

DELIVERABLE 1

For

Elizabeth Dancy

Instructor of Software design and development

Sheridan College

Brampton, Ontario

By

Kaitlin Saqui: 991723734

Arshvir Singh: 991709700

Mandeep Kaur: 991701614

Manpreet Kaur: 991680973

June 14, 2023

TABLE OF CONTENTS

Team Contract	3
UML Class Diagram.....	7
Design Document	8
Introduction	8
Main Section	8
Project Background and Description.....	8
Project Scope	10
High-Leve Requirements.....	14
Implementation Plan.....	15
Design Considerations	17
Conclusion.....	19
References	20

Team Contract

SYST 17796 TEAM PROJECT

Team Name: Group #5


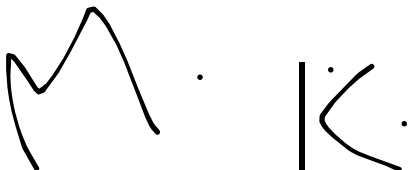
Please negotiate, sign, scan and include as the first page in your Deliverable 1.

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 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.

For further information, read Academic Integrity Policy here :

<https://caps.sheridancollege.ca/student-guide/academic-policies-and-procedures.aspx>

Team Member Names (Please Print)	Signatures	Student ID
Project Leader: Kaitlin Saqui		991723734
Arshvir Singh	Arshvir Singh	991709700
Mandeep Kaur		991701614
Manpreet Kaur	MK	991680973

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

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.
- £ Notifying the professor of any issues in a timely manner so that corrective measures can be taken.
- £ Any other duties as deemed necessary for project completion.

What we will do if . . .

Scenario	Accepted initials	We agree to do the following (Put an X corresponding to your choice in each box)
Team member does not regularly attend team meetings and/or does not respond to communications in a timely manner.	MK AS M. K.	Project leader emails the student citing the concerns and cc's the professor so they are aware of the situation at the very onset <u>X</u> (Mandatory) . a) ___ In addition to above, the leader/team will (add your own content here):
Team member does not deliver component on time due to severe illness or extreme personal problem.	MK AS M. K. AS	a) Team absorbs workload temporarily <u>X</u> b) Team seeks advice from professor ___ c) Team shifts target date if possible ___ d) ___ Other (specify):
Team member has difficulty delivering component on time due to lack of understanding or ability.	MK AS M. K. AS	a) Team reassigns component ___ b) Team helps member <u>X</u> c) Team member must ask professor for help ___ d) ___ Other (specify):

Scenario	Accepted initials	We agree to do the following (Put an X corresponding to your choice in each box)
Team member does not deliver component on time due to lack of effort.	MK AS M. K. J.L.	a) Team absorbs workload ____ b) Team member(s) ask professor to request a Participation Form from <u>all</u> team members. This <i>may</i> result in individualized grades being awarded for a deliverable ____ c) Both a. and b. above <u>X</u> d) ____ Other (specify):
Team cannot achieve consensus leaving one or more member(s) feeling that their voice(s) is/are not being heard in a decision which affects everyone.	MK AS M. K. J.L.	a) Team agrees to abide by majority vote <u>X</u> b) Team seeks advice from the professor ____ c) ____ Other (specify):
Team members do not share expectations for the quality of work on a particular deliverable.	MK AS M. K. J.L.	a) Team members will draw on each other's strengths to help bring the quality of the deliverable to a minimal acceptable level <u>X</u> b) Team votes on each submission's quality ____ c) Team member(s) ask professor to request a Participation Form from all team members, which may result in individualized grades being awarded for a deliverable ____ d) ____ Other (specify):
Team member behaves in an unprofessional manner, e.g. being rude, uncooperative and/or making one or more	MK	a) Team agrees to avoid use of all vocabulary inappropriate to a business/college setting <u>X</u>

