Homework #3

TA in charge: Te-Kang Jan

RELEASE DATE: 03/31/2009
DUE DATE: 04/14/2009, 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 POOCasino is ready to open. You, as its core programmer, needs to design a Blackjack game in the casino. In addition to letting the regular users play, you also like to implement some automatic Blackjack player in the game. Here are the rules of the Blackjack game that you are asked to design. They are simplified from the commonly used rules in real-world casinos (without split, insurance, nor surrender). Check `http://en.wikipedia.org/wiki/Blackjack` for some information about the game.

- Each table consists of a dealer and several players.
- Each table comes with a shuffler that uses `nDeck` (say, 4) decks of cards. The shuffler would shuffle the cards and use about three quarters of them in the game. The cards are re-shuffled when needed.
- One round of the game can be roughly described as follows:
  - In the beginning of the game, every player i is asked to make a bet B_i that is no more than her/his number of chips in the pocket.
  - Then, every player gets two face-up cards, and the dealer gets a face-down card and a face-up card.
  - With the two cards on hand, each player is asked to decide whether she/he wants to "hit"—that is, get another face-up card. The goal, as you may have known, is to get the sum of face values high and close to 21. The player hitting process ends until the player says "stand" or until the player gets busted (i.e., go above 21). Note that the ACE card can be treated as either 1 or 11. Also, all 10, JACK, QUEEN, KING cards are of value 10—we call them the TEN cards.
  - Then it is the dealers turn to hit. The dealer needs to faithfully implement the following rule. If her/his sum of face values is less than 17 or is a soft-17 (i.e., a 17 with ACE treated as 11), she/he hits. Otherwise she/he stands. The dealer hitting process ends until the dealer stands or gets busted.
- The winning/losing decisions are summarized as follows:
  - If player i busted, the bet B_i would go to the casino.
  - If player i gets a Blackjack (i.e., a 21 with ACE and TEN), the player gets $2B_i$ 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.

  - Otherwise, if the dealer gets busted, each player gets B_i more chips; if the dealer gets a Blackjack, the bet B_i goes to the casino.
  - Finally, 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. In 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 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
- a "Shuffler" class that implements the shuffling of N decks as described above
- a "Table" class that distributes the cards to the dealer/players and controls the game
- a "Player" class that represents the player as well as her/his betting/hitting strategies
- a "Dealer" class that represents the dealer as well as her/his faithful hitting strategy
- a "POOCasino" class that simulates a game

Note that you need to design the strategy of the player by yourself. You can do so by querying 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?). 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

- Read the document of the **Card** class (source code NOT provided), and use it in your program.
- Fill in your own code to all the TODO parts of the source files provided by the TAs.
- Run your program with `java POOCasino nDecks nRounds`, where `nDecks` is the number of decks to use (say, 8) and `nRounds` is the number of rounds to play (say, 100).
- Write a short report with at **most** two A4 pages that contains the following items:
  (1) your name and school ID
  (2) the player's strategy that you implemented, and the amount of chips that each of the three players has after 1000 rounds
  (3) the output of your program from executing `java POOCasino 2 1`
  (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 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:

- `Shuffler.java`
- `Table.java`
- `Player.java`
- `Dealer.java`
- `POOCasino.java`
- **Your report file in PDF format**

The TAs would re-compile and test your code with `demo.sh` or `demo.bat` provided on the course website.