Projekt **Pogramowanie obiektowe (Java)**

Wydział Elektrotechniki Automatyki i Informatyki

Politechnika Świętokrzyska

Studia: **Stacjonarne I stopnia**	Kierunek: **Informatyka**
Temat projektu: **Gra UNO**	Grupa: **2ID11A**
	1. Yevhenii Stavnychyi 2. Szymon Paweł Baran

1 Ogólny opis projektu

1.1 Co to jest Uno?

UNO – amerykańska gra karciana, zbliżona koncepcyjnie do Pan. Do gry w UNO używa się specjalnej talii kart.

Talia liczy w sumie 108 kart i składa się z:

Kart zwykłych:

- 19 czerwonych kart ponumerowanych od 0 do 9 (jedna karta z numerem 0 oraz po dwie karty z numerami od 1 do 9)
- 19 zielonych kart ponumerowanych od 0 do 9 (jedna karta z numerem 0 oraz po dwie karty z numerami od 1 do 9)
- 19 niebieskich kart ponumerowanych od 0 do 9 (jedna karta z numerem 0 oraz po dwie karty z numerami od 1 do 9)
- 19 żółtych kart ponumerowanych od 0 do 9 (jedna karta z numerem 0 oraz po dwie karty z numerami od 1 do 9)

oraz kart funkcyjnych:

- 8 kart stopu (Skip) po dwie z każdego koloru
- 8 kart zmiany kierunku (Revers) po dwie z każdego koloru
- 8 kart +2 (Draw two) po dwie z każdego koloru
- 4 czarne karty +4 ze zmianą koloru (Wild +4)
- 4 czarne karty zmiana koloru (Wild)

1.2 Zasady gry

Graczy może być od 2 do 4. Na początku gry rozdaje się po 6 kart każdemu graczowi i jedną z talii kładzie się na środek. Gracz musi dopasować swoją kartę numerem, kolorem lub symbolem do odkrytej karty. Jeżeli gracz nie posiada żadnej karty pasującej do tej odkrytej, musi pociągnąć kartę z talii. Jeśli wyciągnięta karta pasuje do odkrytej, jeszcze w tej samej kolejce gracz może ją dołożyć. Jeżeli nie - ruch ma kolejny gracz. Nie ma przymusu w dokładaniu kart. Gracz, który jako pierwszy zostanie bez kart, wygra(2 gracz na grugim miejscu i td.).

W talii są także karty specjalne takie jak Skip, Reverse, Draw two, Wild + 4, Wild.

- Skip - następny gracz traci (stoi) kolejkę

- Reverse - karta zmieniająca kierunek gry
- Draw two - następny gracz bierze dwie karty
- Wild + 4 - zagrywający kartę deklaruje zmianę koloru na dowolnie przez siebie wybrany, następny gracz bierze 4 karty.
- Wild - zagrywający kartę deklaruje zmianę koloru na dowolnie przez siebie wybrany (jeden z kolorów dostępnych w grze).

Wygrywa ten, kto pierwszy pozbędzie się wszystkich kart.

1.3 Zastosowane technologie

Do wykonania projektu zastosowano środowisko programistyczne Eclipse IDE.

1.4 Uruchamianie

Aplikację uruchamia się przez program Eclipse IDE.

2 Pakiety aplikacji

2.1 Pakiet Uno

2.1.1 UnoApp.java

UnoApp.java jest programem pozwala na uruchomienie aplikacji.

Ten program jest połączony z innymi programami:

- Uno.game.GameManagement

Ten program zawiera implementację aplikacji i przeciążoną metodę startThread() klasy GameManagement.

2.1.2 Deal.java

Deal.java jest programem, który zawarte są tam metody umożliwiającą przyszłe „play again”.

Biblioteki użyte w programie Deal.java:

- Java.util.ArrayList

Ten program zawiera class Deal który dziedziczy CardHadler, tablicę kart(w ręce) i inne metody: metodę addCard, removeCard, getCard, getSize, printArray, getLast.