Scenario	Accepted initials	We agree to do the following (Put an X corresponding to your choice in each box)
member(s) feel uncomfortable.	AS M. K. H.	b) Team attempts to resolve the issue by airing the problem at a team meeting <u>X</u> c) Team requests a meeting with the professor to discuss further ____ d) ____Other (specify):
There is a dominant team member who insists on making all decisions on the team's behalf leaving some team members feeling like subordinates rather than equal members	MK AS M. K. H.	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 <u>X</u> c) Team seeks advice from the professor ____ d) ____Other (specify):
Team has a member who refuses to participate in decision making but complains to others that s/he wasn't consulted	MK AS M. K. H.	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 <u>X</u>

UML CLASS DIAGRAM

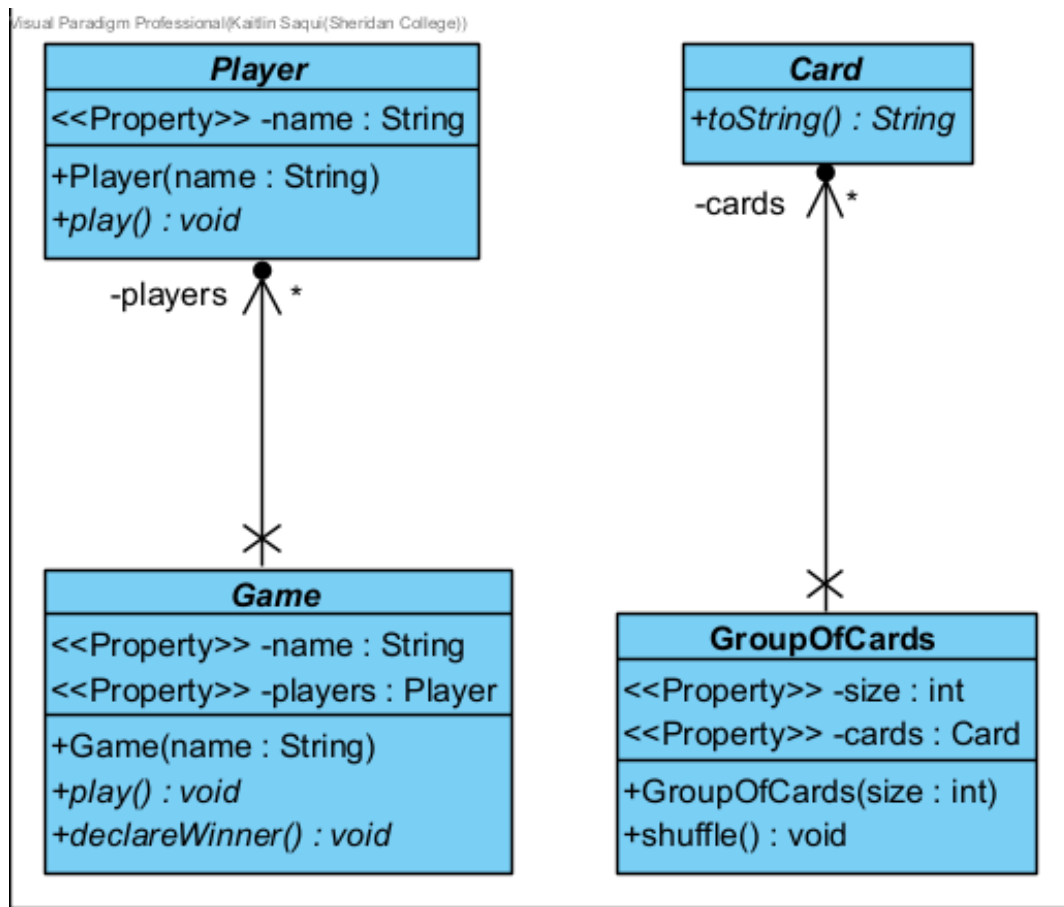


Figure 1 - UML class diagram for the Project Starter Code

This UML class diagram was designed in Visual Paradigm and is used to model the blueprint of the base code's system. It is a visual representation of the systems classes, data fields, methods, and their relationships to one another. In Figure 1 we can determine which classes are associated with each other. For example, there is a unidirectional association between the Game class and Player class, meaning that the Player class is unaware of their relationship with the Game class. The same relationship is represented between the GroupOfCards class and Card class. Throughout this project we will continue to expand this diagram to help illustrate the game's system.

