Deliverable 1 – Group 7

By: Malachi, Hai, Khoa, Judd

SYST17796 Fundamentals of Software Design and Development

Dr. Nagma

Contents

Team Contract

SYST 17796 TEAM PROJECT

Team Name: Group 7

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: MALACHI BAMPOE	Malachi Bampoe	991734540
THE HAI NGUYEN	The Hai Nguyen	991745555
JUDD ROSAYAGA	Judd Nikolai Rosayaga	991748399
ANH KHOA TRAN NGUYEN	Anh Khoa Tran Nguyen	991762341

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.

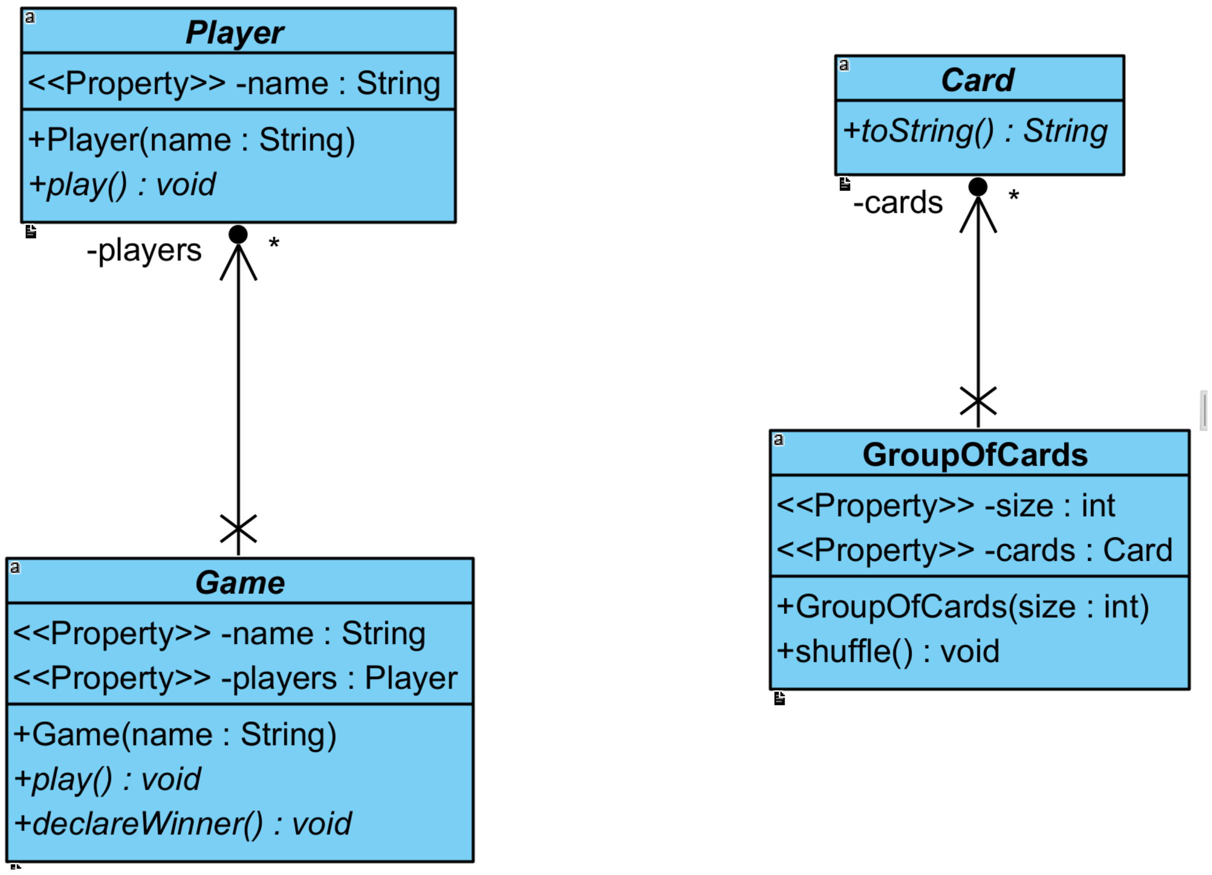
What we will do if . . .

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.	M.B.	Project leader emails the student citing the concerns and cc's the professor so they are aware of the situation at the very onset _X_ (**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.	M.B.	a) Team absorbs workload temporarily _X_ 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.	M.B.	a) Team reassigns component __ b) Team helps member _X_ c) Team member must ask professor for help __ d) __ Other (specify):
Team member does not deliver component on time	M.B.	a) Team absorbs workload __ b) Team member(s) ask professor to request

Scenario	**Accepted initials**	**We agree to do the following (Put an X corresponding to your choice in each box)**
due to lack of effort.		a Participation Form from <u>all</u> team members. This *may* result in individualized grades being awarded for a deliverable __ c) Both a. and b. above _X_ 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.	M.B.	a) Team agrees to abide by majority vote _X_ 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.	M.B.	a) Team members will draw on each other's strengths to help bring the quality of the deliverable to a minimal acceptable level _X_ 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	M.B.	a) Team agrees to avoid use of all vocabulary inappropriate to a business/college setting __ b) Team attempts to resolve the issue by