2.1.3 Player.java

Player.java jest programem, który inicjuje działanie gry gracza.

Biblioteki użyte w programie Player.java:

- java.util.InputMismatchException;
- java.util.Random;
- java.util.Scanner;

Ten program zawiera metody które realizują działania gracza: enemyGame, getEnemyChoice, findWild, findDTwo, findSkip, findReverse, findDrawFour, findPlayable, hasWild, hasSkip, hasReverse, hasDrawTwo, hasDrawFour, hasPlayable, wildEnemyColor, checkDraw, draw2, draw4, getCardNumber, printDiscard, wildColor, printHand, checkUno.

To była implementacja gry w konsoli, która nas wzorowała jak ma wyglądać ogólnie gra.

2.1.4 CardHandler.java

CardHadler.java jest programem, który zawarte są tam talia kart i metody.

Biblioteki użyte w programie CardHandler.java:

- Java.util.ArrayList;
- Java.util.Collections;

Ten program zawiera własną tablicę kart i inne metody: shuffleDeck, getLast, addCard, printDeck, getSize, removeLast.

2.1.5 Card.java

Card.java jest programem do obsługi kart(wagi i kolor).

Na tym programie nie użyto żadnej biblioteki.

Ten program zawiera konstruktor klasy Card i jej inne metody: numRet, colorRet, toString, clone.

2.2 Pakiet Uno.game

2.2.1 GameManagement.java

GameManagement.java jest programem, który inicjuje okno aplikacji które jest w Uno.window.AppWindow.

Biblioteki użyte w programie GameManagement.java:

- Java.awt.Font;
- Java.awt.FonrMetrics;
- Java.awt.Graphics;
- Java.awt.image.BufferStrategy;
- Java.awt.image.BufferedImage;

Ten program jest połączony z innymi programami:

- Uno.game.hendler.GameHandlerer;
- Uno.game.input.*;
- Uno.window.*;
- Uno.window.screens.*;

Ten program zawiera klasę która odpowiada za przechowywanie rzeczy które są potrzebne do naszej aplikacji takich jak: listener do klawiatury,myszki,okna menu,connection itp.

2.3 Uno.game.handler

2.3.1 GameHadlerer.java

GameHandlerer jest programem, który zawiera klasę ktorą używa jako medium komunikacji miedzy inymi klasami(np. Aby zdobyć zmienna od jednej klasy i wykorzystać jej w drugiej).

Biblioteki użyte w programie GameHandlerer:

- Java.awt.Font;
- Java.awt.FontMetrics;
- Java.awt.image.BufferedImage;

Ten program jest połączony z innimy programami:

- Uno.Card;
- Uno.Deal;
- Uno.game.GameManagement;
- Uno.game.input.KeyboardMangaer;
- Uno.game.input.MouseManager;
- Uno.net.client.GameClient;
- Uno.net.server.GameServer;
- Uno.window.screens.ConnectionScreen;
- Uno.window.screens.Screens;
- Uno.window.screens.Screens;

- Uno.window.ui.manager.UIManagerS;
- Uno.window.CardLoader;
- Uno.window.ImagesLoader;

2.4 Uno.game.input

2.4.1 KeyboardMangaer.java

KeyboardManagaer.java jest programem, który nasłuchuje zdarzeń z klawiatury.

Biblioteki użyte w programie KeyboardManagaer.java:

- Java.awt.event.KeyEvent;
- Java.awt.event.KeyListener;

Ten program jest połączony z innymi programami:

- Uno.window.ui.manager.UIManagerS;

2.4.2 MouseManager.java

MouseManager.java jest programem, który nasłuchuje zdarzeń z myszy.

Biblioteki użyte w programie KeyboardManagaer.java:

- Java.awt.event.MouseEvent;
- Java.awt.event.MouseListener;
- Java.awt.event.MouseMotionListener;

Ten program jest połączony z innymi programami:

- Uno.window.ui.manager.UIManagerS;

2.5 Uno.net.client

2.5.1 GameClient.java

GameClient.java jest programem, za pomocą którego klient otrzymuje informacje z serwera.