DESIGN DOCUMENT

INTRODUCTION

This document is going to discuss about the implementation ideas for the blackjack card game project. First, we will step into discussing the project background and description which includes the basic game rules and the explanation about the starting code. Next comes the project scope where the team members names, and the work assigned to them is listed. This also goes into the description about what we are expecting for our projects interface and how we are going to make out if it is completed. Third comes our high-level requirements for our project. Then comes the basic implementation plan for the starter code. Finally, describing the OOP principles and citing examples from the starter code.

1)PROJECT BACKGROUND AND DESCRIPTION:

The project aims to develop a robust and interactive blackjack game with a user-friendly interface. The final vision is to create a coherent gaming experience, allowing players to enjoy the thrill of blackjack from the comfort of their own devices. The game should follow the standard blackjack rules and provide clear instructions and scores at ever stage.

Blackjack is card game played between the players and a dealer. The goal of a game is to get a hand value to 21 without exceeding it (wikiHow, 2019). Each card has a specific value.

The game follows the basic rules:

- 1) Players place their bets.
- 2) Players and the dealer are initially dealt with two cards each. The player can see their two cards and one of the dealer's cards.
- 3) Numbered card 2 to 10 are worth their face values, for the face cards jack, king, and queen the count is 10 whereas for the aces the value can be 1 or 11 you can decide it's value throughout the game if you have one.
- 4) The player can choose to "hit" and receive another card to increase their hand values or "stand" to keep their current hand.
- 5) If the player's hand value exceeds 21, they bust and loose the game.
- 6) If the player's hand value hits 21, the player gets one and half if their bet.
- 7) One the player stands; the dealer reveals their face down card and hits until their face value is 17 or greater.
- 8) If the dealer bust every player in the game receives twice their bet and if the dealer doesn't bust only the player whose card value sum is greater than that of dealer wins twice their bet.

(wikiHow, 2019)

Base code is providing a base framework that can be expected and modified for additional card games. It contains four classes Card, Game, Group of cards and Player with .java extension. Card class is an abstract class used to implement methods for their subclasses (public abstract String toString()). Game class is also an abstract class that models the game. It includes the data members for the name of the card game and uses ArrayList of the players and incorporates their get and set methods as well. It sets methods to implement in their child classes (public abstract void play(): method call to play the game and a public abstract void

declareWinner(): to declare the winner). GroupOfCards is a class build with the ArrayList and Collection classes import to shuffle the cars, get cards from the deck of cards and to get and set the size of the deck. The code follows common coding conventions, such as using meaningful variable and function names, proper indentation for blocks, and comments for clarity.

The code implements the game logic, including card shuffling. The code is written in java, a high-level language. It likely uses object-oriented programming principles to organize the game components into the classes, “GroupOfCards”, “Player”, “Card”, and “Game”. These classes likely contain methods and attributes to manage the game state and perform necessary operations.

2)PROJECT SCOPE:

Team members are:

1) Kaitlin Saqui

- Deliverable 1 Task(s):
 - i. Prepared and organized the GitHub repository for all members to access the base code, with separate branches for each member.
 - ii. Created the Class Diagram on Visual Paradigm and added it to the GitHub repository.
- Deliverable 2 Task(s):
 - i. Design the Use Case Diagram(s) on Visual Paradigm based on the rules of the game and will act as a visual representation of how a user may interact with our game and the details of our system.
 - ii. Produce narratives of the main path, along with the alternate paths.

