Software Engineering and Programming Basics - WS2021/22
Assignment 5

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Professorship of Software Engineering

11| 2021

Organisational

Deadline

13.12.2021 - 23:59

Submission

To submit your answers, please use the Task item titled 'Submission' in the menu of the Assignment 05. You can upload your /.java files here. There are sample files shown in Samples section highlighted.

Please remember that package name is the name of assignment, i.e. assignment5.

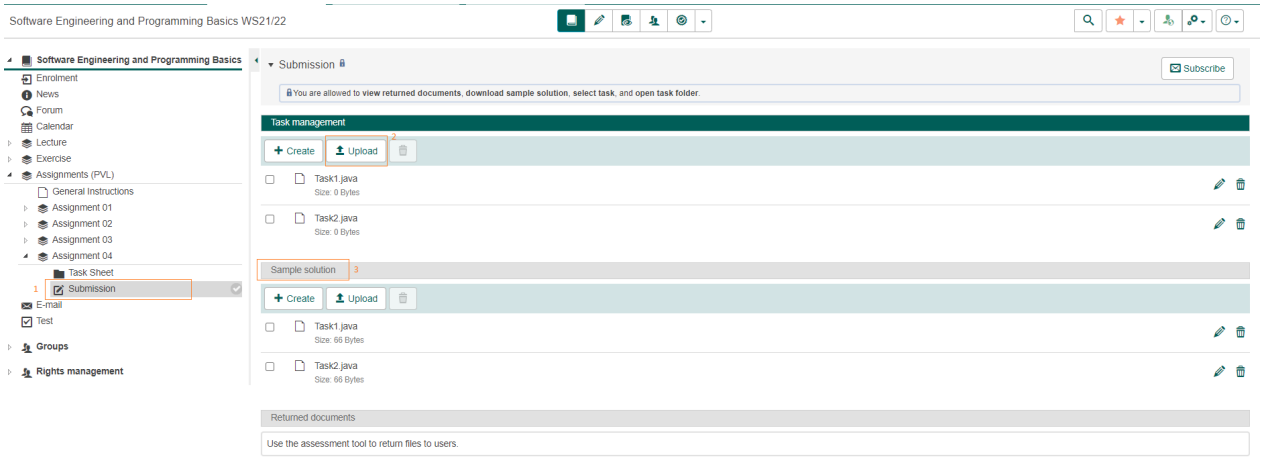
Make sure your files are correctly named: Package, class and function names should be exact same of what is mentioned in the Task Sheet in order to get grades.

This is an automated checking system. If the uploaded files have wrong names then your code will not be graded.

You also need to adhere to the General Assignment Instructions.

Questions

Since this is a PVL, it is important that all students are able to access all necessary information. Therefore, if you have any questions, please ask them in the course forum in the thread
'Assignment 5: Questions'.



Task: The N-Gram Stemmer

"Stemmers" are tools used to determine the similarity of two words. One usage of them is in the field of media retrieval, for example by search engines, to find out whether a keyword from a resource is similar enough to the keyword entered by the user and thus if the resource is relevant for the user.
"N-gram stemmers" do this based on a simple algorithm. They split a keyword into substrings of a certain length n, thus these substrings are called "n-grams".
Your task for this assignment is to program a simplified n-gram stemmer:

Create a class called **Stemmer**.
The class has the following public static methods:

- A method called **getNGrams** which receives a *keyword* and a *number*. The method should split the keyword into n-grams of the length of the number and return them in an array.
  - n-grams split the keyword into every possible substring of the given length. For example, the Bigrams (= n-grams with the length 2) of the keyword "cat" would look like this:

    `ca, at`
  - If the given keyword is shorter than the given number, use asterisks '*' to bring the String to the needed length. For example, if we take the same keyword "cat" and 'split' it into 5-grams, it would look like this:

    `cat**`
- A method called **getShared** which receives two *sets of n-grams* as String arrays. The method returns an array that contains the n-grams that are present in both sets.
  Note: You can assume that both sets will contain n-grams of the same length.
- A method called **getDistance** which receives two *sets of n-grams* as String arrays. The method should calculate and return the "distance" between the two sets.
  The "distance" is the amount of shared n-grams of two keywords in relation to the total n-grams. It is thus calculated by counting the amount of shared n-grams and dividing that number by the sum of total n-grams for both keywords.
  Note: You can assume that both sets will contain n-grams of the same length.
- A method called **isRelevant** that receives a *keyword* and a *text* to compare it with. The method returns **true** if:
  - The text contains a word that has a distance of at least 0.4 when compared with the keyword.

  Else the method returns **false**.
  Use n-grams with n = 3 to determine the distance.
  Note: You can assume that we will use simple texts that don't contain any punctuation to test this.

Example

```
houseNGrams = getNGrams("house", 3); // returns {"hou", "ous", "use"}

trousersNGrams = getNGrams("trousers", 3); // returns {"tro", "rou", "ous", "use", "ser", "ers"}

getShared(houseNGrams, trousersNGrams); // returns {"ous", "use"}

getDistance(houseNGrams, trousersNGrams); // returns 0.222...
/*
Explanation:
amount of n-grams in houseNGrams = 3
amount of n-grams in trousersNGrams = 6
total n-grams = 3 + 6 = 9
amount of shared n-grams = 2
so the distance is = 2/9 = 0.222...
*/

isRelevant("house", "trousers"); // returns false

isRelevant("house", "the house is clean"); // returns true
```