Name:___________________________

OOP Assignment: Black Jack

Rules

Black jack is the most popular card game in the world. The game is also extremely simple. One or more players play against the dealer. Each player, and the dealer, draw cards one at a time. Each card is worth a certain number of points. The 2-10 cards are worth 2-10 points. All face cards are worth 10 points. The ace is worth either 1 point or 11 points. Anyone that gets more than 21 points is "bust" (loses). You win by having more points than the dealer, as long as you have 21 points or less. Multiple players can win at the same time. If the dealer goes bust, everyone wins.

Each player can choose when to stop drawing cards. Every card is placed face up for everyone to see. The exception is the dealer, who places the first card face up, and all subsequently drawn cards face down. The rules require the dealer to always draw another card until they have at least 17 points. Once the round is over, the dealer turns all cards face up for everyone to see.

Requirements

- Print an ordered deck of cards at the beginning of the game
- Randomly shuffle the deck then print the shuffled deck
- Allow a single player to face the dealer in black jack
- Show the black jack table after every draw (face up player cards and face up/down dealer cards)
- Display the points the player and dealer have. For the dealer the points shown are only the points from the one face-up card.
- When the round is over display dealer cards face up with total points
- At the end of the round display the remaining cards in the deck
- Ask the player after each round if they would like to play again
- Each subsequent round uses the same, partially depleted deck of cards
- The cards must not be displayed in the same order they will be drawn in (you can shuffle the deck after displaying it, or use whatever method you wish)
- Only refresh the deck to a new deck if there are fewer than 10 cards remaining
- Properly apply ace cards as 11 points, or 1 point if 11 causes a bust
- The solution must have at least 3 classes, and at least 2 static variables
- The solution must use inheritance
- Include all proper comments, naming styles, etc.
- Validate all user input

Name:___________________________

- Create a UML diagram of your work

Challenge Features

Completing all of the above requirements will earn you a maximum mark of 90%. To achieve up to 100%, you must complete <u>one</u> of the following challenge features:

- Allow multiple players (up to 4) to face the dealer simultaneously. If this is chosen, the deck of cards should be refreshed when the remaining cards are fewer than 5*number of players
- Give the dealer multiple behaviours. The dealer can play risky or play safe. Either randomly choose the dealer behaviour or let the player choose.
- Include sound in your game. If this is chosen, you must include at least 3 different sound effects which are played in certain situations. You must submit your work in such a way that the sound will still play even when submitted through the network
- Include a gambling feature in your game. Each round, player(s) receive their first card before placing bets. If the player wins, their bet is doubled. If the player loses, they lose their bet. The player can enter how much money they have when they start the game. This money persists through each round of the game until the application ends. Always display the amount of money the player has and how much is currently being bet. The game automatically ends when the player runs out of money

How to Submit

Submit your work the usual way. Your batch file must launch your client code (the code with main() method). Other classes will be run from your client code.

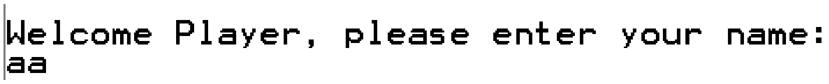
How to Display Suits

In Eclipse: https://www.youtube.com/watch?v=Ze6YAtb-B6w

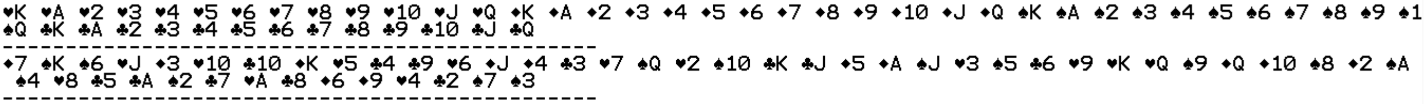
In Command Prompt: (char)3 should print ♥, (char)4 should print ♦, (char)5 should print ♣, (char)6 should print ♠.

