Project Title: Black Jack

Professor Name: Amandeep Sidhu

Date of Submission: 11-Febraury-2020

Group Members Name: Anshuk, Ekta Rao, Khushpreet Singh Brar, Vanshdeep Singh Sandhu (Leader)

Team Contract

SYST 17796 TEAM PROJECT

Team Name: JOKERS

Please negotiate, sign, scan and include as the first section in your Deliverable 1.

Please note that if cheating is discovered in a group assignment each member will be charged with a cheating offense regardless of their involvement in the offense. Each member will receive the appropriate sanction based on their individual academic honesty 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.

For further information read Academic Honesty Policy on AccessSheridan or visit the faculty office and speak with the Program Support Specialist.

Team Member Names (Please Print)	Signatures	Student ID
Project Leader: Vanshdeep Singh Sandhu	Vansh Sandhu	991569364
Khushpreet Singh Brar	Singh Brar	991559087
Anshuk Anshuk	Anshuk	991544464
Ekta Rao	Ekta	991575575

By signing this contract, we acknowledge having read the Sheridan Academic Honesty Policy as per the link below.

https://policy.sheridanc.on.ca/dotNet/documents/?docid=917&mode=view

Responsibilities of the Project Leader include:

- Assigning tasks to other team members, including self, in a fair and equitable manner.
- Ensuring work is completed with accuracy, completeness and timeliness.
- Planning for task completion to ensure timelines are met
- Any other duties as deemed necessary for project completion

What we will do if . . .

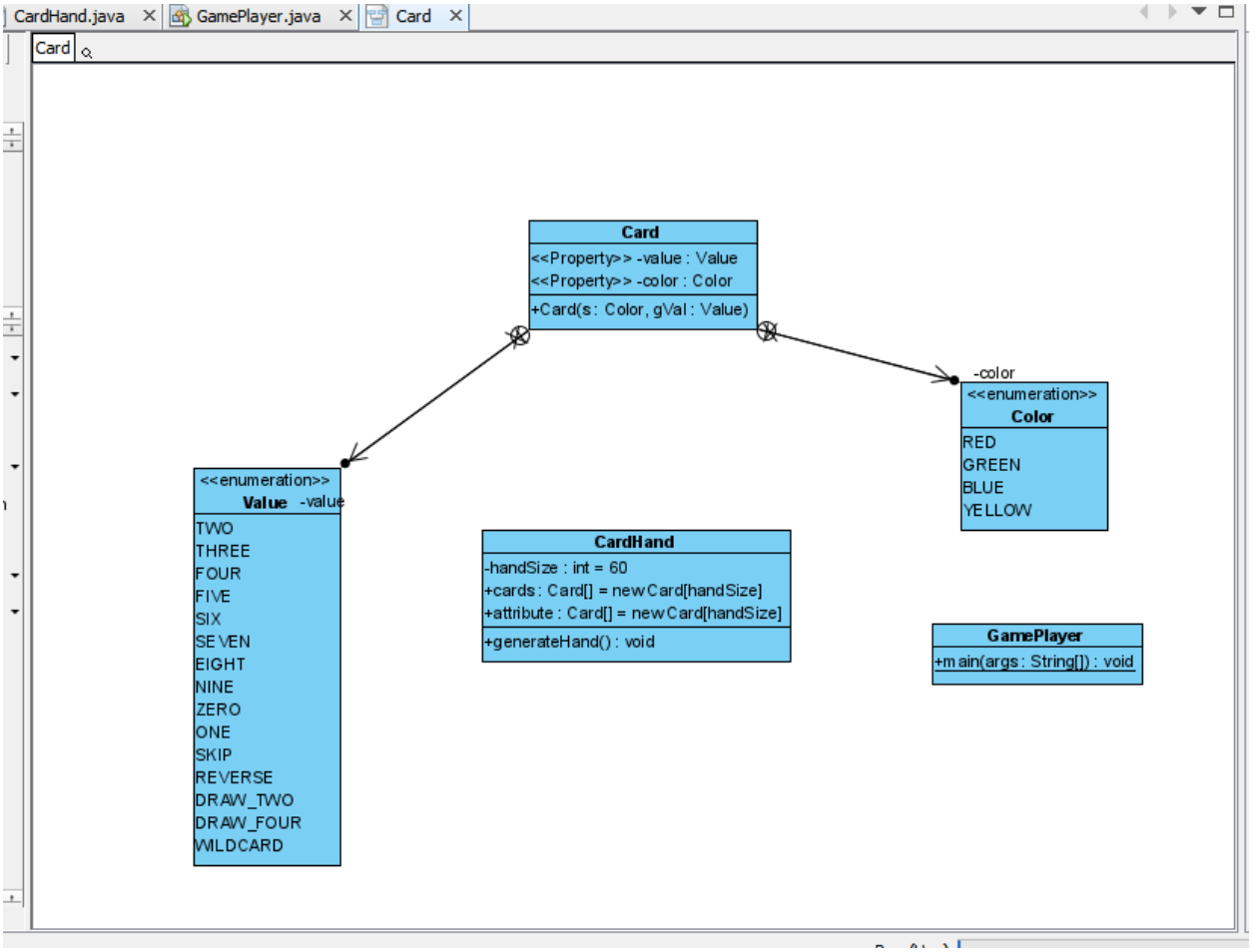
Scenario	Accepted Y/N + initial	We agree to do the following
Team member does not deliver component on time due to severe illness or extreme personal problem	Y E.K.A.V	a) Team absorbs workload temporarily __ b) Team seeks advice from professor __ c) Team shifts target date if possible ✓ d) Other:
Team member cannot deliver component on time due to lack of ability	Y E.K.A.V	a) Team reassigns component __ b) Team helps member ✓ c) Team member must ask professor for reference material ✓ d) Other:
Team member does not deliver component on time due to lack of effort	Y E.K.A.V	a) Team absorbs workload ✓ b) Team "fires" team member by not permitting his/her name on submission __

