SYST17796 Deliverable 1

Design Document Template

# **Overview**

## **1.** **Project Background and Description**

Describe the project goals and final vision. Include a brief description of how to play the game you have chosen and a reference to the rules of the game you have chosen. Also describe the current starting base code. Use technical terms to describe the code including what language it is written in, any patterns you can see and any coding conventions used.

* Project Goals and Final Vision
  + The goal of our project is to design and implement a basic Blackjack game in Java while utilizing object-oriented principles.
* Brief Description
  + Firstly, Blackjack is a card game that incorporates a dealer and a **single** player. The player is individually competing against the dealer where the goal is to try and have a hand value as close to 21 without exceeding it.
  + **Rules:** 
    - The dealer and player are dealt two cards each at the start
      * One of the dealer’s cards is face up (known value) the other card is face down (unknown value)
    - The player has the opportunity to choose “hit” (draw another card) or “stand” (keep their current total)
    - Face cards (Jack, Queen, King) are worth a value of 10 and Aces are worth 1 or 11
      * Ace in hand can reduce the value of an Ace from 11 to 1 if you were going to bust
    - The dealer must hit until they reach a total value of 17 or higher
    - If the player hits and their total value goes over 21 they bust (automatically lose the round)
* Starter Base Code Overview
  + Base code is written in Java and uses object-oriented concepts
    - Consist of abstract classes that we can then implement into future child classes with the extends keyword
  + Includes four classes:
    - Card
      * Abstract class that represents a playing card
      * Abstract method
        + toString()
    - GroupOfCards
      * Class that represents a collection of cards
    - Player
      * Abstract class that represents the player in the game
      * Abstract method
        + play()
    - Game
      * Abstract class that provides a template for game flow
      * Abstract methods
        + play() and declareWinner()
  + Coding conventions
    - CamelCase naming notation is used

## **2.** **Project Scope**

Describe the names and roles of each team member. Describe the technical scope of the project by talking about the interface and how you will know when the project is complete.

* Names and Roles (many tasks were shared overall)
  + Jimmy Dao – Game design, game rules, and class diagram
  + Grant Okawa – Project manager and overseas design document template
  + Nghi Lam Vo – Set up base code and refactors code for everything to be cohesive
* Technical Scope
  + Our game will run in the console as it provides simple and fast implementation for us to start testing our program
  + The project will be considered complete when:
    - The player can start the game
    - Cards are dealt to the dealer and player
    - Both player and dealer turns are completed
    - The program will correctly identify if each player has won, lost, or pushed (exceed the value of 21)
    - The total score of Wins/Losses will be printed out
    - The game will then end and ask if the player wants to play again or exit the program

## **3.** **High-Level Requirements**

|  |  |
| --- | --- |
|  | [Describe the high level requirements for the project. For example:] |

The new system must include the following:

· Ability for each player to register with the game

· Ability for the game to communicate a win or loss

· Ability for players to know their status (score) at all times

* High-Level Requirements
  + Ability for the player to register with the game and have a single dealer
  + Ability to keep track of player score
  + Ability to enforce the Blackjack rules we define
  + Ability for the player to hit or stand during their turn
  + Ability to automate dealers’ behavior
  + Ability to display outcome at the end of each round

## **4.** **Implementation Plan**

Include your Git repository URL here and a brief description of the expected use (i.e. each developer checks in code at the end of each day/week). Text files are stored under a separate directory, code, UML diagrams have their own folders etc.

Include information on coding standards you intend to follow and tools you expect to use (VP, NetBeans, eclipse, Junit…)

* Git Repository:
  + GitHub link to repository: <https://github.com/GrantOkawa/SYST17796-Project>
  + Created a public repository
  + Contributors:
    - Jimmy Dao
    - Grant Okawa
    - Nghi Lam Vo
* Brief Description of Expected Use
  + Each classmate will have their own branch to keep commit history organized
  + Keep commits concise and have clear commit comments
  + Open line of communication to discuss features being merged to the main branch
  + File structure (code, UML diagrams, files)
    - Project
      * src/cs.sheridancollege/project
        + All project code
      * UML diagrams
        + UML diagram
      * Files
        + Team contract
* Coding Standards
  + Use meaningful variable and method names
  + Comments for classes and methods
  + Single responsibility
  + Dependency inversion
  + DRY
    - Do Not Repeat yourself
* Tools
  + IDE: NetBeans
  + Diagram: Visual Paradigm for UML diagrams
  + Version Control: GitHub

## **5.** **Design Considerations**

Talk about how the current code is structured as it relates to the following OO principles. Each principle should have 2 or 3 specific examples from the base code or your intended additional code (i.e. potential for improvement).

* Encapsulation
  + GroupOfCards class has a private ArrayList of cards that keeps it hidden from external modifications
  + Player class will encapsulate their own hand of cards and score
  + Card class uses private variables of value and suit with public getters
* Delegation (assigning specific responsibility for a specific task or behavior to another class or object, rather than handling it all in one place)
  + The player will delegate their deck/card interaction to the GroupOfCards class
  + The Game class will delegate game behavior to the play() and declareWinner()
* Flexibility / Maintainability
  + Easy scalability, our game will be able to implement new features easily without the need to change a large amount of code
  + Following object-oriented principles will allow us to edit or update any necessary logic without major rewrites
  + Having the Game class abstract allows us to potentially create different card games that inherit and implement its logic