# Class Diagram for the Online Blackjack Game

Learn how to create a class diagram for the Blackjack game using the bottom-up approach.

**We'll cover the following**

- Components of Blackjack
  - Card
  - Deck
  - Shoe
  - Hand
  - Players
  - Blackjack controller
  - Blackjack game view
  - Blackjack game
  - Enumerations and custom data types
- Relationship between the classes
  - Association
  - Aggregation
  - Composition
  - Inheritance
- Class diagram for the Blackjack game
- Design pattern
- AI-powered trainer

We'll create the class diagram for the Blackjack game. In the class diagram, we will first design the classes, and then identify the relationship between classes according to the requirements for the Blackjack game problem.

## Components of Blackjack

In this section, we'll define the classes for the Blackjack game. As mentioned earlier, we are following the bottom-up approach to designing a class diagram for the Blackjack game.

### Card

**Card** belongs to a suit and has a face value. The face value of the card is according to the card number. For example, if we have a number card 5, its face value is also 5. The face value for the King, Queen, and Jack is 10, and 1 or 11 for the Ace, depending on the situation.


The class diagram of the Card class


R3: Every card has points associated with it. The criteria to calculate the face value of the card is as follows:

| Card | Face value |
| --- | --- |
| Ace | 1 or 11 |
| From 2 to 10 | Equals the card number |
| Face cards (King, Queen, Jack) | 10 |

### Deck

**Deck** is a 52-cards of four suits, and contains nine number cards (2–10) and four face cards (King, Queen, Jack, and Ace). The **Deck** class contains a list of cards where the top card is in the first index.


The class diagram of the Card class


R2: The deck will consist of 52 cards in four suits, where each suit contains 13 cards: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, and King.

### Shoe

**Shoe** is a device to hold multiple **Deck**, and it has a shuffle operation.


The class diagram of the Shoe class


R1: The Blackjack game contains the shoe of cards which contains one or more decks of cards in it.

### Hand

The **Hand** class represents a Blackjack hand used in this game which can contain multiple cards.


The class diagram of the hand class


R9: The player can draw the additional card if their hand has less than 21 points.

R9: The dealer can draw an additional card if their hand is less than 17.

R10: If a player or the dealer's hand is more than 21, they bust and lose the game.

### Players

**Player** is an abstract class. There are two types of players, **BlackjackPlayer** and **Dealer**. These classes can be derived from the **Player** class.

**BlackjackPlayer**: They place the first wager and update the stake with winning and losing sums. They can choose between hit and stand options.

**Dealer**: They are primarily in charge of dealing cards and playing the Blackjack rules.


The class diagram of Player and its derived classes


R4: There can be two types of users that can play the Blackjack game, i.e., dealer and the player.

### Blackjack controller

The **BlackjackController** class validates the action(hit, stand) and responds accordingly.


The class diagram of the BlackjackController class

### Blackjack game view

The **BlackjackGameView** class represents the gaming view. The **BlackjackGame** class updates the **BlackjackGameView** class.


The class diagram of the BlackjackGameView class

### Blackjack game

The **BlackjackGame** class represents how we can play this game or its basic sequence of play. It controls the game, accepts wagers from players, and distributes cards from the **Shoe**. It also updates the game's status, gathers lost wagers, pays winning wagers, divides hands, and responds to player decisions to hit or stand.


The class diagram of BlackjackGame class


R5: The Blackjack game contains the shoe of cards which contains one or more decks of cards in it.

### Enumerations and custom data types

The following provides an overview of the enumerations and custom data types used in this problem:

- **Suit**: We need to create an enumeration to keep track of the suit of the card, whether it is diamond, spade, heart, or club.
- **GameStatus**: We need to create an enumeration to keep track of the account status, whether it is active, canceled, closed, blocked, or none.


Enums in the Blackjack game

- **Person**: Used to store information related to a person like a name, street address, country, etc.




The class diagram of the Person class

## Relationship between the classes

Now, we'll discuss the relationships between the classes we have defined above in our Blackjack game.

### Association

The class diagram has the following association relationships:

- The **Shoe** class has a one-way association with **Deck**.
- The **BlackjackGame** class has one-way association with **BlackjackGameView**.
- The **BlackjackGameView** class has a one-way association with **BlackjackGame**.
- The **Player** class has a one-way association with **BlackjackGame** and **BlackjackController**.


The association relationship between classes

### Aggregation

The class diagram has the following aggregation relationships.

- The **BlackjackGame** class contains the **Dealer** and **BlackjackPlayer**.
- Both the **Hand** and **Shoe** classes contain the **Card** class.




The aggregation relationship between classes

### Composition

The class diagram has the following composition relationships.

- The **BlackjackGame** class is composed of **Shoe**.
- The **Player** class is composed of **Hand**.




The composition relationship between classes

### Inheritance

The following classes show an inheritance relationship:

- Both the **BlackjackPlayer** and **Dealer** classes extend the Player class.

**Note:** We have already discussed the inheritance relationship between classes in the component section above one by one.

## Class diagram for the Blackjack game

Here's the complete class diagram for our Blackjack game:




The class diagram of the Blackjack game

## Design pattern

The iterator design pattern can be applied, since cards are assigned to the players from the deck of cards to just iterating through the list of cards.

We can also use the State design pattern for our online Blackjack game because this game has a finite number of states. Some of these states are as follows:

- Shuffle the deck
- Draw a card and give it to the dealer
- Draw a card and give it to the player
- Deal cards
- Player hit
- Player stand

## AI-powered trainer

At this stage, everything should be clear. If you encounter any confusion or ambiguity, feel free to utilize the interactive AI-enabled widget below to seek clarification. This tool is designed to assist you in strengthening your understanding of the concepts.