2) Arshvir Singh:

- Deliverable 1 Task(s):
 - i. Worked on the following sections of the Design Document: High-Level Requirements and Implementation Plan
- Deliverable 2 Task(s):
 - i. Prepare and work on the Design Document for the following sections: Project Background and Description, and Design Considerations

3) Mandeep Kaur:

- Deliverable 1 Task(s):
 - i. Worked on the following sections of the Design Document: Design Considerations
- Deliverable 2 Task(s):
 - i. Prepare and work on the Design Document for the following sections: Project Background and Description, and Design Considerations

4) Manpreet Kaur:

- Deliverable 1 Task(s):
 - i. Worked on the following sections of the Design Document: Project Background and Description, and Project Scope
 - ii. Organized the Design Document to ensure all sections were professionally presented and well articulated.

- Deliverable 2 Task(s):
 - i. Design the Class Diagram on Visual Paradigm in which it will closely follow the rules of the game and will act as a visual blueprint for the object-oriented system.

TECHNICAL SCOPE OF THE BLACKJACK GAME PROJECT:

USER INTERFACE: The project will include a user-friendly interface that allows players to interact with the game. The interface should display the player's hand, the dealer's hand, and relevant game information such as current bet amount, total balance, and outcome of each round.

GAME LOGIC: The project will implement the core game logic of blackjack, including shuffling and dealing cards, calculating hand values, evaluating win/loss conditions, and managing the progression of the game. It should handle player decisions, such as hitting or standing, and simulate the dealer's actions according to the rules of blackjack.

RULES AND VALIDATIONS: The game should enforce the standard rules of blackjack, including the value of cards, valid moves, and win/loss conditions. It should validate player inputs and prevent any illegal or invalid moves. Additionally, it should manage the player's balance, and handle bets.

ERROR HANDLING: The project should incorporate robust error handling mechanisms to gracefully handle unexpected situations or errors that may arise during gameplay. It should provide informative error messages or prompts to guide the player and ensure a smooth gaming experience.

TESTING AND DEBUGGING: The codebase should be thoroughly tested to identify and fix any bugs, or logic errors, that could potentially impact the game's functionality or user experience.

DOCUMENTATION AND USER GUIDE: The project should be well-documented, including inline comments and a separate user guide or documentation that explains the game rules, controls, and any additional features or options. Clear instructions and explanations will help players understand and enjoy the game.

COMPLETION CRITERIA:

The project can be considered complete when it satisfies the following criteria:

- The game logic is fully implemented, and the basic rules of blackjack are accurately followed.
- The user interface provides a visually appealing and intuitive experience, displaying the necessary game elements and allowing players to make decisions and place bets.
- The game validates player inputs, enforces rules, and handles edge cases effectively without crashing or producing unexpected behavior.
- Adequate testing and debugging have been conducted to ensure the stability and correctness of the codebase.
- The project is well-documented, including inline comments within the code and a user guide or documentation explaining the game rules and controls.
- The game provides an enjoyable and engaging experience, simulating the excitement of playing blackjack at a casino.

By meeting the above criteria, it can be concluded that the project has successfully achieved its technical goals and is ready for deployment and usage by players.

3)HIGH-LEVEL REQUIREMENTS:

The game will include the following:

- **SIGN-UP:** players will be able to sign-up using their name
- **CARDS' VISIBILITY:** players will be able to see other player cards and one of the dealer cards whereas, dealer will be able to see everyone's cards.
- **DECLARATION OF WINNER:** The game automatically determines if a player has won, lost, or pushed (tie) based on the card values in a player's hand.
- **BETTING:** Players will be able to bet initially in the game and at the beginning of every new round if a player chooses to play again.
- **PLAYER'S DECISION:** Players can choose to hit or stand after viewing their hand of cards.
- **BUST CONDITION:** Either a player or dealer can bust, but if both the player and dealer bust, the dealer wins.