Name:___________________________

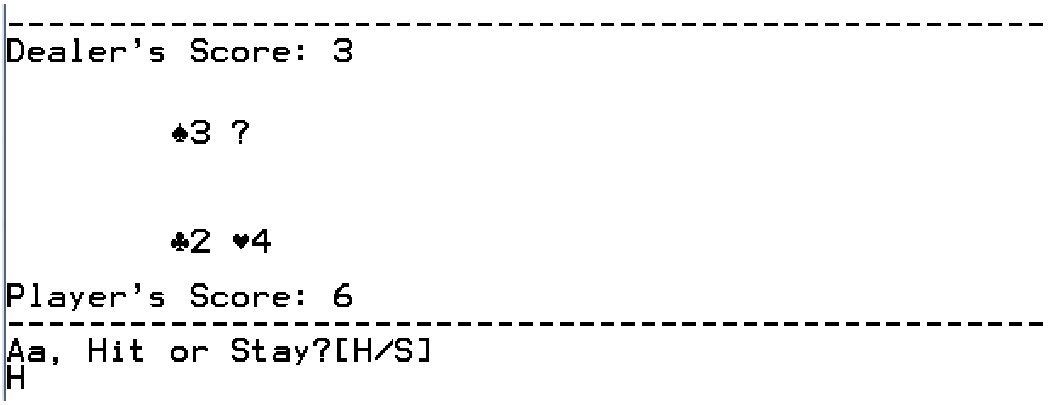
Example Output



```
Welcome Player, please enter your name:
aa
```



```
♥K ♥A ♥2 ♥3 ♥4 ♥5 ♥6 ♥7 ♥8 ♥9 ♥10 ♥J ♥Q ♦K ♦A ♦2 ♦3 ♦4 ♦5 ♦6 ♦7 ♦8 ♦9 ♦10 ♦J ♦Q ♠K ♠A ♠2 ♠3 ♠4 ♠5 ♠6 ♠7 ♠8 ♠9 ♠1
♠Q ♣K ♣A ♣2 ♣3 ♣4 ♣5 ♣6 ♣7 ♣8 ♣9 ♣10 ♣J ♣Q
-----------------------------------------------
♦7 ♠K ♠6 ♥J ♦3 ♥10 ♣10 ♦K ♥5 ♣4 ♣9 ♥6 ♦J ♦4 ♣3 ♥7 ♠Q ♥2 ♠10 ♣K ♣J ♦5 ♦A ♠J ♥3 ♠5 ♣6 ♥9 ♥K ♥Q ♠9 ♦Q ♦10 ♠8 ♦2 ♠A
 ♠4 ♥8 ♣5 ♣A ♠2 ♣7 ♥A ♣8 ♦6 ♦9 ♥4 ♣2 ♠7 ♠3
-----------------------------------------------
```

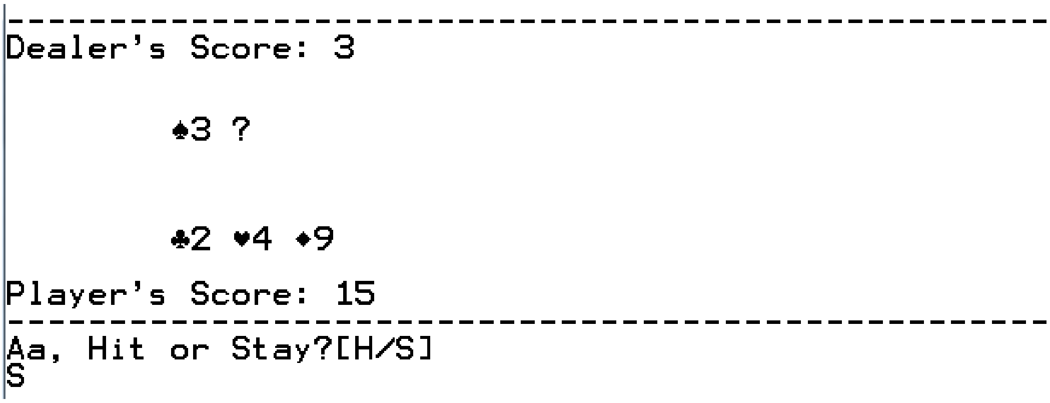


```
---------------------------------------------------
Dealer's Score: 3

        ♠3 ?


        ♣2 ♥4

Player's Score: 6
---------------------------------------------------
Aa, Hit or Stay?[H/S]
H
```

