syst 17796 Deliverable 1

design document

# Overview

## Project Background and Description

The goal of the project is to design a fully working blackjack card game written in Java. The game would start off with the user placing their bet and the program would generate two random cards for both the “dealer” and the “user”. The user will be given two options, “hit” or “stand”. If the user chooses “hit”, the program will give a randomly generated card to the user. If the user chooses “stand”, the program will not give a card to the user and move on to the next player which is the dealer. If the dealer or the user goes over 21, they will lose and the game is over. If the user or player gets a score of 21, they will win. If the user wins, the user receives twice their bet. If the dealer wins, the user will lose their bet.

The project base code consists of four classes: Card, Game, GroupofCards, and Player.

* class GroupOfCards is responsible for holding an ArrayList of cards.
* class Game is responsible for holding the player in an Arraylist
* class Card is the base class for the cards used in the game, in this case blackjack uses standard playing cards.
* class Player models each player in the game, it has a constructor that allows the players to set their unique ID.

The following additions will be made to the code:

* The program will start off by asking the user for money to bet using scanner
* program will randomly generate the cards using Math.random
* the cards will be stored in an arraylist
* program will use “if statements” to determine if the user or the dealer goes over 21

The game will be programmed in JDK 14 along with the Netbeans IDE and a Github repository. The code will adhere to Sheridan’s programming assignment standards and follow the main pillars of object oriented programming (abstraction, encapsulation, inheritance, and polymorphism).

## Repository

Link to the github repository: <https://github.com/GeoffreyFornoles/BlackJack.git>

## Project Scope

The project consists of two entities, the “dealer” and the “user” and their direct interaction with each other. Each entity is subject to different tasks and actions based on different outcomes. The program must first be able to follow the basic rules of blackjack detailed below in Section 3. Secondly it must be able to interact and communicate with the user both instructions and resulting outcomes. Finally for the program to be deemed complete it must pass all unit tests created.

The following group members are designated to these roles:

Martin Matov: Responsible for setting project deadlines

Usama Yasin: Responsible for keeping project on schedule

Geoffrey Gabriel Fornoles: Responsible for updating previous deliverables and maintaining code

## High-Level Requirements

The new system must include the following:

* Follow the followings rules of Blackjack from HitorStand.net (Excluding the “Doubling” and “Splitting” components of the game):
* The goal of blackjack is to beat the dealer's hand without going over 21.
* Face cards are worth 10. Aces are worth 1 or 11, whichever makes a better hand.
* Each player starts with two cards, one of the dealer's cards is hidden until the end.
* To 'Hit' is to ask for another card. To 'Stand' is to hold your total and end your turn.
* If you go over 21 you bust, and the dealer wins regardless of the dealer's hand.
* If you are dealt 21 from the start (Ace & 10), you got a blackjack.
* Blackjack usually means you win 1.5 the amount of your bet. Depends on the casino.
* Dealer will hit until his/her cards total 17 or higher.
* Doubling is like a hit, only the bet is doubled and you only get one more card.
* You can only double/split on the first move, or first move of a hand created by a split.
* You cannot play on two aces after they are split.
* You can double on a hand resulting from a split, tripling or quadrupling you bet

(HitorStand.net, 2011)

* 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
* Ability for the users to know the cards they have
* Ability for the users to choose between “stand” or “hit”
* Ability to communicate instructions to the user

## Implementation Plan

The Github repository will consist of all files related to the project including both code and design files. Members are to be notified when changes or additions have been made to the repository. They are also expected to check the repository every other day. Files are to be separated into three main sections: Code, UML diagrams, and design file.

The following are the tools and standards that will be used for this project:

* NetBeans: Default IDE for the project
* JDK 14: Default Java version for project
* JUnit: Default testing method
* Visual Paradigm: Default design tool for UML diagrams
* GitHub: Default Git repository
* Discord: Default method of communication for pair-programming and group communication method
* Google Drive: Default file editing software

## 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:
* Data fields are declared private
* Data fields are only accessible by way of constructors, accessor and mutator methods.
* This practice will continue throughout the code development to maintain security and minimize errors.
* Delegation:
* Class “GroupOfCards” will take the card properties in class “Card” through the process of delegation. Class “GroupOfCards” will store these card properties in ArrayList<Card> .
* The base code does not consist of any inheritance as of currently, all classes are delegated to serve a specific purpose
* Flexibility/Maintainability:
* base code consists of abstract classes to allow for class specific implementation of methods
* base code does not consist of interfaces to make for a more flexible and maintainable code.
* Abstraction:

Base code avoids the use of repetitive code and implementation of methods and data fields by the method of abstraction in following cases:

* Card class is set up to model a deck of standard playing cards
* Class Card, Game, and Player will be abstract classes, these classes are restricted. In order to access these classes, they must be inherited from another class.
* Each of the abstract classes will have abstract methods, these methods does not have a body. The body is provided by the subclass.

**References**

Hit or stand - blackjack rules and strategy. (2011). Retrieved February 28, 2021, from http://www.hitorstand.net/strategy.php