Biblioteki użyte w programie GameClient.java:

- java.io.BufferedReader;
- java.io.IOException;
- java.io.InputStreamReader;
- java.io.PrintWriter;

- java.net.Socket;
- java.net.UnknownHostException;

Ten program jest połączony z innymi programami:

- Uno.Card;
- Uno.Deal;
- Uno.game.handler.GameHandlerer;
- Uno.net.packets.Packet;
- Uno.net.packets.Packet.PacketsIDs;
- Uno.net.packets.PacketDisconnect;
- Uno.net.packets.PacketGetCard;
- Uno.net.packets.PacketLogin;
- Uno.net.packets.PacketSetCard;
- Uno.net.packets.PacketSizeP;
- Uno.net.packets.PacketUpdate;

2.6 Uno.net.packets

Wszystkie programy w tym pakiecie są potrzebne do wysyłania informacji do serwera/klienta, aby coś zrobić, pierwszy charakter z wiadomości mówi jaki to pakiet.

2.7 Uno.net.server

2.7.1 ClientHandlerer.java

ClientHandlerer.java jest programem(server), który jest połaczony z klientem, wysyła do niego komunikaty.

Biblioteki użyte w programie ClientHandlerer.java:

- java.io.BufferedReader;
- java.io.IOException;
- java.io.InputStreamReader;
- java.io.PrintWriter;
- java.net.Socket;

Ten program jest połączony z innymi programami:

- Uno.Card;
- Uno.Deal;
- Uno.net.packets.Packet;
- Uno.net.packets.PacketLogin;
- Uno.net.packets.Packet.PacketsIDs;

2.7.2 GameServer.java

GameServer.java jest programem, w którym klient dostaje inforamcje od serwera.

Biblioteki użyte w programie ClientHandlerer.java:

- java.io.IOException;
- java.net.ServerSocket;
- java.util.ArrayList;
- java.util.Random;

Ten program jest połączony z innymi programami:

- Uno.Card;
- Uno.CardHandler;
- Uno.net.packets.PacketDisconnect;
- Uno.net.packets.PacketGetCard;
- Uno.net.packets.PacketSetCard;
- Uno.net.packets.PacketSizeP;
- Uno.net.packets.PacketStartGame;
- Uno.net.packets.PacketTurn;
- Uno.net.packets.PacketUpdate;

2.8 Uno.window

2.8.1 AppWindow

AppWindow jest programem, który inicjuje działanie okna aplikacji.

Biblioteki użyte w programie AppWindow.java:

- Java.awt.Canvas;
- Java.awt.Dimension;
- Java.awt.event.WindowEvent;
- Javax.swing.JFrame;

Ten program zawiera konstruktor klasy AppWindow i inne metody: canvasRet, windowRet, resizeWindow, exitWindow.

2.8.2 CardLoader.java

CardLoader.java jest programem, potrzebne do załadowania, pokolorowania, obrócenia obrazów kart.

Biblioteki użyte w programie CardLoader.java:

- Java.awt.Color;
- Java.awt.Graphics;
- Java.awt.Graphics2D;
- Java.awt.imge.BufferedImage;

Ten program jest połączony z innymi programami:

- Uno.Card;
- Uno.game.handler.GameHandlerer;

Ten program zawiera klas CardLoader i metody: loadRotatedCards, loadCards, paintOver, RotateNCards.

2.8.3 ImagesLoader.java

ImagesLoader.java jest programem, który ładuje zdjecia do aplikacji.

Biblioteki użyte w programie ImagesLoadre.java:

- Javax.imageio.ImageIO;
- Java.awt.image.BufferedImage;
- Java.io.IOException;

Ten program zawiera klas ImagesLoader i metody: loadImage, cropOutImage.

