Politechnika Świętokrzyska w Kielcach
Wydział Elektrotechniki, Automatyki i Informatyki
Katedra Informatyki, Elektroniki i Elektrotechniki

Kierunek: Informatyka	Przedmiot **Programowanie Obiektowe Java**	
Grupa laboratoryjna: 2ID11A	Temat projektu: Gra karciana UNO	Wykonali: **Wiktor Chabera** **Mikołaj Widanka**

1. Wstęp

- **Gra UNO zasady**

Zwykle 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)

Kart funkcyjnych:

- 8 kart stopu (Postój) po dwie z każdego koloru
- 8 kart zmiany kierunku (Zmiana kierunku) po dwie z każdego koloru
- 8 kart +2 (Weź dwie) po dwie z każdego koloru
- 4 czarne karty +4 ze zmianą koloru (Wybierz kolor + Weź cztery)
- 4 czarne karty zmiana koloru (Wybierz kolor)

Na początku gry rozdaje się po 7 kart każdemu graczowi i jedną z talii kładzie się na środek. Grę rozpoczyna osoba po lewej stronie rozdającego. 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.

W talii są także karty specjalne takie jak Postój, Zmiana kierunku, Weź dwie, Wybierz kolor + Weź cztery, Wybierz kolor.

- Postój (jak 4 w makao) – następny gracz traci (stoi) kolejkę
- Zmiana kierunku (jak As w kunjo) – karta zmieniająca kierunek gry
- Weź dwie – następny gracz bierze dwie karty
- Wybierz kolor + Weź cztery – zagrywający kartę deklaruje zmianę koloru na dowolnie przez siebie wybrany, następny gracz dobiera 4 karty.
- Wybierz kolor – zagrywający kartę deklaruje zmianę koloru na dowolnie przez siebie wybrany (jeden z kolorów dostępnych w grze)

Karty Postój, Zmiana kierunku, Weź dwie można kłaść do danego karcie koloru – natomiast karty Wybierz kolor + Weź cztery można kłaść na dowolną kartę.

Wygrywa ten, kto pierwszy pozbędzie się wszystkich kart. Celem gry jest pozbycie się wszystkich kart ze swojej talii.

- Technologie użyte w projekcie

Podczas tworzenia projektu oprócz standardowych bibliotek języka Java korzystaliśmy z biblioteki JavaFX do tworzenia grafiki do naszej gry. Wbudowany w nie język FXML ułatwił zarządzanie komponentami, okna i powiązanie ich z napisanym na ich potrzeby kodem. Podczas tworzenia projektu korzystaliśmy również z zewnętrznego edytora interfejsów graficznych jakim jest SceneBuilder. Jest to proste lecz przydatne narzędzie pozwalające projektować proste warstwy graficzne do aplikacji okienkowych. Edytor ten posłużył nam m.in. do zorganizowania i ustalenia pozycji elementów na ekranie i przygotowania ich do pełnienia odpowiednich funkcji zaprogramowanych w kodzie (np. karty jako przyciski, czy też talia przewijana sliderem u dołu ekranu).

2. Opis rozgrywki i funkcji projektu

Rozgrywka w naszej grze oparta jest na oryginalnym wariancie gry UNO. Na początku gry każdy z uczestników dobiera po 7 kart, a zmiana gracza odbywa się domyślnie zgodnie z ruchem wskazówek zegara. W głównym oknie gry znajduje się kilka istotnych komponentów takich jak nazwy poszczególnych graczy i ilości posiadanych przez nich kart, przyciski do pominięcia tury, dobrania karty, czy też powiedzenia „UNO” kiedy to gracz pozbędzie się wszystkich kart. Wartym wspomnienia mechanizmem który zaprojektowaliśmy jest dobieranie kart do momentu kiedy zostanie dobrana karta możliwa do położenia na stosie. Jest to bardzo wygodnie rozwiązanie gdyż w pewnym momencie kart jest zbyt wiele, dlatego przewijanie i sprawdzanie ich za każdym razem byłoby uciążliwe i spowalniałoby rozgrywkę. W lewym górnym rogu okna znajduje się informacja o graczach biorących udział w partii, a także ilości posiadanych przez nich kart, natomiast w/w przyciski funkcyjne znajdują się poniżej. Najistotniejszy element jakim jest talia kart znajduje się na dole ekranu. Talie można przeglądać w całości używając paska przewijania tzw. „slidera”. Zdecydowaliśmy się na takie rozwiązanie gdyż okno jest zbyt małe aby wyświetlić większą ilość kart, a pomniejszanie ich lub powiększanie okna mogłoby zaburzyć efekt wizualny i porządek na ekranie. Oprócz

