INTRODUCTION TO PROGRAMMING
DM550, DM857, DS801 (Fall 2019)
Exam project: Part III — Deadline: 23h59 on Friday, December 13th, 2019

Overview

In this part, your task is to develop the classes in the middle layer of the card game: classes `Deck`, `Board` and `AutoPlayer`.

Class `Deck`

This class implements a collection that stores a set of cards and returns them, one by one, in random order. You need to consult the documentation of class `Random` and decide on the best way to implement this functionality.

This class should provide the following methods:

- a constructor that returns a new deck with forty cards.
- a method `Card next()` that returns the next card from this deck, also removing it;
- a method `boolean isEmpty()` returning `true` if this deck is empty.

Your class should also implement the `Iterable<Card>` interface. The iterator should return all the cards in the order in which they are returned by `next()`.

Class `Board`

Objects of this class represent a state of the board during game play, including all relevant information.

This class should provide the following methods:

- a constructor taking two `Player`s as input and constructing a new board set up in order to start playing a new game;
- a method `Card trumpCard()` returning the trump card;
- a method `Player next()` that returns a reference to the player starting the next round;
- a method `Player move(Card card1, Card card2)` that updates this board after the first player to play in this round plays `card1` and the other player plays `card2`, returning a reference to the player who won the round;
- a method `boolean gameOver()` that returns `true` if the game is finished.

Observe that updating the state of the board (in method `move`) also includes updating the hands and sets of collected cards of both players.

Note. According to the rules of the game, the trump card is the card in the *bottom* of the deck. For the point of view of the implementation, though, it is irrelevant where it is taken *from*, and you are allowed to obtain it by a call to method `next()` from class `Deck`. However, you should make sure that it *behaves* as though it were at the bottom of the deck, i.e. that it is eventually added to the hand of the player who loses the round where the last card from the deck is dealt.

You are of course also allowed to take the card from the bottom of the deck.

Class `AutoPlayer`

This class extends class `Player` with two new methods, returning the automatic player's move in the case where he is the first or the second to play.

This class should provide the following methods:

- a method `Card next()` returning the next move by this autoplayer, in case it is the first to play in a round;
- a method `Card next(Card card)` returning the next move by this autoplayer, in case it is the second to play in a round and the other player has played `card`.

Additionally, your implementation can override any methods from class `Player`, if necessary.

Try to make your autoplayer as clever as possible.

Expected results

You must hand in classes `Deck.java`, `Board.java` and `AutoPlayer.java` implementing the contracts described above. Your classes should be clients of `Card`, developed in the second part of the project, and providers for the top-level class `Play`. Compiled versions of these classes are provided with this project. Your classes should be compatible both with your own implementation and with the ones given.

You should also hand in a report describing the design of your classes – your choice of attributes, whether you provide any additional getters and setters, which universal methods you override, and any other design choices that you think are important to document. One of the members of the group must be clearly identified on the title page as the coordinator for this part of the project; the coordinator cannot be the same as in any of the first two parts (unless the group does not have enough elements). The source code for the developed classes should be included as appendix.

The report will be the basis for the evaluation.