2.8.4 SheetHolder.java

SheetHolder.java jest programem, który trzyma sheests i zwraca je.

Biblioteki użyte w programie SheetHolder.java:

- Java.awt.image.BufferedImage;

Ten program zawiera klas SheetHolder i metody: getCardSheet, setCardSheet, getButtonSheet, setButtonSheet.

2.8.4 WindowHandlerer.java

WindowHandlerer.java jest programem, który odpowiedzialny za wyłączenie serwera i klienta(jeśli są włączone), gdy okna programu są zamknięte.

Biblioteki użyte w programie SheetHolder.java:

- Java.awt.event.WindowEvent;
- Java.awt.event.WindowListener;

Ten program jest połączony z innymi programami:

- Uno.game.handler.GameHadlerer;

2.9 Uno.window.screens

2.9.1 ConnectionScreen.java

ConnectionScreen.java jest programem, który służące do połaczenia lub hostowania gry.

Biblioteki użyte w programie ConnectionScreen.java:

- java.awt.Color;
- java.awt.Font;
- java.awt.Graphics;
- java.awt.Window;
- java.awt.image.BufferedImage;
- javax.swing.JLabel;
- javax.swing.JOptionPane;
- javax.swing.SwingUtilities;

Ten program jest połączony z innymi programami:

- Uno.game.handler.GameHandlerer;
- Uno.window.ui.UIButton;
- Uno.window.ui.UIButtonImage;
- Uno.window.ui.UIClicker;
- Uno.window.ui.UITextEnter;
- Uno.window.ui.manager.UIManagerS;

Ten program zawiera class ConnectionScreen który dziedziczy Screens, konstruktor klasy ConnectionScreen, metody: addComponents, update, render, create_Label, create_Dialog, disposeOptionDialog, startGameForClient.

2.9.2 GameScreen.java

GameScreen.java jest programem, który inicjuje okno w którym rozgrywa się gra.

Biblioteki użyte w programie GameScreen.java:

- java.awt.Color;
- java.awt.Graphics;
- java.awt.image.BufferedImage;
- java.util.ArrayList;
- javax.swing.JOptionPane;

Ten program jest połączony z innymi programami:

- Uno.Card;
- Uno.Deal;
- Uno.CardHandler;
- Uno.game.handler.GameHandlerer;
- Uno.window.ui.UIButtonImage;
- Uno.window.ui.UICard;
- Uno.window.ui.UIClicker;
- Uno.window.ui.UIHolder;
- Uno.window.ui.UITakeCard;
- Uno.window.ui.manager.UIManagerS;

Ten program zawiera class GameScreen który dzidziczy Screens, konstruktor klasy GameScreen, metody: addComponent, update, render, drawPacketCards, exchangeCenter, addCardtoUI, rearangeCards, drawOnTop, commitAction, UpdateDeck, checkIfFinished, assignNames.

2.9.3 MenuScreen.java

MenuScreen.java jest programem, który inicjuje okno „Menu”.

Biblioteki użyte w programie MenuScreen.java:

- java.awt.Color;
- java.awt.Font;
- java.awt.Graphics;
- java.awt.image.BufferedImage;

Ten program jest połączony z innymi programami:

- Uno.game.handler.GameHandlerer;
- Uno.window.ui.UIButtonImage;
- Uno.window.ui.UIClicker;
- Uno.window.ui.manager.UIManagerS;

Ten program klas MenuScreen który dzidziczy Screens, konstruktor klasy MenuScreen, metody: addComponent, update, render.

2.9.4 Screens.java

Screens.java jest programem, który inicjuje abstrakcyjne klasa po której wszystkie screeny dziedziczą, definiuje rzeczy które są potrzebne dla każdego z nich(zmienne,funkcje).

Biblioteki użyte w programie Screens.java:

- Java.awt.Graphics;

Ten program jest połączony z innymi programami:

- Uno.game.handler.GameHandlerer;
- Uno.window.ui.manager.UIManagerS;

2.10 Uno.window.ui

Wszystkie programy w tym pakiecie są odpowiada za user interface programu.