okna rozgrywki istnieją jeszcze okna: powitalne z funkcją uruchomienia gry lub przejrzenia rankingów (element architektury klient-serwer), oraz okno wyboru graczy gdzie wpisując lub wybierając odpowiednie nazwy możemy dowolnie ustalać imiona i liczbę graczy lub ich usuwać.

3. Opisy metod i klas

```
public enum Color {} - enumerate zawierający właściwości dotyczące koloru kart

public enum Value {} - enumerate zawierający właściwości dotyczące wartości kart

public class UnoCard {} - obiekt karty zawiera metody dostępu do swoich właściwości oraz pozwalające na
                          porównanie

public class UnoDeck {} - obiekt będący talią

        public void fillDeck() {} - metoda wypełniający talię kartami

        public UnoCard drawCard() {} - metoda pozwalająca na dobranie karty z talii

public class UnoGame {} - obiekt zawierający stan oraz reguły rozgrywki

        public UnoGame(int numOfPlayers, ArrayList<String> names){} - konstruktor tworzący nową rozgrywkę

        public void drawCard() {} - wrapper funkcji drawCard z klasy UnoDeck pozwalający na dobranie karty
                                    przez obecnego gracza

        public void updateValid() {} - metoda aktualizują obecnie prawidłowe wartości karty

        public void playCard(UnoCard playedCard) {} - metoda pozwalająca na  zagranie karty

        public boolean cardValidation(UnoCard card) {} - metoda sprawdzają

        public void changePlayer() {} - metoda zmieniająca obecnego gracza zgodnie z kierunkiem gry

        public void performSkip() {} - metoda wykonująca kartę specjalną skip

        public void performDrawTwo() {} - metoda wykonująca kartę drawTwo

        public void performReverse() {} - metoda wykonująca kartę zmieniającą kierunek gry

        public void performColorChange() {} - metoda zmieniająca kolor obecnej karty

        public void performWild() {} - metoda zmienia kolor oraz gracza (zgodnie z kierunkiem rozgrywki)
                                       następny gracz dobiera cztery karty

        public Color getColorChoice() {} - metoda wywołująca dialog pozwalający na wybór koloru

        public void incrementCounter(){} - inkrementująca licznik wykorzystywany w rankingu

public class UnoPlayer{} - klasa zawierająca metody oraz właściwości gracza

        public void giveCard(UnoCard card) {} - metoda dodająca graczowi nową kartę

        public boolean removeCard(UnoCard toRM) – metoda usuwająca kartę gracza
```

public class HallController{} - klasa będąca kontrolerem okna rankingowego

protected void setup(){} - metoda przygotowująca ranking

protected void handleCancel(ActionEvent event){} - przejście do menu głównego

public class Main extends Application {} - klasa inicializująca aplikację

public class MainMenuController {} - kontroler menu głównego

protected void handleHotSeatOnAction(ActionEvent event) {} - przejście do menu inicjalizującego rozgrywkę

protected void handleExit(){} - handler pozwalający na wyjście z programu

protected void handleHall(ActionEvent event) {} - handler przejści do rankingu

public class PlayerLabel extends TextField{} - element interfejsu wyświetlający nazwę gracza

public class ScoreLabel extends TextField{} - element interfejsu wyświetlający wynki i nazwę graczach

public class UnoCardButton extends Button{} - element interfejsu wyświetlający kartę (przycisk)

public static void loadAssets(){} - metoda wczytująca pliki graficzne będące kartami

