SYST 17796 Deliverable 1 Design Document Template (Kojo Ampofo)

1. **Project Background & Description**

* The final goal for this project is to design a game of Blackjack that will allow the user to generate cards and keep track of the card values the game will then run and establish a winner after the value of the cards are tallied.
* Blackjack is a popular card game played in casinos around the world. The objective of the game is to beat the dealer by having a hand that is worth more points than the dealer's hand, without going over 21. The rules are each player is dealt two cards and can choose to "hit" and receive additional cards to improve their hand, or "stand" and keep their current hand. Aces can be worth either 1 or 11 points, and face cards (kings, queens, and jacks) are worth 10 points. If a player's hand exceeds 21 points, they lose the game (known as "busting"). If the dealer's hand exceeds 21 points, they lose, and all remaining players win. If neither the player nor the dealer busts, then the player with the highest hand value less than or equal to 21 wins the game.
* This code is written in the Java programming language and consists of a single public class called "CardHandGenerator". It contains one static method called "generateHand" which returns an array of "Card" objects. The method generates a hand of cards of a specified size, It then obtains the number of possible values and suits for a card by accessing the length of the "Card.Value" and "Card.Suit" enums respectively, The code follows common Java coding conventions, such as using CamelCase for class and method names, and using indentation to indicate block structure. The code also uses inline comments to provide explanation for each step of the generation process.

1. **Project Scope**

* I. Kojo Ampofo will be creating this game from start to finish, my players will know that they are finished playing the game because I will have an indicator declaring whether each player bust or has the best hand.

1. **High-Level Requirements**

* **In my program I incorporated key aspects of Blackjack such as the ability for the game to communicate a win or loss**
* **And the ability to always know the score of the game**

1. **Implementation Plan**

<https://github.com/kampofo6/Deliverable-1.git>

**I used this repository to store and organize my project files even though I had no other partners to work with. All of the required documents are stored within my repository for ease of access.**

1. **Design considerations**
2. **Encapsulation:**

**Encapsulation is the practice of keeping an object's state private, and providing public methods for accessing or modifying that state. This helps to maintain the integrity of the object's data, and prevents external code from accidentally or maliciously modifying it.**

**The following are examples of encapsulation in the provided code:**

* **All of the methods in the GamePlayer class are private, which means they can only be accessed from within the class itself. This helps to prevent other code from accidentally modifying the state of the game or the cards.**
* **The Card class has private instance variables for its value and suit, and public getter methods (getValue() and getSuit()) for accessing them. This helps to keep the card's state private and maintain its integrity.**

**Potential for improvement:**

* **The generateHand(), addCardToHand(), and calculateScore() methods could be made instance methods of the GamePlayer class, which would make them more closely tied to the game and the cards and provide better encapsulation.**

1. **Delegation:**

* **Delegation is the practice of assigning responsibility for a task to another object or method. This can help to simplify code, improve organization, and make it easier to modify or extend functionality.**

**The following are examples of delegation in the provided code:**

* **The GamePlayer class delegates responsibility for generating a hand of cards to the generateHand() method, which in turn delegates responsibility for generating individual cards to the Card constructor.**
* **The GamePlayer class delegates responsibility for adding a card to a hand and calculating the score of a hand to the addCardToHand() and calculateScore() methods, respectively.**