2.11 Uno.window.ui.manager

2.11.1 UIManagerS.java

UIManagerS.java jest programem, który robi wszystkie rzeczy dla obiektów z uiLista (każdy screen ma innego uiLista) kiedy jakieś wydarzenie się pojawi (renderuje, updatuje, kiedy jest ruch myszy lub klawiatury itp.).

Biblioteki użyte w programie UIManagerS.java:

- java.awt.Graphics;
- java.awt.event.KeyEvent;
- java.awt.event.MouseEvent;
- java.util.ArrayList;

Ten program jest połączony z innymi programami:

- Uno.game.handler.GameHandlerer;
- Uno.window.ui.UiComponent;

3 Server

Aby więc było spotkanie co najmniej dwóch obiektów na terytorium jednego serweru - serwer musi zająć określony port, a drugi musi znaleźć miejsce spotkania znając adres IP.

Port - to unikalny numer, z którym powiązane jest określone socker, innymi słowami, jest zajęty przez określoną usługę, aby móc się z nią skontaktować.

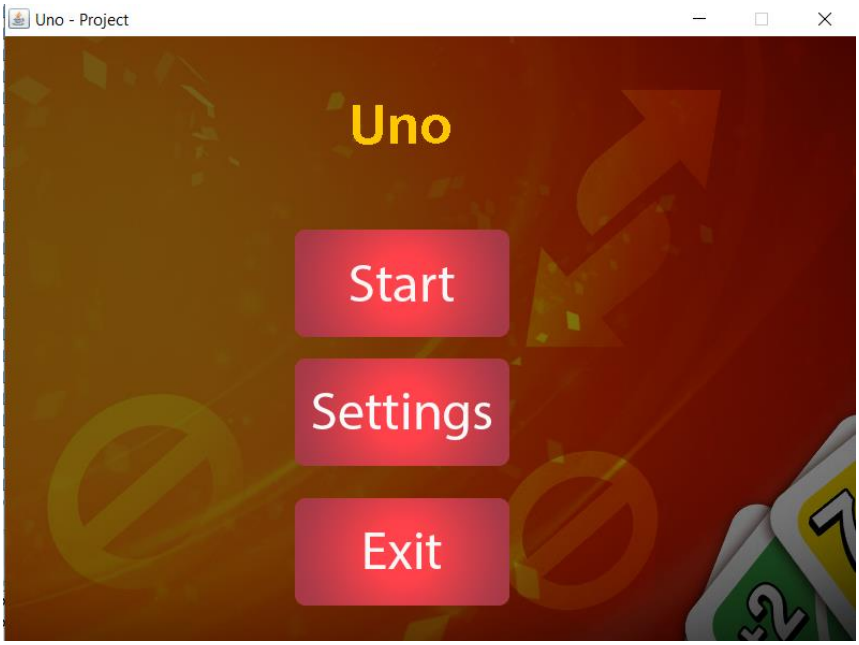
clientSocket – implementuje ideę gniazda. Poprzez kanały wejściowe/wyjściowe klient komunikuje się z serwerem.

