Homework #6

TA in charge: Te-Kang Jan

RELEASE DATE: 05/26/2009
DUE DATE: 06/09/2009, 14:20

As directed below, you need to upload your submission file to the designated place on the course website.

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.

Both English and Traditional Chinese are allowed for writing any part of your homework (if the compiler recognizes Traditional Chinese, of course). We do not accept any other languages.

1 Description

The boss of the POOCasino is very satisfied with your performance last time. Now, you are asked to do a more difficult task: Implement a one-deck 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 Casino'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.

- a `Card` class that represents one of the 52 cards in a standard deck of playing cards.
- an abstract `Player` class that to be extended to create your own player.

You can reuse your strategy in HW3, make it better, or create a new one. 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?). As in HW3, a well-explained strategy can allow you to win not only more chips in the game, but also more score points in real life.

2 Requirements

- Implement a new `POOCasino` class that allows the abstract `Player` to play the Blackjack game with the `Card` provided. Your `POOCasino` should be able to allow 6 different players in the game with the command `java POOCasino nRound nChip Player1 Player2 Player3 Player4 Player5 Player6` 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.
- Write a short report with at **most four** 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) any part that you implemented that is worth getting "bonus" points

  You should submit your report in **PDF** format. See `http://jsc.cc.ntu.edu.tw/ntucc/pcroom/manual/Word2Pdf.htm` for some possible instructions for converting from Word to PDF.

3 Special Notes

- Readability of your source code would be worth 10 points out of 100 this time. The source code would be read by all the three TAs, each giving points based on the following qualitative measure:

  10 very readable
  8 readable
  6 mostly readable, but with some unreadable parts
  4 mostly unreadable, but with some readable parts
  2 unreadable
  0 very unreadable

  Your score in this part would be the average number of points from the three TAs rounded to the nearest integer.

4 Submission File

Please upload a single ZIP encrypted file to CEIBA. The zip file should be like `b86506054.zip`, where the file name should be changed to your own school ID. The ZIP file should contain the following items:

- `*.java`, which represent any other classes that you implemented
- README (optional), which instructs the TA to compile your files
- **Your report file in PDF format**