Scenario	Accepted Y/N + initial	We agree to do the following
Team member does not attend team meeting	Y E.K.A.V	a) Team proceeds without him/her and will assign work to the absent member ✓ b) Team doesn't proceed and records team member's absence __ c) Team proceeds for that meeting but "fires" member after __ occurrences __
A piece of production equipment fails such as a printer, disk drive, or laptop	Y E.K.A.V	a) Backup copies will be made and kept in the college ✓ b) A locker or "share" directory will be used for joint access __ c) A photocopy and duplicate disk of all deliverables will be made __ d) Other:
An unforeseen constraint occurs after the deliverable has been allocated and scheduled (a surprise test or assignment)	Y E.K.A.V	a) Team meets and reschedules deliverable ✓ b) Team will cope with constraint __ c) Other:
Team cannot achieve consensus leaving one member feeling "railroaded",	Y E.K.A.V	a) Team agrees to abide by majority vote __

"ignored", or "frustrated" with a decision which affects all parties		b) Team flips coin ✓ c) Other:
Team members do not share expectations for grade desired	Y E.K.A.V	a) Team will elect one person as "standards-bearer" who has the right to ask that work be redone __ b) Team votes on each submission's quality ✓ c) Team will ask for individual marking and will identify sections by author __ d) Other:

Scenario	Accepted Y/N + initial	We agree to do the following
Team member behaves in an unprofessional manner by being rude or uncooperative	Y E·K·A·V	a) Team attempts to resolve the issue by airing the problem at team meeting __ b) Team requests meeting with professor to problem-solve __ c) Team ignores behaviour __ d) Team agrees to avoid use of all vocabulary inappropriate to the business setting ✓
Team member assumes or requests that his/her name be signed to a submission but has not participated in production of the deliverable	Y E·K·A·V	a) Team agrees that this is cheating and is unethical ✓ b) Friends are friends and should help each other __ c) Team will submit with signature but will advise professor who will take action __
There is a dominant team member who is content to make all decisions on the team's behalf leaving some team members feeling like subordinates rather than equal members	Y E·K·A·V	a) Team will actively solicit consensus on all decisions which affect project direction by asking for each member's decision and vote __ b) Team will express subordination feelings and attempt to resolve issue ✓ c) Other:
Team has a member who refuses to participate in decision making but complains to others that s/he wasn't consulted	Y E·K·A·V	a) Team forces decision sharing by routinely voting on all issues __ b) Team routinely checks with each other about perceived roles ✓ c) Team discusses the matter at team meeting __

Class-Diagram



Deliverable 1 Design Document

1. Project Background and description

1.Creating a Blackjack card game.

The objective of the game is to have a set of cards whose total value beats the dealer's set of cards' total value. The cards are scored by their intended value except for the face cards and the ace. The face cards are all ten each, and the ace is either one or eleven depending on what the player wants. All the players are playing against the dealer.

Here are some terms you will need to know to play the game.

Hit - When the player wants another card from the dealer.

Stand - When the player doesn’t want another card from the dealer.

Bust - When the player or dealer goes over 21.

Push - When the dealer’s and a player’s hands are equal.

Split - When the player has two of the same cards. He has the option to play two hands.