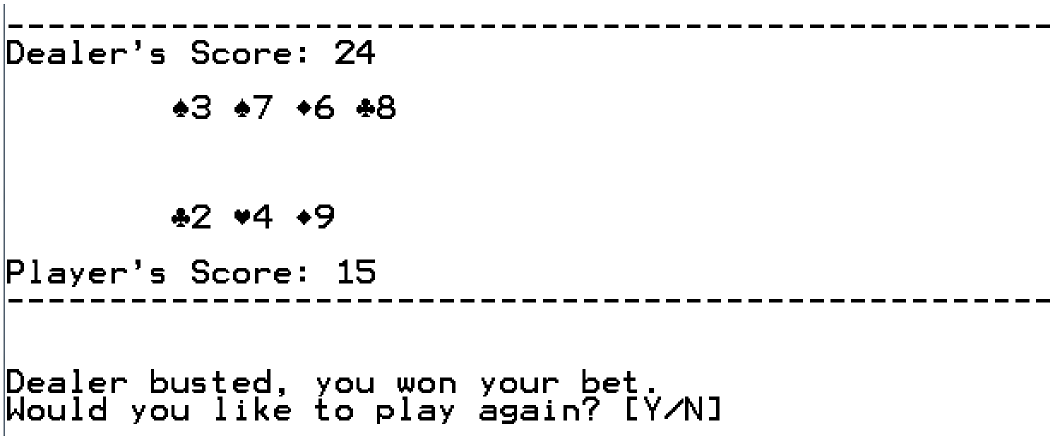


```
---------------------------------------------------
Dealer's Score: 3

        ♠3 ?


        ♣2 ♥4 ♦9

Player's Score: 15
---------------------------------------------------
Aa, Hit or Stay?[H/S]
S
```



```
---------------------------------------------------
Dealer's Score: 24
        ♠3 ♠7 ♦6 ♣8


        ♣2 ♥4 ♦9

Player's Score: 15
---------------------------------------------------


Dealer busted, you won your bet.
Would you like to play again? [Y/N]
```

Name:___________________________

Marking Rubric

	Level 4	Level 3	Level 2	Level 1
Code Format	The code is properly formatted. There are descriptive header and body comments, proper variable names, good use of white space, and generally good code structure.	The code is mostly properly formatted. However, the comments may not always be helpful or descriptive, or there are poor variable names, or slightly disorganized code structure.	The code is not formatted well. Variable names, comments, white space, and/or general code structure have not been organized well.	The code is very poorly formatted. No attention has been paid to formatting techniques such as comments or variable names.
Required Features	The program includes all required features. The features have been executed effectively and thoroughly.	The program includes most or all features. However, either execution is sloppy or one feature is missing.	The program only includes some required features. It is significantly limited in what it can do.	The program includes few if any functioning features. Substantial work is needed.
Additional Features	The program includes an additional feature. You have gone beyond what is expected of you!	The program does not include a completed additional feature, but does have some elements that go beyond requirements.	The program does not include any additional features. You should aim higher!	The program includes an attempt at additional features that causes major errors and ruins execution.
Creative Solutions	The implementation is effective, efficient, and unique. The writer of this code is working beyond what is expected of them!	The implementation is effective and efficient. The writer of this code is working at the level expected of them.	The implementation is effective. The writer of this code should think about how to make more efficient and direct solutions in the future.	The implementation is ineffective. The solution is incomplete and/or extensively error-prone.

Knowledge	Communication	Application	Thinking
/10	/10	/10	/10