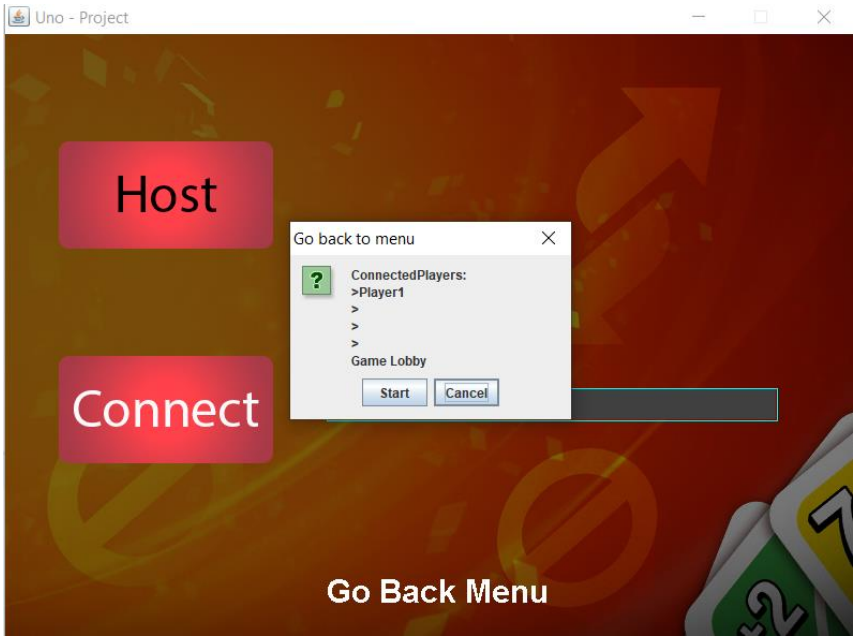
ClientSocket jest zadklarowany po stronie klienta, a serwer odtwarza ją, odbierając sygnał połączenia. Tak odbywa się komunikacja sieciowa.

serverSocket – to to samo co clientSocket, tylko dla serwera. Ale odgrywa zupełnie inną rolę niż clientSocket.

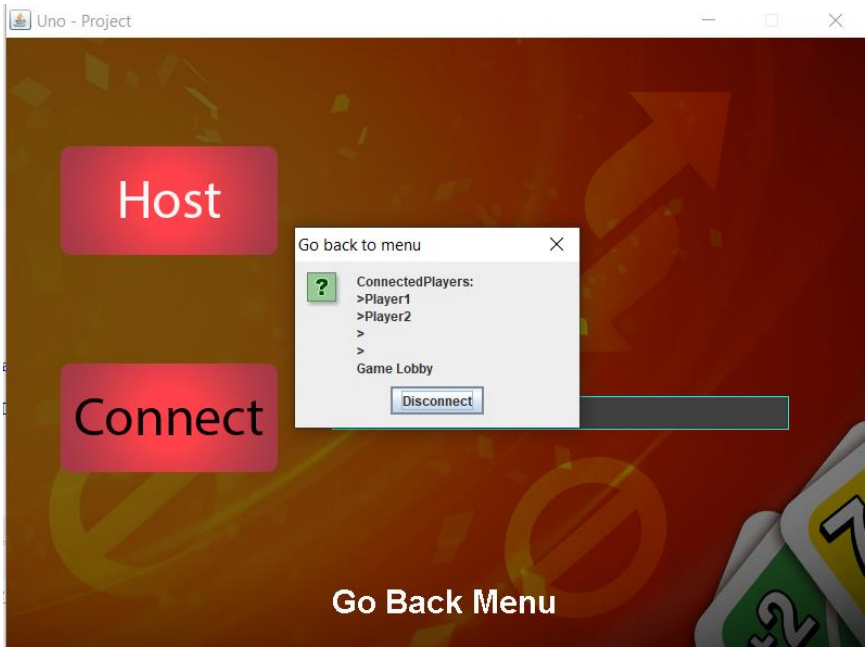
Używamy TCP.