public class UnoGameController{} - kontroler rozgrywki

protected void setupGame(ArrayList<String> players){} - metoda inicjalizująca nową rozgrywkę w kontekście okna

protected void refreshScreen(){} - metoda pozwalająca na odświeżenie okna rozgrywki

protected void playCard(ActionEvent curEvent){} - handler pozwalający na zagranie karty zawartej w klasie UnoCardButton

protected void clickDraw(){} - handler dobierający karty dla obecnego gracz dopóki dobranej karty nie będzie można zagrać

protected void handlePassTurn(){} - handler zmieniający gracza

protected void handleUno(){} - handler kończący rozgrywkę (dane zwycięzcy zostają wysłane do serwera)

public class UnoHotSeatSetupController{} - kontroler okna inicjalizacji rozgrywki

protected void handleCancel(ActionEvent event) {} - handler powrotu do głównego menu

protected void handleAddPlayer(){} - handler dodania nowego gracza do rozgrywki

protected void handleRemovePlayer(){} - handler usunięcia gracza z rozgrywki

protected void handleStartGame(ActionEvent event){} - handler rozpoczęcia gry

public class Klient{} - klasa zawierająca metody komunikacji z serwerem

public void establishConnection(String ip, int port){} - metoda nawiązująca połączenie z serwerem

public void sendMessege(UserScore msg){} - metoda wysyłająca pakiet do serwera

```
        public void requestRanking(){} - metoda wysyłająca zaoytanie o obecny ranking
public class ScoreSort{} - klasa pozwalająca na sortowanie listy rankingowej graczy
public class UserScore{} - klasa zawierająca dane odnośnie rankingu graczy
        public String getCSV(){} - metoda zwracająca ciąg znaków w konwencji CSV
public class Server{} - klasa zawierająca metody oraz dane serwera rankingowego
        public void loadData(){} - metoda wczytująca dane z pliku w formacie CSV
        public void saveData(UserScore savedData){} - metoda zapisująca dane do pliku
        public void start(int port){} - metoda inicjalizująca strumienie oraz Sockety nasłuchująca na określonym
                                        porcie
        public void action(){} - metoda określająca decyzje serwera
        public void stop(){} - metoda zamykająca strumienie oraz Sockety
```

4. Mechanizm klient-serwer

Nasz mechanizm oparty na architekturze klient serwer działa w bardzo prosty i intuicyjny sposób. Po każdej zakończonej rozgrywce informacja o wyniku jest wysyłana od klienta do serwera, który ją aktualizuje zapisując otrzymane informacje w „rankingach”. Dodatkowo dane na bieżąco po aktualizacji zapisywane są do pliku csv, który służy jako lokalna baza danych, gdzie widnieją nazwy graczy wraz z uzyskanymi przez nich wynikami. Rankingi możliwe są do przejrzenia z poziomu aplikacji poprzez naciśnięcie przycisku „Hall Of Fame” lub z poziomu w/w pliku csv.

5.Wnioski

Podczas tworzenia projektu zdobywaliśmy wiele przydatnych informacji z zakresu języka Java i jego możliwości, oraz poszerzaliśmy swoją wiedzę w dziedzinie programowania obiektowego (którą rozpoczęliśmy m.in. od tworzenia aplikacji języka C++). Nauczyliśmy się również korzystać z ogólnodostępnych narzędzi do edycji i tworzenia grafiki dla języka Java takich jak JavaFX, czy SceneBuilder. Nauczyliśmy się również obsługi bardzo istotnych mechanizmów takich jak klient serwer. Podczas prac nad aplikacją wykorzystywaliśmy wiedzę zdobytą na laboratoriach i wykładach, a także rozszerzaliśmy ją sięgając np. do różnych bibliotek Javy. Pracując przy projekcie opanowaliśmy również korzystanie z repozytoriów Git. Wiedza ta pozwoliła zbudować fundamenty pod kolejne projekty, a także znacznie poszerzyć nasze umiejętności z zakresu programowania i tworzenia aplikacji czy elementów sieciowych.