Software Engineering and Programming Basics - WS2021/22
Assignment 8

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Professorship of Software Engineering

1| 2022

Organisational

Deadline

24.01.2022 - 23:59

Submission

To submit your answers, please use the Task item titled 'Submission' in the menu of Assignment 8. Please upload all your classes as /.java files here. There are sample files shown in the highlighted Samples section.

The package name is the name of the assignment, i.e. assignment8. Do not forget to add it!

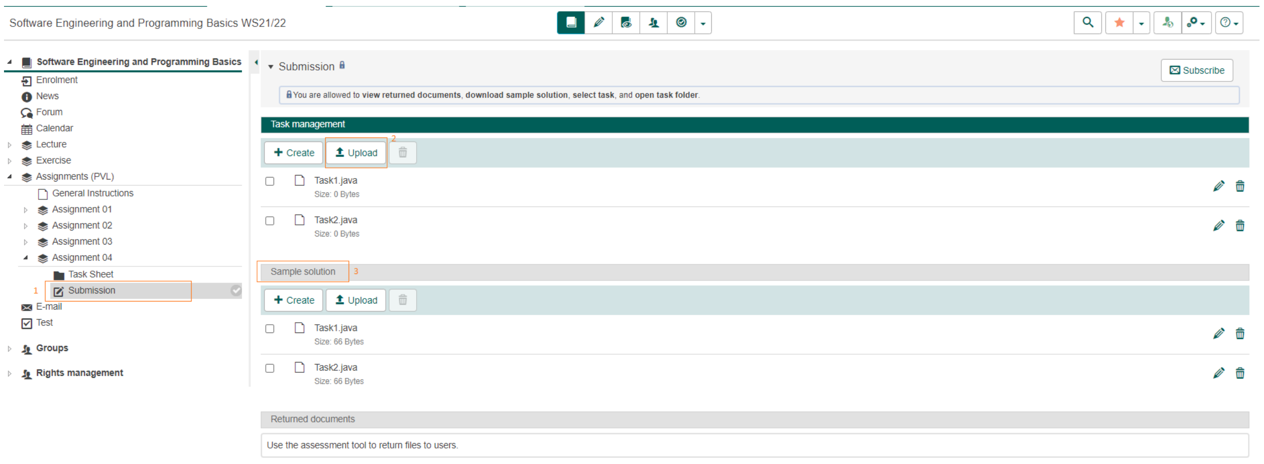
Make sure your files are correctly named: Package, class and method names should be exactly as mentioned on the Task Sheet in order to get grades.

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 Instructions** found in Opal.

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 8: Questions'.



You are allowed to **view returned documents**, **download sample solution**, **select task**, and **open task folder**.

Use the assessment tool to return files to users.

Task: Let's Play a Game

For this assignment, your task is to build the basis for a card game based on the standard 52-card deck (also known as French-suited playing cards). By now you know multiple data structures to organise several Objects of the same type and we purposely designed this assignment so you can work with the data structure of your choice. However, this means that you should pay extra attention in which format data is given to you and in which format you need to be able to return it. It also means that you might need extra methods and classes that might not be explicitly mentioned in this task sheet. Remember, adding *more* methods is always allowed.

- Create an enum called **Suit** and an enum called **Rank**. They should contain constants for the suits and ranks of a standard 52-card deck respectively.
- Create a class called **Card**. It contains the attributes **suit** and **rank** of the respective type. It also contains a constructor that receives a Suit and a Rank as argument, as well as the getter methods **getSuit** and **getRank**.
- Create a class called **Player**. A player has an attribute **hand** that contains cards. You can decide on your own whether you want to work with arrays, a linked list or a different data structure of your own choice and creation.
  - However, you need to provide a method **getHand** that returns the hand of the player as an *array* of Cards.

    The class also has the following public methods:
  - A method called **sortHand** which sorts the hand of the player according to the suits and ranks of the cards.
- Create a class called **Game**. A Game has an attribute **deck** that contains a complete deck of Cards. A Game also has an attribute **players**. You can again decide for yourself what data structure you want to use to represent multiple cards and players.
  - The class has a constructor which receives the players as an *array* of Player objects. A game can have between one and four players.
  - The constructor should also initialize the deck with all the necessary cards. Being a complete deck means that every possible card needs to be contained exactly once.

    The class also has the following public methods:
  - A method called **deal** which deals out the cards randomly amongst the players of the game. This means that the cards should be added to the players' hands. All cards should be distributed, no card should be distributed more than once. If there are only three players, the deuce of clubs is not dealt to anyone, so that each player still has the same amount of cards on their hands.
    Note: You are explicitly allowed to use the Math library and the Math.random() method for this.
  - A method called **reset** which resets the game by emptying all the player's hands and resetting the deck to contain each card again.
  - A method called **displayDeck** which prints each card that is currently contained in the Game's deck. Use a user friendly format for this.

Example

```
player1 = new Player();
player2 = new Player();
arr = {player1, player2};

gameOfBridge = new Game(arr);

gameOfBridge.deal();
player1.getHand(); // returns a hand containing 26 cards
player2.getHand(); // returns a hand that contains the other 26 cards

player2.sortHand();
player2.getHand(); // returns a hand with the same cards as before, but they are now sorted

gameOfBridge.reset();
player1.getHand(); // returns an empty hand
gameOfBridge.displayDeck(); // prints the deck that should contain every card
```

The Standard 52-card Deck

The following are the suits and ranks of a standard 52-card deck for your reference. For your sortHand() method, use the order in which the suits and ranks are displayed here. Meaning if you have the following cards:
Ace of Hearts, King of Clubs, Deuce of Clubs, Ten of Spades, Eight of Diamonds
the sorted Hand would look like this:
Deuce of Clubs, King of Clubs, Eight of Diamonds, Ace of Hearts, Ten of Spades

Suits

Clubs, Diamonds, Hearts, Spades

Ranks

Ace, Deuce, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King