SYST 17796 DELIVERABLE 1
DESIGN DOCUMENT TEMPLATE

OVERVIEW

1. Project Background and Description

Our group project builds on the Sheridan SYST17796 starter code to implement a playable card game in Java. The starter code gives us the fundamental classes Card, GroupOfCards, Player, and Game. These are a beginning with basic card and player skeletons that we can build upon in order to implement our own game solution.

We decided to create the game UNO, because it's a fun, popular card game with a good balance of rules and strategy.

UNO rules :

- 7 cards are dealt to players initially.
- Players take turns playing one card from their hand that is the same color or number as the top card on the discard pile.
- Special cards like Skip, Reverse, Draw Two, Wild, and Wild Draw Four interrupt the game flow.
- If a player can't play, he must draw a card.
- The first one to run out of all his cards wins but must say "UNO!" when he still has one card left.
- Base Code Description:
- The initial code is written in Java. It uses object-oriented methods:
- Encapsulation: attributes like cards inside GroupOfCards and name inside Player are private and referred to by methods.
- Delegation: Game class controls the sequence of play but leaves details to Player and GroupOfCards.
- Flexibility: because Game is an abstract class, we can subclass it to our own UnoGame class without altering the base code.

2. Project Scope

Team Name: Code-Crusaders

Team Members and Roles:

Project Leader – Sankalp Taneja

UML/Design Specialist – Gurjap singh

Developer/Tester – Gurjap singh

Documentation Lead – Jaspinder Kaur

Technical Scope:

- It will be a console-based Java game. No GUI should be added in Deliverable 1, as the focus lies on object-oriented design and rules of the game.
- The project is considered complete when:
- Multiple players can join a game.
- The deck and discard pile is managed properly.

- Special cards (Skip, Reverse, Draw, Wild) are implemented.
- Turns are rotated in the right manner and the game ends when a player exhausts all their cards.
- The game announces a winner at the end.

3. High-Level Requirements

The UNO game must have:

- Feature to enable joining and registering of players.
- A deck to manage drawing, shuffling, and discarding.
- Proper turn rotation (Reverse/Skip included).
- Action cards implemented (Draw Two, Wild, etc.).
- Some way to indicate a player when they reach one card ("UNO").
- Automatic winner declaration once a player has no cards left.

4. Implementation Plan

Repository URL: [https://github.com/Gurjap-singh/Code-Crusaders]

How we'll use the repo:

- Everyone pulls the latest code before working, and pushes changes after.
- Commit messages must explain what was changed.
- The instructor will have access (repo is public / or private with invite).

Folder setup:

- /SYST17796_ProjectStarterCode → main Java code
- /docs → UML diagram, Team Contract, Design Document
- /tests → JUnit tests (for later deliverables)

Tools & Standards:

- IDE: NetBeans or IntelliJ
- UML Tool: Visual Paradigm
- Version Control: GitHub
- Testing: JUnit (later)

Coding conventions:

- Classes in PascalCase (UnoGame, UnoPlayer)
- Methods/variables in camelCase (drawCard(), playTurn())
- Javadoc for methods and consistent indentation

5. Design Considerations

Encapsulation:

- Player's hand (cards) is private, and only the Player object can deal with it.
- Deck (GroupOfCards) keeps its cards list private and provides controlled methods to draw or shuffle.

Delegation:

- UnoGame will be delegating turn actions to UnoPlayer.
- GroupOfCards handles shuffling and dealing, so the game doesn't need to worry about card storage.

Maintainability/Flexibility:

- The Game class abstraction makes it easy to add UNO without needing to re-code the base.
- Everything UNO-specific (special cards, rules) will go into UnoGame and UnoPlayer, so the original framework can be reused.
- Adding scoring or other rules later will be easy by extending without altering current code.