4 Interfejs gry

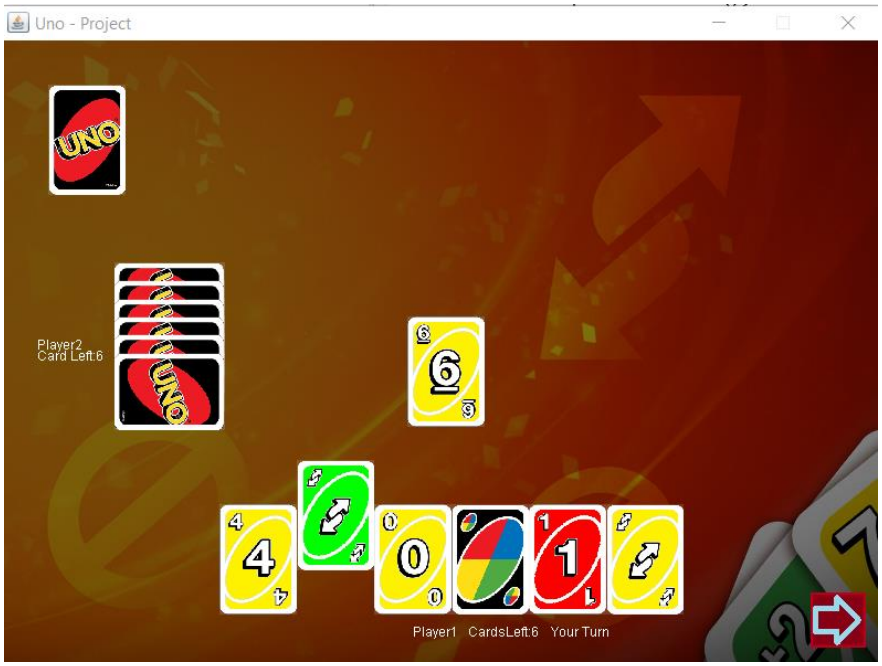




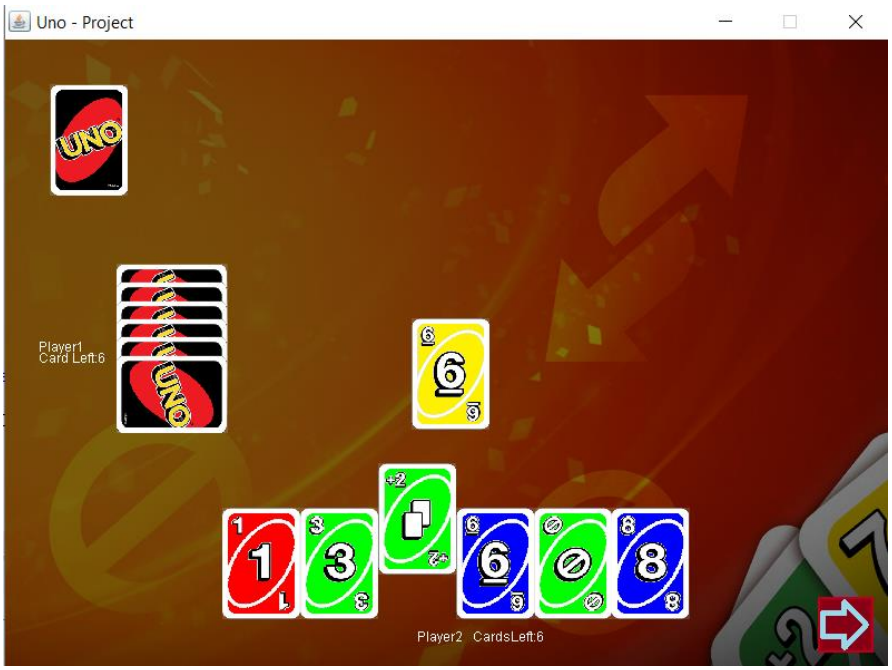
Serwer



Klient



Talia kart serwera



Talia kart klienta

Uwaga! Z gry należy kolejno wyjść najpierw klienci, a następnie serwer.

5 Wniosek

Wszystkie postawione cele zostały wykonane. Zrealizowałyśmy projekt wykorzystujący architekturę klient-serwer napisany w języku Java. Nam zajeło dużo czasu żeby zrozumić jak realizować prawidlowo server i zrobić to w programie. Dla

wykonania potrzebnym było duże rzeczy znaleźć i nauczyć się temu. Ale na końcu zdążyłyśmy realizować wszysko co było potrzebne.

Wspólnymi siłami daliśmy radę stworzyć aplikacje, z której możemy być dumni.