A close-up of a logo

Description automatically generatedA blue circle with a white letter

Description automatically generated

Software Design

DELIVERABLE 1

Student ID : 991727895

Student Name : Ronish Makaju

Class : 1249\_83045

Professor : Nagma Nagma

Submitted on : <dd-mm-yy>

# 

# 

Contents

[Team Contract 3](#_Toc179407000)

[Git hub repository: 4](#_Toc179407001)

[Class Diagram 4](#_Toc179407002)

[Design Documentation Template 5](#_Toc179407003)

[1. Project Background and Description 5](#_Toc179407004)

[How to Play Blackjack: 5](#_Toc179407005)

[The base code provided has the following classes: 5](#_Toc179407006)

[2. Project Scope 5](#_Toc179407007)

[Tasks: 6](#_Toc179407008)

[The project will be done when: 6](#_Toc179407009)

[3. High-Level Requirements 6](#_Toc179407010)

[4. Implementation Plan 6](#_Toc179407011)

[GitHub Repository URL: 6](#_Toc179407012)

[Tools we will use: 7](#_Toc179407013)

[5. Design Considerations 7](#_Toc179407014)

|  |  |
| --- | --- |
|  |  |

# Team Contract

**Please that if cheating is discovered in a group assignment each member will be charged with a cheating offense regardless of their involvement in the individual academic integrity history.**

**Please ensure that you understand the importance of academic honesty. Each member of the group is responsible to ensure the academic integrity of all of the submitted work, not just their own part. Placing your name on a submission indicates that you take responsibility for its content.**

|  |  |  |
| --- | --- | --- |
| Team Member Names (Please Print) | Signatures | Student ID |
| Project Leader: Ronish Makaju |  | 991727895 |

By signing this contract, we acknowledge having read the Sheridan Academic Integrity Policy

Git hub repository:  
<https://github.com/MakaRonish/project_black_jack>

Class Diagram  
A screenshot of a computer program

Description automatically generated

# Design Documentation Template

## 1. Project Background and Description

This project is about creating a digital version of the card game Blackjack. The game will be written in Java and will use Object-Oriented Programming (OOP). In Blackjack, players try to get their cards to add up to 21 without going over, while trying to beat the dealer.

## How to Play Blackjack:

Each player and the dealer are dealt two cards. Players can choose to take more cards ("Hit") or keep their current cards ("Stand"). The dealer must take cards until their total is at least 17. The goal is to have a hand that is closer to 21 than the dealer's hand without going over.

## The base code provided has the following classes:

Card: This is an abstract class that other card types can be based on (like playing cards for Blackjack).

Game: An abstract class for the overall game structure, with methods like play() and declareWinner().

GroupOfCards: This class manages a set of cards, like a deck of playing cards.

Player: An abstract class for players in the game, with a name and a play() method that can be customized for different games.

# 

## 2. Project Scope

Team Members:

Ronish Makaju

## Tasks:

Game Logic: Implement the rules of Blackjack, like dealing cards and calculating scores.

User Interface (UI): Design and implement the interface players will use.

Testing: Test the game to make sure everything works and fix any problems.

## The project will be done when:

Players can play a full game of Blackjack, including actions like hitting and standing. The game can show the winner and display player scores.

## 

## 3. High-Level Requirements

Players must be able to register and enter their name before starting.The game should shuffle the deck and deal cards to players and the dealer.The game must show the result of each round and let players see their current hand.The game should be able to decide the winner and announce it.

# 

## 4. Implementation Plan

GitHub Repository URL: https://github.com/MakaRonish/project\_black\_jack

The repository will have:

Source Code: The Java code for the game.

Documentation: Instructions and other files.

UML Diagrams: Diagrams showing how the code is structured.

## Tools we will use:

NetBeans: For writing and running the code.

Visual Paradigm (VP): For creating UML diagrams.

## 

## 5. Design Considerations

The code follows basic OOP principles like:

Encapsulation:

The Player class keeps player data private, so only the necessary parts of the program can access it.

We will improve this by making the GroupOfCards class more secure, so only certain methods can handle the cards.

Delegation:

The GroupOfCards class is responsible for managing the cards (shuffling, etc.).

We will create a new class to handle the dealer’s actions to make the code cleaner.

Flexibility/Maintainability:

The code is built in a way that makes it easy to add new features or games.

We will make the code even easier to maintain by creating a separate file for important constants (like the value 21) to make updates simpler in the future.