STUDENT: Mohammad Adeel Khilji

STUDENT NUMBER: 991636478

COURSE: Software Design Fundamentals

COURSE CODE: 1215-42219

ASSIGNMENT: Deliverable 1 of 3

PROJECT: Term Project - Card Game (WAR)

DUE DATE: 13th of July, 2021

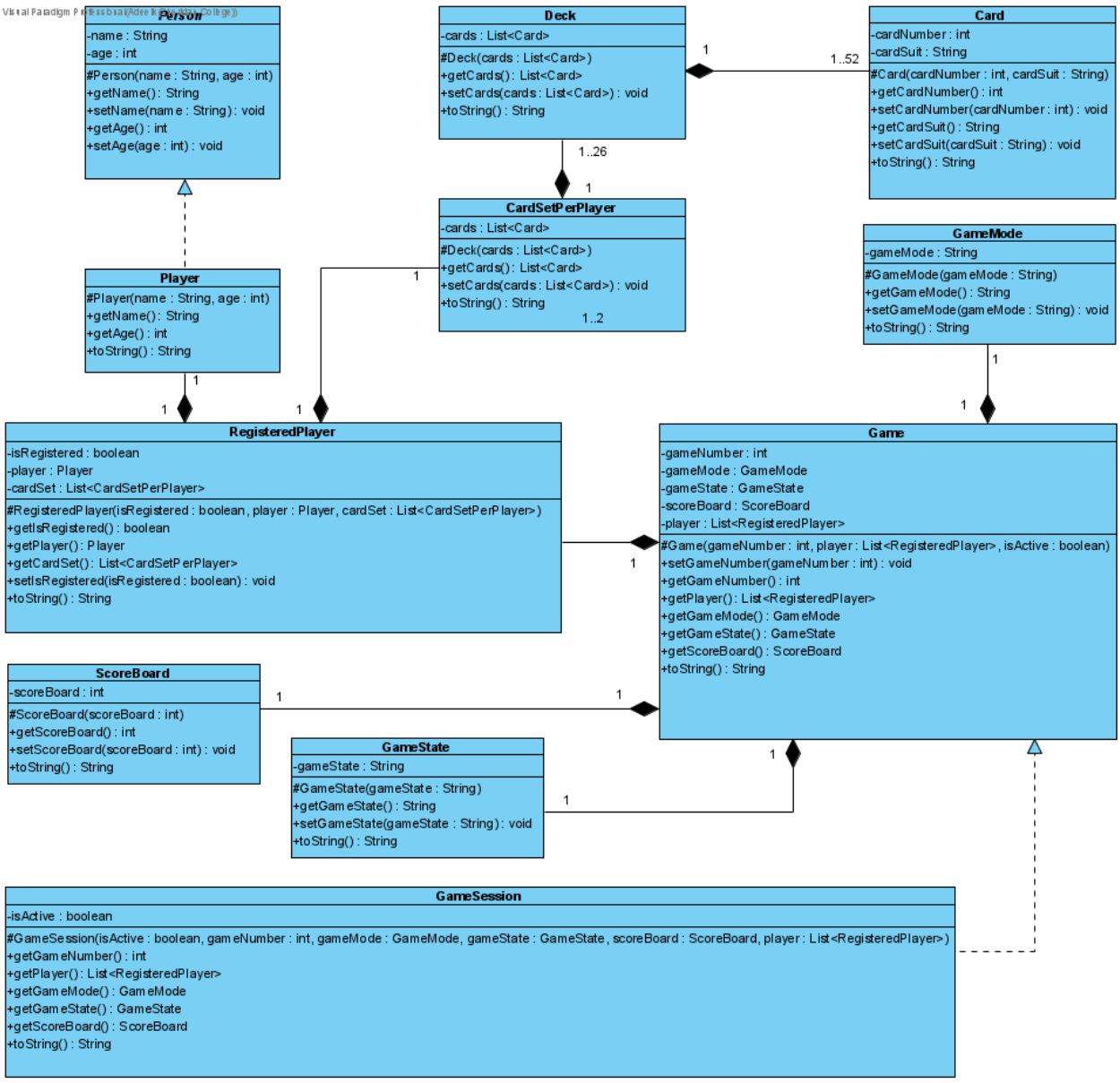
DISCLAIMER: This base project is subject to change based as the it progresses, based on the requirements and coding boundaries. This is just a progress report of the initial code. Documents will also change over time based on the requirements and code progression.

