# Assignment #2 – Multi-Player BlackJack

Software Requirements Specification

Ka Yee Ching

Daniel Jensen

Chun Yin Lee

Aastha Sharma

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Description** | **Author** |
| 06/15/2022 | 1.0.0 | Initial Version | Daniel Jensen |
| 6/20/2022 | 1.0.1 | Added Sections 1 & 2 & 4 | Daniel Jensen |
| 6/21/2022 | 1.0.2 | Added Sections 2.1, 3.\*, 4.1, 4.3. Updated 2.5. | Daniel Jensen |
| 6/22/2022 | 1.0.3 | Added reference | Chun Yin Lee |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Purpose [4](#__RefHeading___Toc19440719)

1.1. Scope [4](#__RefHeading___Toc19440720)

1.2. Definitions, Acronyms, Abbreviations [4](#__RefHeading___Toc19440721)

1.3. References [4](#__RefHeading___Toc19440722)

1.4. Overview [4](#__RefHeading___Toc19440723)

2. Overall Description [5](#__RefHeading___Toc19440724)

2.1. Product Perspective [5](#__RefHeading___Toc19440725)

2.2. Product Architecture [5](#__RefHeading___Toc19440726)

2.3. Product Functionality/Features [5](#__RefHeading___Toc19440727)

2.4. Constraints [5](#__RefHeading___Toc19440728)

2.5. Assumptions and Dependencies [5](#__RefHeading___Toc19440729)

3. Specific Requirements [6](#__RefHeading___Toc19440730)

3.1. Functional Requirements [6](#__RefHeading___Toc19440731)

3.2. External Interface Requirements [6](#__RefHeading___Toc19440736)

3.3. Internal Interface Requirements [7](#__RefHeading___Toc19440737)

4. Non-Functional Requirements [8](#__RefHeading___Toc19440738)

4.1. Security and Privacy Requirements [8](#__RefHeading___Toc19440739)

4.2. Environmental Requirements [8](#__RefHeading___Toc19440740)

4.3. Performance Requirements [8](#__RefHeading___Toc19440741)

# Purpose

This document outlines the requirements for the multi-player blackjack game.

## Scope

This document will catalog the user and system requirements for the multi-player blackjack game. It will not, however, document how these requirements will be implemented.

## Definitions, Acronyms, Abbreviations

* + 1. Blackjack – A winning hand; where someone is dealt 2 cards, an ace (card value of 11) and a card with a value of 10 to hit a winning score of 21. The game gets its namesake from this winning hand.
    2. House – A term referring to the company running the game. The casino in most cases.

## References

Use Case ID: 1

Use Case Name: generateDeck

Primary Actor: Deck

Pre-conditions: Deck is not exist

Post-conditions: New deck is created successfully

Basic Flow or Main Scenario: When a new game starts, it need a new deck to play Blackjack

Exceptions: Deck is exist, need to shuffle the deck instead

Related Use Cases: 2

Use Case ID: 2

Use Case Name: shuffleDeck

Primary Actor: Deck

Pre-conditions: Deck is being used

Post-conditions: Deck is shuffled successfully

Basic Flow or Main Scenario: When a second game starts, it need to shuffle all the cards back to play Blackjack

Exceptions: Deck is not exist, need to generate the deck instead

Related Use Cases: 1

Use Case ID: 3

Use Case Name: Hit/addCard

Primary Actor: Card, Playerhand, DealerHand

Pre-conditions: after the player bets or when the player need additional card

Post-conditions: A new card is added to player hand

Basic Flow or Main Scenario: After the player bets, it will use addCard twice on the player hand, they can call for addCard again until it bust, or get closer or hit 21

Exceptions: N/A

Related Use Cases: 3, 5

Use Case ID: 4

Use Case Name: Stand

Primary Actor: Card, Playerhand, DealerHand

Pre-conditions: When the player decide not to add more card into hand

Post-conditions: New card does not add to player hand

Basic Flow or Main Scenario: After two cards originally dealt to the player, they think there cards on hand is close enough to 21, and decide not to add additional card, the call stand

Related Use Cases: 3

Use Case ID: 5

Use Case Name: checkHand

Primary Actor: Card, Playerhand, DealerHand

Pre-conditions: After two original cards are sent to the player or a new card added to hand

Post-conditions: The cards are checked what is the total and if it is blackjack or busted

Basic Flow or Main Scenario: After two cards originally dealt to the player and there has a new card adding to hand, it need to check what is the total on hand and if it is a blackjack or busted

