Software Engineering and Programming Basics - WS2021/22
Assignment 6

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Professorship of Software Engineering

12| 2021

Organisational

Deadline

20.12.2021 - 23:59

Submission

To submit your answers, please use the Task item titled 'Submission' in the menu of the Assignment 06. You can upload your /.java files here. There are sample files shown in the highlighted Samples section. To submit several classes, you can put all your files in a .zip folder.

Please remember that package name is the name of assignment, i.e. assignment6.

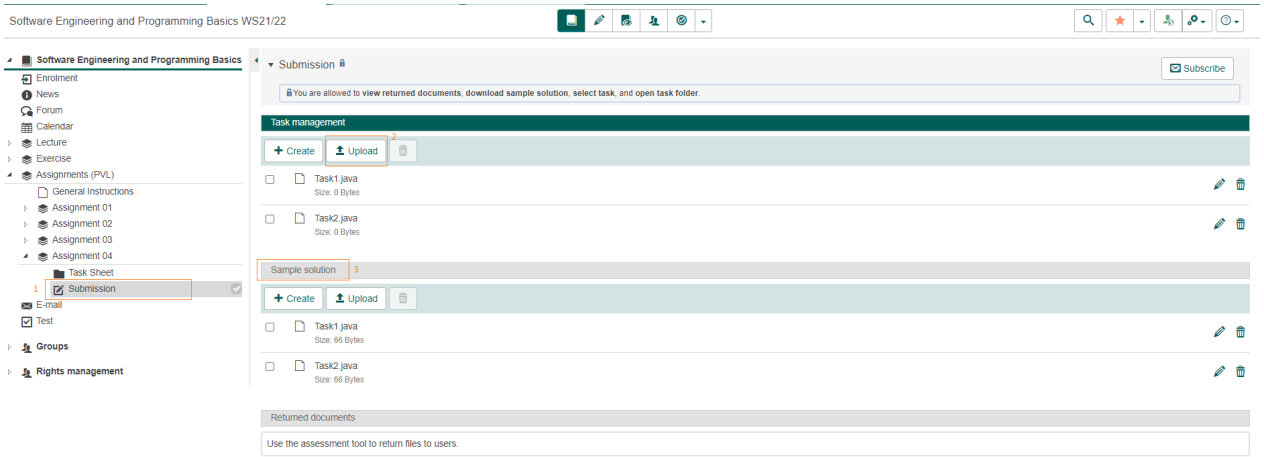
Make sure your files are correctly named: Package, class and method names should be exactly as mentioned on the Task Sheet in order to get grades.

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 6: Questions'.



Task: The Voting

For this assignment, your task is to program a class that can help us administer the data of a simple voting, where people could vote either yes or no on a question. Since it is typically not known how many people will attend a voting, we need a flexible data structure. Thus, we will be working with a simple linked list:

- Create an enum called **Vote**. It should contain the constants *YES*, *NO* and *INVALID*.
- Create a class called **Node**. It contains an attribute **data** of the type Vote and an attribute **next** of the type Node.
  - The class has one constructor which receives data of the type Vote.
  - The class has getter methods for its attributes, i.e. **getData** and **getNext**.
- Create a class called **VoteList**. It has an attribute **head** of the type Node.
  - The class has one constructor which initializes the head with null.
  - The class has a getter method **getHead** for the head attribute.
  - Note that the end of the list should be marked by the final Node's 'next' attribute pointing to null.

    The class also has the following public methods:
  - A method called **add** which receives a Vote as *data*. It creates a new Node containing the data and adds the Node to the list that starts with the calling object's head. It returns 'true' if a new Node has been added successfully and 'false' if there was an issue.
    Note: What kind of add method you use (whether it adds the new node at the front of the list, the back of the list or in a sorted position) is up to you.
  - A method called **importVotes** which receives an array of Votes as *data to import*. It adds all votes from the array to the calling VoteList. It should keep track of the number of votes that it adds to the list and return the number of added votes once it is finished.
  - A method called **isValid** that returns 'true' if the overall VoteList is considered valid and 'false' if it isn't. A VoteList is considered invalid if more than 40% of its Nodes contain the data *INVALID*.
  - A method called **countVotes** that returns the result of the overall VoteList:
    * If there are more Nodes that contain *YES* votes than *NO* votes, the method returns *YES*.
    * If there are more Nodes containing *NO* votes than *YES* votes, the method returns *NO*.
    * If the amount of *YES* and *NO* votes are exactly the same, or if the VoteList overall is considered invalid (see the isValid method), the method returns *INVALID*.

Example

```
christmasPartyVoting = new VoteList();

friendVotes = {YES, YES, NO, INVALID, INVALID, NO, YES, NO};
familyVotes = {INVALID, YES, YES, NO};

christmasPartyVoting.importVotes(friendVotes); // returns 8
christmasPartyVoting.isValid(); // returns true
christmasPartyVoting.countVotes(); // returns INVALID

christmasPartyVoting.importVotes(familyVotes); // returns 4
christmasPartyVoting.isValid(); // returns true
christmasPartyVoting.countVotes(); // returns YES

/*
Assuming we use and add method that adds new Nodes at the end of the list,
the final list could look like this:
[YES][YES][NO][INVALID][INVALID][NO][YES][NO][INVALID][YES][YES][NO]
*/
```