TABLE OF CONTENTS

UML CLASS DIAGRAM 2
PROJECT BACKGROUND AND DESCRIPTION 3
PROJECT SCOPE 4
HIGH-LEVEL REQUIREMENTS 5
IMPLEMENTATION PLAN 6
DESIGN CONSIDERATIONS 7

UML CLASS DIAGRAM

Following is the initial class diagram based on the initial user requirements.



Class diagram is subject to change as the project progresses. This will depend on how this code is implemented and structured while coding. It may also change based on the further business requirements if needed.

PROJECT BACKGROUND AND DESCRIPTION

For this project, I decided to pick a card game called WAR. It is a two-player based game, played with 52 cards

HOW TO PLAY:

- Each player is given 26 cards and they must be faced down in a pile in front of them.
- Each player draws a top card from their pile and place them in the middle.
- During WAR, each player draws four cards all facing down except for the top card which is facing up.

RULE OF THE GAME:

- Aces is the highest card in the game.
- 2 is the lowest card in the game.
- The highest card wins the round.
- If the cards are the same, it is WAR.
- If During war, the card that is facing up is the higher then the war card facing up, player wins.
- If the cards drawn during war are the same, you repeat drawing four cards.
- If a player runs out of all cards, he loses the game. The other player wins the game.
- If a player runs out of all cards during the WAR, the player loses the game. The other player wins the game.

INITIAL CODE BASE:

For this project I will be using Java, which is a general-purpose programming language, I will be using object-oriented principals such as abstraction, inheritance, encapsulation, with high cohesion and loose coupling. Later in the project I will be using polymorphism and get into more details by separating generalized classes and specific classes relevant to generalized classes. This will change over time if more constructive procedure comes into place or requirements force me to change the code. Coding conventions, for classes will be Pascal case, where first letter of each word must be capital, this will allow me to write proper names of each class. The rest of the coding conventions will be camel case where first letter of the first word is small and rest of the words start with capital letter; for instance variables, accessors, and methods. The naming convention of initial code is as follows:

PLAYER CLASSES:

- Person (Abstract)
- Player (Subclass to Person)
- RegisteredPlayer (Composite)

CARD CLASSES:

- Card (Composite)
- Deck (Composite)
- CardSetPerPlayer / Hand (Composite class to player)

GAME CLASSES:

- Game (Generalized class)
- GameSession (Sub class of Game)

- GameMode (Composite class to Game)
- GameState (Composite class to Game)
- ScoreBoard (Composite class to Game)

PROJECT SCOPE

My name is Mohammad Adeel Khilji, and I will be working alone on this project. I will be making a game called War. It is a 52-card based, turn based game. Each player deals with 26 cards, each player draws a card, and the highest card wins the round. The one who collects all the cards wins the game. There is a state of war in this game, which means the game changes from normal to war where you draw four cards, three faced down and one faced up. The completion of the project is met after the following:

- Players are registered.
- 52 cards are created and added to the list. (randomized)
- Cards are distributed amongst players; each player should have 26 cards (not randomized but from the same deck of cards that was shuffled.
- Game is created and displays appropriate game state as per rules.
- Player can draw cards in the normal mode.
- Player can draw four cards in war mode.
- Score is properly calculated and displayed, during and when the game ends. This should also display who won and who lost.
- Game mode is displayed appropriately.
- Game state is displayed appropriately.
- All the rules and user requirements are met.

