplicate, non-repeated character in a String.

\* It demonstrate three simple example to do this programming problem.

\*

\* @author Javarevisited

\*/

**public** **class** **Programming** {

/\*

\* Using LinkedHashMap to find first non repeated character of String

\* Algorithm :

\* Step 1: get character array and loop through it to build a

\* hash table with char and their count.

\* Step 2: loop through LinkedHashMap to find an entry with

\* value 1, that's your first non-repeated character,

\* as LinkedHashMap maintains insertion order.

\*/

**public** **static** **char** **getFirstNonRepeatedChar**(String str) {

Map<Character,Integer> counts = **new** LinkedHashMap<>(str.length());

**for** (**char** c : str.toCharArray()) {

counts.put(c, counts.containsKey(c) ? counts.get(c) + **1** : **1**);

}

**for** (Entry<Character,Integer> entry : counts.entrySet()) {

**if** (entry.getValue() == **1**) {

**return** entry.getKey();

}

}

**throw** **new** **RuntimeException**("didn't find any non repeated Character");

}

/\*

\* Finds first non repeated character in a String in just one pass.

\* It uses two storage to cut down one iteration, standard space vs time

\* trade-off.Since we store repeated and non-repeated character separately,

\* at the end of iteration, first element from List is our first non

\* repeated character from String.

\*/

**public** **static** **char** **firstNonRepeatingChar**(String word) {

Set<Character> repeating = **new** HashSet<>();

List<Character> nonRepeating = **new** ArrayList<>();

**for** (**int** i = **0**; i < word.length(); i++) {

**char** letter = word.charAt(i);

**if** (repeating.contains(letter)) {

**continue**;

}

**if** (nonRepeating.contains(letter)) {

nonRepeating.remove((Character) letter);

repeating.add(letter);

} **else** {

nonRepeating.add(letter);

}

}

**return** nonRepeating.get(**0**);

}

/\*

\* Using HashMap to find first non-repeated character from String in Java.

\* Algorithm :

\* Step 1 : Scan String and store count of each character in HashMap

\* Step 2 : traverse String and get count for each character from Map.

\* Since we are going through String from first to last character,

\* when count for any character is 1, we break, it's the first

\* non repeated character. Here order is achieved by going

\* through String again.

\*/

**public** **static** **char** **firstNonRepeatedCharacter**(String word) {

HashMap<Character,Integer> scoreboard = **new** HashMap<>();

// build table [char -> count]

**for** (**int** i = **0**; i < word.length(); i++) {

**char** c = word.charAt(i);

**if** (scoreboard.containsKey(c)) {

scoreboard.put(c, scoreboard.get(c) + **1**);

} **else** {

scoreboard.put(c, **1**);

}

}

// since HashMap doesn't maintain order, going through string again

**for** (**int** i = **0**; i < word.length(); i++) {

**char** c = word.charAt(i);

**if** (scoreboard.get(c) == **1**) {

**return** c;

}

}

**throw** **new** **RuntimeException**("Undefined behaviour");

}

}