Homework #4
RELEASE DATE: 12/14/2015
DUE DATE: 1/11/2016, 12:00 (NOON)

As directed below, you need to upload your submission files to the github repository under the exact guidelines of the TAs.

Any form of cheating, lying or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts. Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.

Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.

Only English is allowed for writing any part of your homework. We do not accept any other languages.

1 Description

The boss of the POOCasino is very satisfied with your performance. Now, there is a new game opening: a one-deck black-jack game in the real-world casino with seven actions for the players: bet, hit, stand, double-down, split, surrender, and insurance. Check `http://en.wikipedia.org/wiki/Blackjack` for some information about the game, which roughly goes (from the dealer's view) as follows:

(1) Ask every player to make a bet B_i (a positive integer).

(2) Assign a face-up card and a face-down card to each player and the dealer.

(3) If the dealer's face-up card is ACE, ask each player whether to buy an insurance of $\frac{1}{2}B_i$ or not.

(4) Check dealer's face-down card (hole card), and if the dealer does not get a Blackjack, ask each player whether to surrender or not.

(5) For each player who did not choose to surrender,

- Flip up (open) the face-down card.
- If the two cards happen to be of equal face value, decide whether to split. If splitting, the player goes with two separate hands and continues the game with the decisions below. Re-splitting is not allowed.
- Decide whether to double down.
- Decide whether to hit, until a standing decision or busted, of course.

(6) Faithfully execute the following dealer actions:

- If the total card value is ≤ 16 or is a soft-17, hit.
- Otherwise, stand.

(7) Compare the result of the dealer to the result of the player.

- If player i surrenders, $\frac{1}{2}B_i$ goes to the casino.
- If player i gets busted, B_i goes to the casino.
- If player i gets a Blackjack, the player gets $\frac{3}{2}B_i$ more chips unless the dealer also gets a Blackjack. In the latter case, it is a "push" and the player just get 0 more chips.
- If player i doesn't get a Blackjack, and if the dealer gets busted, each player gets B_i more chips
- If player i doesn't get a Blackjack, and if the dealer gets a Blackjack, the bet B_i goes to the casino. If the player bought an insurance, however, she/he gets B_i back from the insurance, making it even.

- Finally, if neither player i nor the dealer gets a Blackjack, and neither of them gets busted, the sum of face values on the dealer's and on player i's hands are compared. If the dealer gets more, the player loses and B_i goes to the casino. If the player gets more, the player wins B_i more chips. Otherwise it is a "push" and the player just get 0 more chips.

You should carefully check `http://en.wikipedia.org/wiki/Blackjack` for the definitions of **double down, split pairs, insurance, and surrender**. You are asked to use the following classes to implement the Blackjack game. Only the JDK 1.7 compiled class files and javadoc generated documents are provided.

- an abstract `foop.Player` class that to be extended to create your own player, including some inner exception classes
- a `foop.Card` class that represents one of the 52 cards in a standard deck of playing cards; a `foop.Hand` class that represents the cards on a player's hand

You can design your own strategy. You can do so by using the dealer's face-up card on the table, your own cards on hand, or other players' cards. You can also "keep count of" the cards that have been shown so far (see the movie twenty-one?).

2 Requirements

- Implement a new `POOCasino` class that allows the abstract `Player` to play the Blackjack game with the other utility classes provided. Your `POOCasino` should be able to allow 4 different players in the game with the command `java POOCasino nRound nChip Player1 Player2 Player3 Player4` when `Playeri` is a subclass of `Player`. Here `nRound` is the number of rounds of the game, and `nChip` is the amount of chips that each player has in the beginning.
- Implement your own player (like `PlayerB86506054`) that extends the abstract `Player` class.
- Find a classmate and takes her/his player to enter a duel in your casino with your player (yours as Player1 and Player3 and his/hers as Player2 and Player4), and let them play for a large number of rounds (with many chips). You need to make both players *bug free* in your casino—so you should start collaborating with your classmate much earlier than the deadline. *We allow you to openly collaborate this time as long as there is no direct copy of your code in any part.*
- Write a short report with at **most five** A4 pages that contains the following items:

  (1) your name and school ID

  (2) the player's strategy that you implemented

  (3) the design of all the classes related to the casino, and the reason that you chose this design

  (4) the result of the duel between you and your classmate (her/his name and school ID), and any experience that you two learned during the duel

  (5) any part that you implemented that is worth getting "bonus" points

- We understand that there are some parts of the spec that may not appear clear enough. Please feel free to ask for the designer's intention on the forum, but **you do not have to follow every intention**. You can design your own amendment of the spec. Then, you can use **one more** page of report to write down the amendment. Illustrate your amendment clearly and you'd get at most 40 bonus out of a total of 200 points.

  You should submit your report in **PDF** format.

3 Submission File

Please submit your code with github as directed in the homework submission guide. Your directory structure (under `hw4`) should be

- `src/*`, your source code

- `lib/*`, `PlayerXXX.class` that you get from your classmate, `Player.class` and other utility classes that you get from the class website
- `Makefile`, where the TAs can use `make` on CSIE R217 linux machines to compile your code, and then `make run` to test your program with your player and your classmate's player
- a PDF file `report.pdf`, which is your report file written in English

Please do not include any other files (e.g. class files) in the repository. Otherwise you may lose some points.