HIGH-LEVEL REQUIREMENTS

The War card game must include the following:

Business Requirements	User Stories
Ability for each player to register with the game	As a player, I want to register with the game, so that I can participate in the game.
Ability for the game to communicate a win or loss	As a player, I want to know the outcome of the game, so that I know whether I won or lost a game.
Ability for players to know their status (score) at all times	As a player, I want to know the score of the game at any time, so that I can know the status at all times.
Ability for drawing the card from the set	As a player, I want to draw the card each round of the game session, so that I can play my turn in the current round.
Ability for drawing four cards from the set, top one facing up	As a player, I want to draw four cards if the cards. match for the round, so that I can play my turn for war in current round of war.
Ability for entering a game session	As a player, I want to enter a game session so that I will know that I am in a game.
Ability for viewing opponent's name while in the active game session	As a player, I want to know the name of my opponent, so that I can know whom I am playing against.
Ability for viewing active turn for player while in the active game session	As a player, I want to know who's turn is it, so that I can know when it is my turn.
Ability for viewing number of cards in the players pile or stack	As a player, I want to know how many cards I have left per round, so that I can keep track of my cards.
Ability for player to know they are in active war	As a player, I want to know the type of session, so that I know whether I am in a war session or a normal session of the game session.
Ability for player to know the round has ended	As a player, I want to know that the round has ended, so that I can keep track of each rounds.

IMPLEMENTATION PLAN

The documentation of the project and code will be found on the following repository.

BITBUCKET REPOSITORY URL:

https://bitbucket.org/AdeelKhilji/termproject_cardgame_war/src

I will be creating a folders for each aspect of this project. DesignDocuments for Action Diagrams and UML diagrams, I will be using Visual Paradigm for design documents, the documentation such as user stories will be in the separate folder called BusinessRequirements, and for the actual project I will be using a folder called CardGameWar and follow coding and code-naming conventions for project as stated above. I will be using Intellij IDEA for this project unless instructed otherwise by the professor to use netbeans but since the project requirement is to have the whole project including the documentation in a repository, I will be using commandr and git commands for version control.

I will be pushing code to the stated bitbucket repository every two to three days, depending on the workload that I am currently dealing with. If workload is taking more time, I will push the updated code on the fourth day.

Structure of the entire project should be as follows:

Project_CardGameWAR:

DesignDocuments:

- Action Diagrams
- Class Diagrams

BusinessRequirements:

- Business Requirements
- Other Business Requirements

Project:

- CardGameWar(Programming project)

DESIGN CONSIDERATIONS

For this project I will be using all pillars of object-oriented programming such as:

ABSTRACTION

- This will be used for the classes that have a specific method which is reused in multiple sub classes.
- Person class have person content and could have a method that can determine player number.
- Person class can also have content related to non player characters that may require the same attribute(s) and methods but are not handled by a player manually.
- Game class can have generalized attributes and general methods that can be reused in the GameSession
- Game class can also have attributes and same methods that can be reused in GameMode

INHERITANCE

- This will be used for subclasses that inherits from super classes and extends to further subclasses.
- GameMode extends Game class.
- GameState class that inherits all the contents of Game and handles each round and turns each round. It is a turn-based game after all.

ENCAPSULATION

- This will be used in all of the necessary classes.
- Besides abstract, unless being used in polymorphic pattern.
- Each non abstract class will contain attributes, a constructor to initialize the attributes.
- Each non abstract class will also contain get and set methods for each attribute of that class.

POLYMORPHISM

- This might be used in necessary classes and may allow me to understand how to approach high cohesion and loose coupling further down the development process.
- There could be further classes detailing the player stats (not implemented yet since it is not one of the requirements) that can display the number of cards he's left with each round.
- There are other classes such as PlayerStats and GameStats (also not implemented yet since it is not one of the requirements) that can be implemented to constantly update the score of each player involved in a game session, and how many cards player is left with. These are not implemented yet.

I will also be using best practices such as:

HIGH COHESION

- This will be used through out the entire project; each class will have its own contents.
- Player will have player contents.

- GameState could have contents related to GameSession and GameMode. It will also have turns, rounds and other necessary classes that will fulfill high cohesion as the project progresses further.

LOOSE COUPLING

- Some classes will be defined later, as it is too early in the project to explain this but there will be other classes that will handle only with methods represented by that class.
- Randomizing deck of cards before distributing them to players involved in a game session can be done in a separate class.
- Calculating score and win or lose will be determined in separate classes.