4) IMPLEMENTATION PLAN:

Here is the link to our public repository used for Deliverable 1:

https://github.com/LinSaki/Group5_Deliverable1

This repository will be used by the group members to work on this project together by initially cloning it, and then committing, pushing, and fetching/pulling the code through NetBeans. The plan is to set goals in the beginning of the week, each member can work on the project according to their flexible time and check the code at the end of the week or as scheduled by the team members. In this repository, the document directory will contain all the text files and pdfs. Whereas UML diagrams will be stored under the diagram directory.

The coding standard that we intend to use is as follows:

NAMING CONVENTION:

- 1) Package names are to be typed in lowercase.
- 2) Class, Enum, interface, and annotation names will be typed in UpperCamelCase.
 - a. Example: Game, GroupOfCards, etc.
- 3) Methods and variable names typed in a lowerCamelCase.
 - a. Example: getPlayers(), setName().

DECLARATION:

- 1) One declaration per line.

- 2) The starting curly brace ({) will be at the end of the same line as the declaration statement or method and the ending curly brace (}) will be on a separate line and aligned with the conditional/class.

FILE ORGANIZATION:

- 1) Blank lines will be used to improve readability by grouping sections of code that are logically related.
- 2) Each Java source file will contain only a single class or interface. Each class will be placed in a separate file.

COMMENTS:

- 1) Comments/Javadoc in a separate line preceding any class, loop, function, method etc.
- 2) Ending comment in the same line of ending braces of any loop, function, method etc.

The tools going to be used are:

- 1) Visual Paradigm: It will be used for making UML class diagrams and use case diagrams.
- 2) Apache NetBeans: It will be used for coding the project to create a functional game.

5) DESIGN CONSIDERATIONS:

ENCAPSULATION:

The code serves as an example of encapsulation by using access modifiers to limit access to class members. For instance, the name variables in the Game class includes access and modification methods (getters and setters), for example: getName(), getPlayers(), and setPlayers() are marked as public. By doing this the class's internal state, such as the private data fields “name” and ArrayList “players”, are controlled, and only declared methods can access it.

DELEGATION:

The code's (Card.java, Game.java, and Player.java) classes use abstract methods. The class's internal state is controlled, and only declared methods can show delegation. Subclasses are required to implement the behaviors that these abstract methods provide. By providing the implementation information to the subclasses, the base classes can establish a similar interface while allowing different implementations in different games or player kinds.

FLEXIBILITY AND MANAGEABILITY:

The code is designed to be flexible and manageable by providing a base framework that can be expanded and modified for additional card games. The abstract nature of the (Card.java, Game.java, and Player.java) classes allows for the creation of specialized child classes that implement logic specific to a particular game. This method makes it simple to tweak and include new features or games without significantly changing the base framework.

COHESION: -

- The Game class, which has great coherence, includes all of the essential aspects of a game, including player management, gameplay, and winner announcement.
- The Card class has a high level of cohesion since it may represent a generic card and provides a mechanism to obtain a string representation of the card.
- The Player class has a high level of coherence since it replicates a person in the game and provides a method to participate.
- The Group of Cards class has a high level of coherence since it represents a collection of cards and provides methods for getting the cards, rearranging them, and adjusting the group's size.

LOOSE COUPLING:

- The classes interact through abstractions and interfaces that are explicitly specified, which causes a loose connection between them. For instance, the Game class uses the Array List interface to communicate with Player objects, allowing for more flexible development of different player types.
- The Game class is reliant on the Card class because the abstraction provided by the toString() method can be implemented differently for various card games.
- Because the Group of Cards class doesn't directly depend on any other classes in the provided code, it is loosely coupled.

Encapsulation, delegation, flexibility, maintainability, cohesion, and loose coupling is all supported by the code's general application of object-oriented programming principles.

CONCLUSION:

In conclusion, we have discussed the ideas to build our project. Project background and description, project scope, high-level requirements, implementation plan and the design considerations.

REFERENCES:

wikiHow. (2019, November 11). How to Play Blackjack [Video]. YouTube.

<https://www.youtube.com/watch?v=eyoh-Ku9TCI>