Given: Wednesday, March 13, 2013
Electronic Copy Due: Tuesday, March 26, 2013 by 11:59 p.m.

Objectives

- Class design. Objects working together , Constructors ,Arrays and ArrayLists
- To make multiple instances of a single class, and understand how they are distinct
- to use incremental problem solving strategies to break a large difficult problem (this assignment) into small, easy to solve problems (methods and classes)
- to utilise error checking throughout, to better handle bad user input

What you will need to hand in

- All Source Code
- An updated testing document to now include some test cases that were generated using the white box methodology.
- You must create some test classes/mains that implement a few automatic tests for your program
    - If you feel inclined you can look into and use JUnit (but it is not mandatory for this assignment)

Program Description

In this assignment you must use the deck and card classes you created in part one to help implement a mini casino program (or the provided solution from part one if you chose. This casino will allow a player to move back and forth between playing two different games. The program must have the following features.

1. When the program starts up you must check to see if a file called `SystemInfo.txt` exists in the current directory.
    a. If it does exist you must load the info from the file
        i. The file will have a player name on the first line and the amount of money the payer has left to bet on the second line.
        ii. If the file does not exist you must ask for the current players name and assign them a starting money amount of $100 that they can use to place bets when playing the games.
    b. When the program shuts down you must save the players information to the file `SystemInfo.txt`. The first line once again must be the payers name and the second line must be the amount of money they have left when exiting the program.
    c. After the startup you must present a menu that contains the following.
        i. Let's them choose to play game 1 (Described later)
        ii. Let's them choose to play game 2 (Described later)
        iii. Let's them choose to exit the program.

d. Before each game starts a new deck(s) of cards should be created and shuffled before use.
    i. During the execution of a single game (even when repeating a play of a game) the same deck of cards should be used. Only when all cards run out should a new deck be created and shuffled.
e. The program does not shut down until they select quit from the main menu.
f. When a game is selected to play the user will be able to keep playing the game until they chose to exit and return to the main menu.
g. In both games the player will be able to make bets (winning and losing money based upon the results). When moving between games the user's current amount of money on hand must be reflected.
    i. For example while playing game one they choose to stop playing and have $120 at the end. They can then take this amount and use it at the start of the next game they select from the main menu.
h. The user can never have a negative amount of money on hand.
    i. This also implies that they can never make a bet larger than the amount of money they have on hand.
    ii. If they have zero money left then they cannot play any more games and must be taken back to the main menu where they can then select to exit the program.

Game 1 Description: Punto Banco card game

The following description is for one play of the game. At the end of the game the user must be asked if they want to play again or return to the main menu.

Upon launching the game, the user will be asked which outcome they would like to bet on. The three outcomes are:

P: Player Wins – This bet pays out if the Player (Punto) wins.
B: Banker Wins – This bet pays out if the Banker (Banco) wins.
T: Tie Game – This bet pays out if there is a tie (Egalite).

The options above must be selectable with the given letters. As before, both upper and lower case characters should be acceptable. An invalid input should print an error message.

Once a bet is made, the game must be played, the winner (Player or Banker) determined, and then the bet should be paid out if the user picked the correct outcome.

Punto Banco Basics:

Punto Banco is a casino game in which gamblers bet on the outcome of a game. What this means is that unlike in most casino games, the gambler is not technically "in" the game. Instead, the Player (Punto) and Banker (Banco) play according to a fixed set of rules, and the gambler may bet on the Player (Punto), Banker (Banco), or a tie (Egalite).

The value of the Player and Banker hands in the game are calculated by summing the value of the cards, modulo 10. Thus a sum of 10 results in a hand value of 0, and a sum of 18 results in a hand value of 8. In the game, cards with a rank of 2 through 9 are worth the value of their rank, face cards (Jack, Queen and King) and 10s are worth 0, and Aces are worth 1.

The sequence of play in the game is as follows:

1. The Player gets a card face up.
2. The Bank gets a card face up.
3. The Player gets a second card face up.
4. The Bank gets a second card face up.
5. Both hands should be printed to the screen (clearly marking each hand)
6. If the Player or Banker has a total of 8 or 9, no more cards are dealt (skip to 8).
7. If the Player has a total of 0-5, the player gets a third card face up.
8. The bank may get a third card face up, depending on whether the Player received a third card, which card, if anything, the Player drew and the Banker's total:
   - If the Player did not draw a card, the Banker draws if he has 0-5, and stands if he has 6-7.
   - If the Player drew a 2 or 3, the Banker draws if he has 0-4, and stands if he has 5-7.
   - If the Player drew a 4 or 5, the Banker draws if he has 0-5, and stands if he has 6-7.
   - If the Player drew a 6 or 7, the Banker draws if he has 0-6, and stands if he has 7.
   - If the Player drew an 8, the Banker draws if he has 0-2, and stands if he has 3-7.
   - If the Player drew an Ace, 9, 10, or face card (Jack, Queen or King), the Banker draws if he has 0-3, and stands if he has 4-7.
9. At this point, the result of the game is determined: the Player wins if his total beats the Banker, the Banker wins if his total beats the Player, or the result is a tie (Egalite) if the totals are the same.

In the casino game, a successful bet on the player wins what was bet ($1 in gets $2 back), a successful bet on the banker wins 95% of what was bet ($1 in gets $1.95 back), and a successful bet on a tie wins 8 to 1 ($1 in gets $9 back).

Game 2 Description: High Low Same guess game

At the start of the game the user is asked for the bet they want to use for the entire game play. Two decks are created, one for the user and one for the banker. All cards are draw from the respective decks until they are empty. At this point new decks should be created. Game play goes as follows:

1. One card is delta from the players deck and displayed to the screen
2. The user is asked if the dealers card will be higher, lower or same rank then the current card they have
3. A card is dealt from the dealer's deck and compared to the user's card. Both cards and the result of the play are displayed to the user.
   a. If the dealers card is higher and the users guess was "Higher" they win ($1 in gets $2 back)
   b. If the dealers card is lower and the users was "Lower" they win ($1 in gets $2 back)
   c. If the cards match in just rank and the user guessed "Same" they win ($1 in gets $4 back)

   d. If the cards match in both rank and suit then they tie and no one wins or loses.
4. Based upon the results the users money and balance are updated
5. They then are asked if they want to play again or go back to the main menu.

Documentation

The program should be documented according to the standards used in COMP 1502. Each method and class (if classes are used) must have a javadoc comment describing what it does (as show in the class examples and the assignment solutions you have been provided with). In addition, inline documentation should be used as needed. Indentation and the use of constants will also be marked.

Submission

All of your source code and supporting documents should be submitted to the submit drive in a folder whose name includes your first and last name. The date and time stamp on your submission will determine if the assignment is on time.

Marking

Your work will be graded not only on program correctness, but also on quality of algorithm design, code style, readability, and quality of documentation. All programs submitted will be tested using a set of test data. Programs that do not compile will receive at **most** 40%.