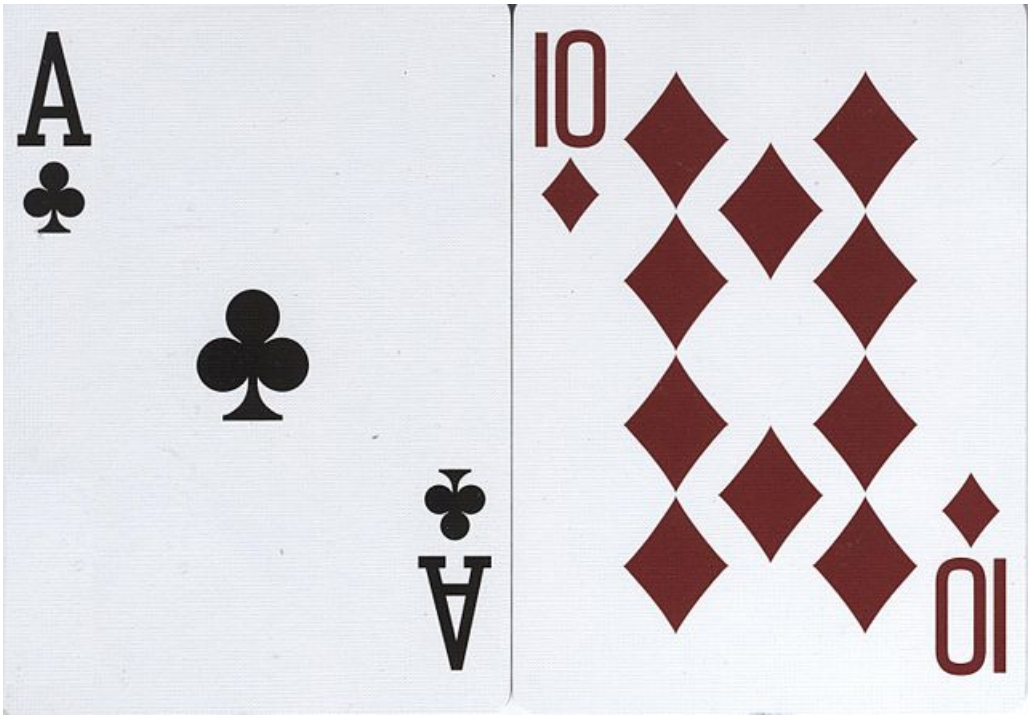
Blackjack

Intro:

Blackjack is a comparing card game between one or more players and a dealer, where each player in turn competes against the dealer. Players do not compete against each other. It is played with one or more decks of 52 cards, and is the most widely played casino banking game in the world.[1]



<Blackjack>

Now, in this project, you are required to implement **1 Human vs 1 Computer** blackjack.

Game Description:

In the game, the winner with the highest the number of points must have 21 points or less. **Blackjack** means get Ace and ten-value at same time in the **licensing phase**. Each player wants to win the game as much as possible, and the criteria are as follows:

- If the player is dealt an blackjack, and the dealer does not, the player wins, *vice versa*.
- If the player exceeds a sum of 21 ("**busts**"), the player loses, *vice versa*.
- If the player attains a final sum higher than the dealer and does not bust, the player wins, *vice versa*.
- If both dealer and player receive a blackjack or any other hands with the same sum called a "**push**", no one wins.

Main rules and flow path:

rules:

- There are 52 cards in each deck, except two jokers.
- Two decks of playing cards are used at a game. The deck will continue to be played in each subsequent round until the number of cards left is less than half or renewing the game. At this point, the cards are reshuffled.
- The value of cards two through ten is their pip value. Jack, Queen, and King are all worth ten. Aces can be worth one or eleven.
- In the game, the computer is the dealer, and the person is the player.
- No one can have more than five cards in his hand.
- The order of the draw is the sequential draw.
- All cards drawn in Hit are exposed.

flow path:

0. Shuffle: Decide whether to shuffle according to the rules.
1. Dealt: Player is dealt two cards facing up, and Dealer is dealt two cards, but one up (exposed) one down (hidden) in the beginning. If anyone has blackjack, skipping to 3.
2. Hit: Players are allowed to draw additional cards to improve their hands. If the player busts, the player loses, and skip to 4. Once all the player has completed his hands, it is the dealer's turn. The dealer can draw additional cards too, but if the sum of points in dealer hands is less than 16, the dealer must draw additional cards to make his sum points equal 17 or more. If the dealer busts, the dealer loses, and skip to 4.
3. Settlement: If no one busts, winners are selected according to the sum of points, and if same sum of points, push.
4. Reveal all the hidden cards and then the next game.