Related Use Cases: 3, 4

Use Case ID: 6

Use Case Name: Bet

Primary Actor: Player, Dealer

Pre-conditions: Before starting the game

Post-conditions: The bet is confirmed and can not take back

Basic Flow or Main Scenario: Before you start the game, you need to bet, if you win, the dealer will pay you based on your bet; if you lose, the dealer will take you money

Related Use Cases: N/A

Use Case ID: 7

Use Case Name: Compare

Primary Actor: Playerhand, Dealerhand

Pre-conditions: When there is no more action from the player and dealer

Post-conditions: It will decide who win the bet

Basic Flow or Main Scenario: After the dealer stand, it need to compare the dealer hand and player hand to see who has the bigger number to win

Related Use Cases: N/A

Use Case ID: 7

Use Case Name: Save

Primary Actor: Player

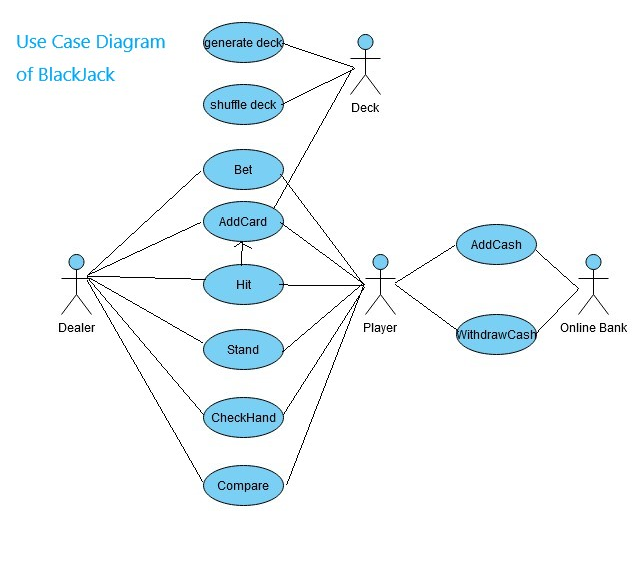
Pre-conditions: Every step of player did in the game

Post-conditions: Everything will save in the txt file

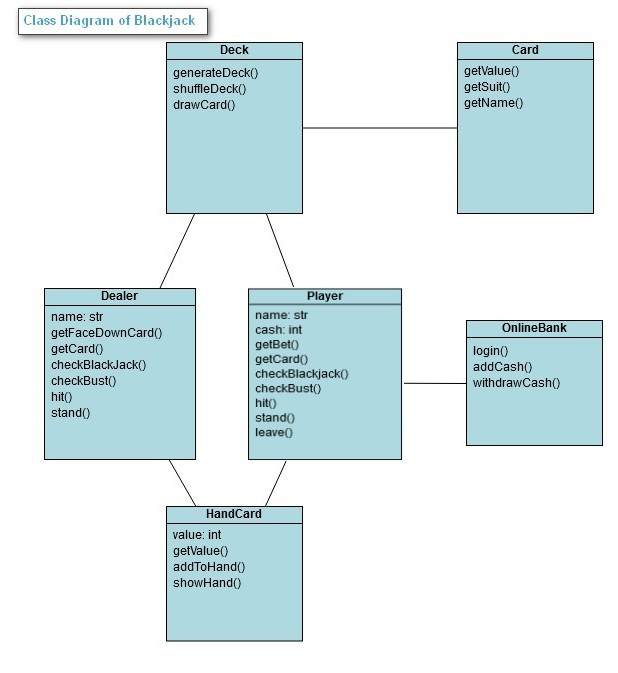
Basic Flow or Main Scenario: Starting from the game start, it will save player id, current balance, amount bet, status in game (in or out), action (hit or stay), cards in hand into a test file