Scenario	Accepted initials	We agree to do the following (Put an X corresponding to your choice in each box)
making one or more member(s) feel uncomfortable.		airing the problem at a team meeting _X_ 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	M.B.	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 _X_ 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	M.B.	a) Team forces decision sharing by routinely voting on all issues __ b) Team routinely checks with each other about perceived roles _X_ c) Team discusses the matter at team meeting __

UML Diagram


Player
<<Property>> -name : String
+Player(name : String)
+play() : void
-players
*
Game
<<Property>> -name : String
<<Property>> -players : Player
+Game(name : String)
+play() : void
+declareWinner() : void
Card
+toString() : String
-cards
*
GroupOfCards
<<Property>> -size : int
<<Property>> -cards : Card
+GroupOfCards(size : int)
+shuffle() : void

Design Document

1. Project background and description

Final Vision:

The goal of the project is to allow the user to play a game of blackjack.

How to play:

To play blackjack both the player and the dealer are given 2 cards at random. The player is then given the option to “hit” or “stand”. If the player chooses “hit”, they receive another card. If the player chooses to “stand”, the player keeps the cards and waits for the dealer’s turn. The dealer then chooses whether to “hit” or “stand”. The point of the game is to try to get a total of 21 or as close to it as possible with the total sum of their cards’ values. If the player wins, they receive money added to their budget. If the player loses, they lose the amount that they bet.

Special Rules:

- If either the player or the dealer receives an Ace card, they have the option to choose whether to have the Ace value set to 1 or 11.
- If either the player or dealer receives 21, they automatically win
- If both the dealer and player receive 21, they draw and nobody wins anything.

Project Technicalities:

- The game is coded in Java.
- Player information, card information, dealer algorithm, score and money calculations will be held in different java classes outside of the main body
- More Java classes outside of what is provided in the base code will be added upon requirement
- While loops are used to process the player’s input.
- Enums are used for card suites and values.
- Match case statement is used for the player to choose the value of the Ace card.

2. Project Scope

Team Members:
Malachi – Team Contract, Design Document
Hai – Git Repository, Design Document
Khoa - Class Diagram
Judd - Class Diagram, Design Document

The project will be complete once all our goals are met, and our final vision displays as we planned. You will be able to play blackjack against a dealer. You can bet as much as you want and can decide whether to hit or stand. The dealer will see your final score and then play after you, ensuring the casino has the best possible chance of winning. You can quit once the game is done or choose to continue. The game will end once you decide to quit or once you run out of chips to bet.

3. High-level requirements

- Ability to register your name
- Ability for the game to communicate a win or loss
- Ability to player to know their amount of held chips at any given point
- Ability for player to hit or stand
- Ability for player to choose what the Ace will turn in to (1 or 11)
- Dealer will choose cards after player turn
- Dealer will always hit if player score is greater than dealer score and less than 21
- If player score is 21 at start, player wins
- If dealer score is 21 at start, dealer wins
- If both scores are 21, draw (no money exchange)
- Ability to bet any amount of money that you currently have
- Win double amount of bet money when player wins
- Lose all bet money when player loses
- Ability to quit at any time
- Once player money runs out, game ends

4. Implementation plan

Git Reposity URL: https://github.com/thehainguyen/Blackjack.git

Will meet up once a week and discuss code. Code will be added/edited then if needed. All work will be stored in 1 folder when submitting, with subfolders labeling each of them.

Tools: Visual Paradigm, NetBeans

5. Design considerations

Loose Coupling: Reduces the dependency that components have with each other to allow changes to be made while not altering the other components.

Two examples are:

1. The card class is independent of the whole game only dealing with the values of the cards that are being dealt allowing us to "hit" a card and choose the value of the ace card.

- The MainPlayer class and the Dealer class extending the Player abstract class. This allows changes on one not affect the other.

Encapsulation: Encapsulation is used to hide data within a class.

Two examples are:

1. The algorithm for how the dealer makes choices will be hidden from the player class only returning the dealers choices.
2. All properties of a class are private and can only be accessed by the getter and setter methods

Flexibility/Maintainability: Ensures the code is easy to update or fix in the future, even by those who did not work on it

Two examples are:

1. Well commented out code that explains what each part of the program is for or what it does. This helps other developers understand the code quickly if updates need to be made later.
2. Use relevant names for methods and variables, making the code simple to read and follow. This reduces mistakes and makes it easier to fix any problems in the future.

Delegation: The process of distributing tasks from one object to another

Two examples are:

1. Separating the tasks of checking if the player won and calculating the bet values in separate classes from the MainGame class.
2. Separating the task of checking if the input and asking the player options into smaller classes outside of the MainGame class.