SYST17796 Deliverable 2 Group-4

# Table of Contents

1. Class Diagram ......................................................... 2

2. Design Document Template ..................................... 3-6

**Name:** Tanishkumar Vyas

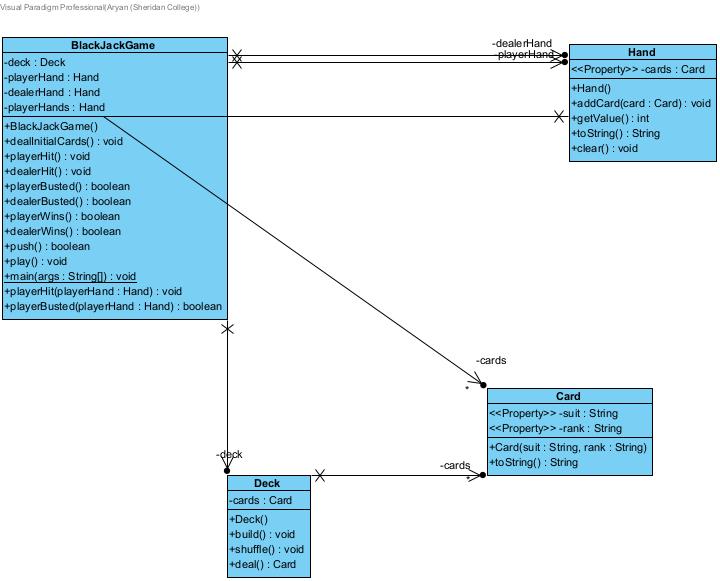
Aayush Soni

Dev Patel

Aaryan kachhiya

# 1. Class Diagram (Figure X)

The class diagram below shows the structure of our Blackjack game, showing the classes and their relationships:



# 2. Design Document Template

**Project Background and Description**

### The idea behind our project will be making the computerized form of the widely known card game Blackjack. In this game, the players play only against the dealer; the idea being to have a total of the cards’ values as close to or equal to 21 as possible. The turn of the game will come to an end when the player will either decide to stand or bust, that is to say when the hand value goes over 21.

### Design Considerations

#### Class Diagram (Figure X)

The class diagram for our Blackjack game includes the following classes and their relationships: The class diagram for our Blackjack game includes the following classes and their relationships:

• **Blackjack Game Class**: Oversees the basic functioning of the game, and relationship of the player to the dealer.

• **Deck Class**: The object is a deck of cards and has methods that construct, shuffle, and deal the cards.

• **Hand Class:** A set of cards that belong to either a player or a dealer; contains functions that assist in determining the worth of the given hand.

• **Card Class**: Symbolizes a person that has to play definite card.

**Associations and Multiplicities:**

• which points to the fact that The Blackjack Game class linked to the Deck class and that every game uses one deck.

• This object type is made up of multiple Card objects to create the cards within the deck.

• In the given Blackjack Game class, there is a one-to-many association with the Hand class, where the latter consists of a player’s or dealer’s hands.

• Each Hand class has several Card object that represents the cards in the players or dealers’ hand.

### Object-Oriented Design Principles

**Encapsulation:**

• Each class hides its data members and the implementation of the operations, while showing only the required operations. For instance, the Hand class is designed to have methods for addition of new cards, for getting the sum of the cards’ worth, but does not have the concrete implementation of these operations.

**Delegation:**

• The Blackjack Game class has set responsibilities like card dealing and determining win/loss drawn out to other classes such as Player, Dealer, and Hand classes, hence attaining modularity.

**Cohesion:**

• The cohesion used in the design of each class is therefore high; that is, the methods and properties of each class are closely related and are dedicated to a specific purpose. For instance, the Player class is strictly related to various actions that affect players only, including managing the hand of cards and decision making when it is the player’s turn.

**Coupling:**

• Coupling between classes stays at a low level so the classes are not coupled strongly, meaning that they are not dependent on many other classes. This implies that, when modifying one class, there are no effects on other classes, an aspect that boosts maintainability. For instance, changes made on the Player class have little impact on the Dealer or the Hand classes.

**Inheritance:**

• Although not commonly used, the use of inheritance has its advantages of reusability and extensibility. This could have been the case if there was a superclass, and Participant which would have been a superclass to both Player and Dealer classes as it contains attributes and methods common to both classes.

**Aggregation:**

• The Player class encapsulates subclasses of the Dealers and organizes the processes that characterize a game of Blackjack.

**Composition:**

### • Composition is used when one class manages the instance of the other. For instance, the Blackjack Game class may contain instances of the Player and Dealer classes; the implementations of create and destroy in this case would be owned by the Blackjack Game class.

### Flexibility and Maintainability

Our design focuses on ensuring the flexibility and maintainability of the system:

* **Flexibility:** The structural flexibility feature makes it possible to add new aspects or modify the current ones. For instance, modifications of the game rules or alteration of the scoring matrix is a simple task which does not require many changes in the core code.
* **Maintainability:** Thus, the low coupling and high cohesion levels cause, which largely implements OOD principles, means that it will be easy to maintain the code and understand it. Interconnected systems do not create confusion with changes in one part of the system affecting another part of the system and hence debugging and updates are relatively easier.

### Conclusion

### In this document we are going to describe the structure of the Blackjack game that we are going to implement, always taking into account modularity, maintainability and extensibility criteria. All through the process of implementing this game, we have strictly observed the principles of object-oriented design and therefore we have come up with a solution that is easy to understand and further extended making the overall implementation of the game of high quality.