Design Document for **BlackJack**

Code running instructions:

1. **main** function present in **src/Main.java** to run the application from the code
2. There is a jar available in **/out/artifcats/BlackJack_jar/BlackJack.jar** which can be run using the command
    a. java -jar BlackJack.jar
3. Or jar can be downloaded from here and run using the command above
    a. https://drive.google.com/file/d/1LBi7mU1uDu_9Fs0MvtyY-Co0fl8ifv4p/view?usp=sharing

Game instructions and rules used:

1. Winning and losing strategy same as BlackJack
2. It is a single player game
3. The player is assigned $1000 for the betting for the game and would be reset to $1000 if the player wants to play but has lost all the money
4. Rules used
    a. Hit
    b. Stand
    c. Double Down

Design Choices:

1. The **Card** class uses a **Suit** which has been assigned Unicodes for better readability on the console
2. Using the **Card** class as an array in the **Deck** along with a pointer to keep a track of the Latest Card like a **stack**
3. All the players including the dealer are **generalized** as a **RiskTaker** as it is a game where both the players would have similar structures
4. All the **RiskTakers** will have **Hands** which allows the code to be using **Single Responsibility Principle** as functionality related to the **Hand** is abstracted for the **RiskTaker**
5. Used **StringBuilder** at several places where the content to be printed was created by appending
6. Clearly indicated constants as **static** and **final** at the start of the **classes** so that we know the **constants** being leveraged in the class

Tools and Choice of language:

1. Java has been chosen as the language for the development for the application
    a. Strong support for Object-Oriented Design helped design the structure of the application
    b. Experience with the language, therefore comfortable during implementation
    c. Easier to execute on the console when created as a jar, therefore making demos always easier
    d. Console interaction is very simple and easy for the user to understand Implement re-usage of code very simply
    e. The modular approach helped me keep the code clean and readable
2. Extension of this application with Java's Multi-threading would help us create a multi-player version of the game