Object Oriented Programming

Assignment Ujjwal Jain and Prajwal Ranjan

1 About UNO!

UNO! is a multi-player card game based on strategic disposal of cards. It consists of two kinds of cards, namely, Normal Cards and Special Cards. The Normal Cards are divided into four colours, Red, Yellow, Green and Blue, and each colour consists of cards numbered from 0 to 9. The Special Cards consist of 3 Cards belonging to each colour, namely, Skip, Reverse and Draw 2 (or +2). There are also two cards, Draw 4 (or +4) and Wild, not belonging to any particular colour.

1.1 Rules

ˆFour players are needed to start the game.

Each player begins with 7 Cards.

ˆ Two decks are available to the players, the Draw deck and the Discard deck .

ˆ Players are allowed to discard a card from their hand to the Discard Deck if,

They have a card of the same colour as the topmost card

They have a card with the same number as the topmost card

They have a Draw 4 or a Wild Card

ˆ In the case of the top card being a Special Card,

If the top card is a Reverse, then game direction reverses and the next player can discard any card of the same color or a wild card.

If the top card is a Skip, then either the player can discard if they have a Skip, or, the turn moves to the next player

If the top card is a Draw Two, then either the player can discard if they have a Draw Two, Draw Four, a Reverse or a Skip, or, they must draw (pick) two cards

If the top card is a Draw Four, then they must draw (pick) four cards

ˆ The game ends when a player does not have any card left

2 Classes Implemented

2.1 Card Classes

2.1.1 Card

A simple class consisting of basic getter and display methods. Similar to an interface

2.1.2 NormalCard

A class extending the Card class, used to represent the basic Numbered cards and their colors. Methods inherited are overridden in the definition.

2.1.3 SpecialCard

A class extending the Card class, used to represent Special cards (Draw Two, Draw Four etc.). Methods inherited are overridden in the definition.

2.2 Player Classes

2.2.1 Player

This class is used to describe the player. It consists of basic attributes, such as player name, player number and the player's hand. It also consists of methods used to describe a player's turn.

2.3 Pile Classes

2.3.1 DrawPile

This class is used to manage a stack of Cards that would be presented to the Player to draw from.

2.3.2 DiscardPile

This class is used to manage a stack of Cards into which a Player would discard a Card into.

2.4 Utility Classes

2.4.1 DeckInitializer

This class is used to create the initial Draw and Discard piles at the beginning of the game.

2.5 GUI Classes and Other classes

2.5.1 StartMenu

This class is used to build the main menu in GUI form.

2.5.2 InitializePlayers

This class is used to Initialize Players with their names

Exactly four players are needed to begin the game

2.5.3 Game

Main game class displaying the top card of discard pile, Player name and turn, and Player's deck.

2.5.4 CardPopup

CardPopup class makes a popup window appear after clicking a card in the player deck in the Game class.

It make the player see if the card is usable or he has to draw a card for his turn

2.5.5 ChooseColor

This window appears when a wild card is played to let the player choose the color for his next turn.

3 Principles of OOP Applied

3.1 Encapsulation

Various parts of the program have restricted access by other parts of the program. Many classes also have getter/setter functions for accessing private fields.

3.2 Abstraction

In the main game methods, only specific parts of classes such as Card classes are displayed to the user, exemplifying Abstraction.

3.3 Inheritance

Most of the classes have inherited methods from other classes, for example,

the NormaCard and SpecialCard class have inherited methods from a Card class. Other components such as the GUI component have inherited methods and attributes from Generalized classes.

3.4 Polymorphism

In the Deck Classes, polymorphism has been used in the DeckInitializer class, in order to create the Discard deck using either a single card or no cards.

4 Design Pattern Followed

ˆ We have followed the Builder Design pattern while working on this project ˆ Basic components such as the Card etc. were first created

ˆ These individual components were later on incorporated together in the main game class

ˆ The overall structure of the program is similar to a building built with bricks, where bricks represent individual components such as the Card class etc.