Project Requirement:

You are required to form groups of **two to four**(In principle, cross-shift group is **not allowed**. Note: Excluding International Students). Please design and implement a Java program to simulate the Blackjack game for **1 Human vs 1 Computer**. Note that the only programming language you can use in this project is Java. (Scala and Kotlin can also be accepted)

There are five tasks below to accomplish, each of which has several points towards the final mark of a group. All task **must** be completed, and additional contents (bonus) can be completed according your team situation. A framework/skeleton of the game will be released shortly to facilitate your programming.

Please kindly note that though you work as a team, every team members would be graded by the individual contribution to the project. Normally the member who does more contributions to the project will be graded higher than others.

With the same quality, the group with **fewer people** will **get more** scores for all tasks.

Task 1: Graphical User Interface (20 points)

- Your program should have a graphical user interface using Java Swing. (FX can also be accepted)

Task 2: Initialize the Blackjack Game (15 points)

- Your program should be able to initialize a new Blackjack game, which includes initialization of the graphical interface and initialization of the program's internal deck.
- Your program should be able to discard your current game and start a new one at any time.

Task 3: Play the Blackjack Game(35 points)

- Your program should let the player to play the game following the rules and the flow path described above.
- Your program should detect the winning status of a round, and go to next round when there is a winner. (Especially shuffle the deck in the right situation)
- Your program should allow machine players to make simple decisions about whether they should continue to draw cards.

Task 4: Game Master Command(15 points)

- Your program should have Game Master commands in terminal in order to facilitate the presentation in the defense.(For example, turning the two cards at the top of the deck into blackjack before starting the next game, turning the next card into Ace, etc.)

Task 5: Friendly Coding(5 points)

- Your program should have necessary comments, proper naming and proper format. If you need a tutorial, please read Alibaba-Java-Coding-Guidelines.

Bonus: (30 points)

If your program satisfies all the basic requirements above, you will get 90 points. The remaining 30 points will be given as bonus. You are highly encouraged to go beyond our requirements. Below are some possible ways to get bonus. Compared to the bonus points, the basic points are easier to get. **Here you need to rely on your own ability to present your programming charm!**

- The human player can be dealer. (5 points)
- When human player is the dealer, make machine player more intelligent. It can consider the probability of getting the cards that it want are left so that changing the drawcard strategy. (10 points)
- Design a platform for your game, such as adding multi-users (more than 2, that means you need to design a more complex GUI), ranking list(means human player can create new username), adding start menu for selecting the game modes, etc. (5 ~ 20 points)
- Design a credits system for your program. For example, every player have a point. Before each game, the player can set a base-point for this game (for example 5 scores). Then, every loser will lose 5 points and the winner get all lost points from the losers. (Full marks will only be given if a ranking list exists) (5 ~ 10 points)
- Make your game looks nice, such as changing the theme, adding sound effect, adding background music, adding more prompt label when playing the game. (5 ~ 20 points)

- Load and save a Blackjack game. The game can be saved at any stage of the game and reloaded. (5 ~ 10 points)
- Play animations of drawing cards. (5 ~ 20 points)

Scores for some requirements fluctuated within a range that means you'll only get the lowest score of this bonus requirement if you achieve very little function, and the more you complete this function, the higher your score will be.

The **total scores** of the group having two people got will be multiplied **1.1**, the group having three people got will be multiplied **1.0**, the group having four people got will be multiplied **0.9**.

Screenshots just for example:

[2]

Submission and Evaluation:

About submission:

Your group must submit the entire project files as compressed file package to Sakai(including anything you use in your program), and then Your group will also need to submit a report to explain how to compile and run your project. The **deadline** is **May 24th at 11 p.m**.

About evaluation:

In lab of week 15 every group is required to give a 10-minute presentation to introduce the functions realized, and all members are required to talk about what they had done and run the program to demonstrate .

Reference:

[1]https://en.wikipedia.org/wiki/Blackjack

[2]Screenshots from APP BJ21黑杰克(App Store)