CSE 398 - Multi-paradigm Software Development in Scala - Fall 2015
Programming Assignment
Program: **2**
Due: **Sept 18th at 11:00pm**
Project Name: Prog2

For this assignment you will write a program that simulates a simplified version of the card game Blackjack. You'll create classes for representing cards, decks, etc. The object is to test out different playing strategies and generate statistics on the effectiveness of each strategy.

Blackjack is a popular American casino game. It is a card game in which the aim of the player is to achieve a hand whose points total nearer to 21 than the dealer's hand, but without exceeding 21.

Blackjack is played with an international 52-card deck without jokers. Casinos normally use several decks mixed together (in a device called a shoe), both in order to speed up the game (with more cards in play you don't have to reshuffle as often) and to make card counting more difficult (card counting is a technique used to gain an advantage over the casino by keeping track of the proportions of different value cards remaining in play.)

The number of decks differs from casino to casino, but there can be anything from two to eight decks in the shoe. The re-shuffling frequency also differs from casino to casino, and of course depends on the number of decks used. Frequent re-shuffling is another way to diminish the value of card counting.

The aim of the game is to accumulate a higher point total than the dealer, but without going over 21. You compute your score by adding the values of your individual cards. The cards 2 through 10 have their face value, J, Q, and K are worth 10 points each, and the **Ace is worth either 1 or 11 points (player's choice).**

The rules that we will use are the ones defined on the following site:
https://www.blackjackinfo.com/blackjack-rules/
I'm particularly interested in testing the strategy defined at:
https://www.blackjackinfo.com/blackjack-basic-strategy-engine/

We will simulate multiple players playing a conventional casino game with standard rules. Here are some *guidelines* for the classes and objects involved in the simulation:

- **object Card**
  - I'm posting a version of this on Coursesite. You are free to alter it as you see fit. The cards in the simulation will be represented by integers.
- **class Shoe**
  - Parameters:
    - ndecks: Int (the number or decks in the shoe)
    - reshufflePercent: Double (the percentage of cards remaining in the shoe that triggers a reshuffle of the cards).
  - Responsibilities:
    - shuffle() - Shuffles the shoe. This will be called automatically when the remaining cards go below reshuffePercent, but can also be called by another object.
    - dealCard() - Deals one card.

- **class Hand**
  - Parameters:
    - cards: Array[Card] - initial list of cards in the hand.
  - Responsibilities:
    - addCard(Card) - adds a card to the hand.
    - value() - returns the current “value” of the hand.
    - hardSoft() - returns whether the hand is hard or soft.
- **Class Player**
  - Represents a generic Blackjack player.
  - Parameters:
    - strategy - a Strategy object for determining the player's actions and bets.
  - Responsibilities:
    - Doing everything a player needs to do in a Blackjack game.
- **class Dealer**
  - The Dealer class represents the play of the Blackjack dealer.
  - Parameters:
  - Responsibilities:
    - Dealing cards, collecting bets, etc.
- **class Strategy**
  - This it the abstract base class for representing a player (or perhaps dealer) strategy.
  - Parameters:
  - Responsibilities:
    - action() - The action the player takes next: Hit, Stand, Split a Pair, Double Down, Buy Insurance.
    - bet() - The amount the player bets.
- **class ThorpStrategy, DealerStrategy, etc.**
  - These extend Strategy and are concrete implementations of the different stragegies.
- **class Blackjack Simulation**
  - This class runs the whole simulation and prints out statistics at the end. For each player you need to track at least: the strategy used, the max, min and final value of the players cash.