Related Use Cases: N/A

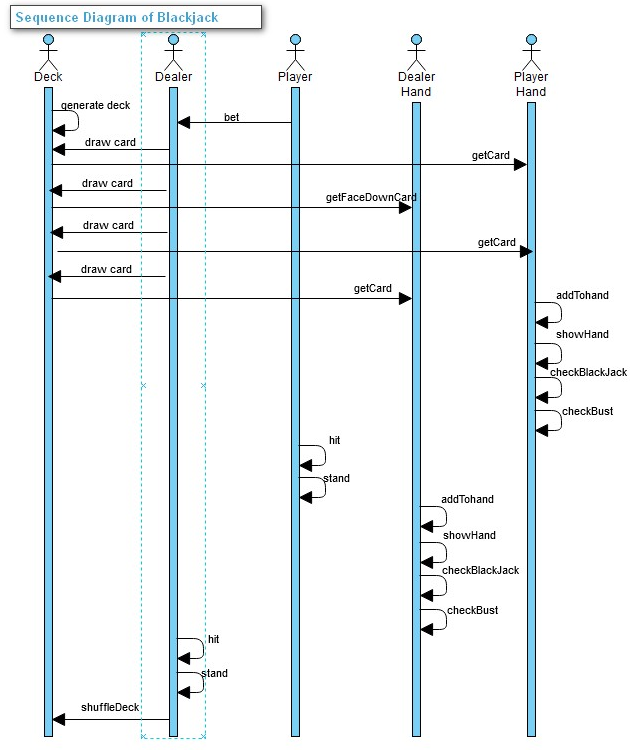
UML Use Case Diagrams Document



Class Diagrams



Sequence Diagrams



## Overview

The multi-player blackjack game will be designed to allow many users to play blackjack together on a game server. Users will be able to login through a client application to connect to the server.

# Overall Description

## Product Perspective

The multi-player blackjack game is a game designed for anyone that wants to play blackjack with other real players online. The game supports multiple people playing together, or one person playing against the house. This is a java based project, in order to allow the widest compatibility with user computers.

## Product Architecture

The system will be organized into 2 major modules: the Server module and the Client module.

Note: System architecture should follow standard OO design practices.

## Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

## Constraints

* + 1. Since each game of blackjack will be limited to 5 players, multiple games will run at the same time.
    2. Since we cannot add new users to a game in progress, new users will be added to new games.
    3. Since players can leave (or disconnect) anytime they want, for any hand that they have bet on but not completed, that money will be forfeited to the house.

## Assumptions and Dependencies

* + 1. It is assumed that each game of blackjack will only be using one deck and said deck will get shuffled between each game.
    2. It is assumed that the game will be played according to the basic rules of blackjack. Extras like blackjack insurance will not be necessary.
    3. It is assumed that users will only be able to be logged into one client at a time.
    4. It is assumed that users will only be playing in one game at a time.
    5. It is assumed users will need to leave a game to add funds to their accounts.

# Specific Requirements

## Functional Requirements

### Common Requirements:

* + - * 1. Users should have login credentials to connect to server from client.
      1. Users should be allowed to play together in groups of max size 5.
      2. Users should be able to add and withdraw funds from their account.
      3. Users should not be able to cheat.
      4. Users cannot have more than one account.

### Server Module Requirements:SR9

* + - 1. The server should create multiple games to handle additional players after a game is in progress or full.
      2. Users should not be allowed to join games already in progress. Player total in each game will shrink until all players leave.
      3. Employee (dealer) accounts will not track a balance.
      4. Only one dealer will be permitted per table. A dealer will only work at one table.

### Client Module Requirements:

* + - 1. Users should be allowed to play together in groups of max size 5. The server should create multiple games to handle additional players.
      2. Users can only be logged into one client at a time.
      3. Users can only play in one match at a time.
      4. Users should be able to log out whenever.

## External Interface Requirements

* + 1. SR9 SR1 The system must provide an interface to our payment system so that players can add funds and withdraw funds as needed. The interface is to be in a comma-separated text file containing the following fields: player id, current balance, amount of money being moved, action. Where “action” is whether the user has added or withdrawn the funds. The file will be exported nightly and will contain new transactions only.
    2. The system must provide an interface to our logging system so that all game moves and cards are logged to prevent cheating. The interface is to be in a comma-separated text file containing the following fields: player id, current balance, amount bet, status in game (in or out), action (hit or stay), cards in hand. The file will be exported after every game session (table) closes.

## Internal Interface Requirements

* + 1. SR24 The system must process a data-feed from the logging system so that all game moves and cards played are stored to prevent cheating. The data feed is to be in a comma-separated text file containing the following fields: player id, current balance, amount bet, status in game (in or out), action (hit or stay), cards in hand. The file will be exported after every game session (table) closes.

# Non-Functional Requirements

## Security and Privacy Requirements

* + 1. The sSR8 system must only allow individual users and the system administrators to view balance and login credentials.
    2. Users must create an account in order to use the system, with their account locked behind a username and password.
    3. Usernames and passwords will follow basic security protocols: usernames cannot be passwords as well, passwords must include both letters and numbers

## Environmental Requirements

* + 1. The sSR8 system will require that a Java Runtime Environment (JRE) compatible with Java 17 be installed on the user’s computer.
    2. The system must be developed using Java 17.

## Performance Requirements

* + 1. System must update game with low latency. The game should play at the same speed it would as if it were in person. The dealer will be controlled by employees (instead of AI) in order to facilitate said game speed.