2.How to Play:

1) Select a person to be the dealer
2) Have the dealer shuffle the deck

3) Now players have the option to bet depending on if they think they can get a higher total than the dealer without busting. If the player thinks they have a good chance of beating the dealer, they should bet more.
4) Have the dealer deal two cards to each person.
5) **Note**: Players' cards should both be laid face up, dealer should have the first one laid face down and the second face up
6) The dealer now gives each player starting to the left of him/herself the option of taking another card or 'hit'. They can keep taking more cards until they are satisfied and decide to stand or until they go over 21, in which case they bust.
7) Once every player has gone the dealer then hits until they reach at least 17 (see soft 17 in special rules) in which case he/she must stand

3. Starter code:

There are in total four classes that we have used. They are internally linked with each other. The four Classes name are Card, Player, GroupOfCards and Game.

4. Final vision:

Complete and Tested Card Game…

5. Others:

- Language used to design the card game is JAVA
- Coding Conventions: - Comments, Naming Conventions, Programming Principles and Practices, Indentation of Code etc.

2. Project Scope

Name	Role
1) Vanshdeep Singh Sandhu 2) Khushpreet Singh Brar 3) Ekta Rao 4) Anshuk Anshuk	1) Team Leader Game Class 2) Card Class 3) Player Class 4) GroupOfCards Class

This report describes all the high-level requirements for the project game. “Blackjack” is a single-player game. The player goes through a lot of steps which are already described in “How to Play” section. We are working mainly with fulfilling all the high-level requirements. If talking about coding part, game have currently four classes. Card.java, used as a base for card itself. Game.java, used for game. It starts a game and forms all the necessary things to run the game. Player.java which is mainly for player ID and last is GroupOfCard.java which is for size of deck and adding array of cards. As we move on to next deliverables, we will be adding more code and child classes to this project and we will be creating user interfaces. So in the end when all of the high requirements and low priority things are completed like ability of user to register with game, communication of win and loss, and ability of knowing scores, we will get know to that project is completed.

3.High Level Requirements

- Play the Game
- Ability to deal the Cards
- Shuffle the Cards

- 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)
- Resolving Ties between dealer and player

4. Implementation Plan

- GitHub Repository Link: https://github.com/Sandhvan/BlackJack
- Use of GitHub: -GitHub is a Git repository hosting service. It provides access control and you can do collaborative work on GitHub. You can either upload your own work or take a project that someone else wrote and modify under your account. If you make changes and you want to share them, you can do pull request. There are features of push, pull, fetch+ merge. All these features make GitHub powerful way of collaborating.
- Tools Used: NetBeans and Visual Paradigm
- Documentation Folder: Contain important Images or Screenshots and PDF for the project

5. Design Considerations

- Abstraction Classes

```
public abstract class Game
{
*/
public abstract class Player
{
public abstract class Card
{
```

- Encapsulation

```
 */
public abstract class Game
{
    private final String gameName;//the title of the game
    private ArrayList <Player> players;// the players of the game
```

```
public class GroupOfCards
{

    //The group of cards, stored in an ArrayList
    private ArrayList <Card> cards;
    private int size;//the size of the grouping

    public GroupOfCards(int givenSize)
    {
        size = givenSize;
```

```
public abstract class Player
{
    private String playerID; //the unique ID for this player

    /**
     * A constructor that allows you to set the player's unique ID
     * @param name the unique ID to assign to this player.
     */
    public Player(String name)
```

- Use of Comments and Java Doc make it easy for Maintainability

```
private String playerID; //the unique ID for this player

/**
 * A constructor that allows you to set the player's unique ID
 * @param name the unique ID to assign to this player.
 */
public Player(String name)
```

```java
/**
 * Ensure that the playerID is unique
 * @param givenID the playerID to set
 */
public void setPlayerID(String givenID)
{
    playerID = givenID;
}

/**
 * The method to be instantiated when you subclass the Player class
 * with your specific type of Player and filled in with logic to play your game.
 */
public abstract void play();
```

- Array List

```java
//The group of cards, stored in an ArrayList
private ArrayList <Card> cards;
private int size;//the size of the grouping

public GroupOfCards(int givenSize)
{
    size = givenSize;
}

/**
 * A method that will get the group of cards as an ArrayList
 * @return the group of cards.
 */
public ArrayList<Card> showCards()
{
    return cards;
}
```


```java
//The group of cards, stored in an ArrayList
private ArrayList <Card> cards;
private int size;//the size of the grouping

public GroupOfCards(int givenSize)
{
    size = givenSize;
}

/**
 * A method that will get the group of cards as an ArrayList
 * @return the group of cards.
 */
public ArrayList<Card> showCards()
{
    return cards;
}
```