Laboratorium - Programowanie komputerów

Sprawozdanie z projektu pt.

Rummikub gra

Emanuel Jureczko gr. 5

Idea projektu

Program jest cyfrową kopią planszówki Rummikub. Gracze mogą przeżyć rozgrywkę poprzez wiersz poleceń. Rozgrywka odbywa się w większości zgodnie z standardowymi zasadami oryginalnej wersji. Tabliczki graczy nie są przed resztą graczy ukryte z racji tego, że rozgrywka odbywa się na jednym ekranie, więc każdy gracz zna żetony pozostałych graczy. Dla jednego będzie to mankament, dla drugiego zaś odskocznia od standardu, co wprowadzi urozmaicenie rozgrywki.

Projekt został napisany w celach dydaktycznych. Z tego powodu kod musiał spełniać określone wymagania, które miały na celu sprawdzenie umiejętności i wiedzy przyszłego programisty, dlatego niektóre fragmenty kodu mogą wydawać się zbędne w rzeczywistej implementacji. Dodatkowo autor zastrzega, że projekt służył jako sposób nauki i sam jest gotów przyznać, że kod posiada niedoskonałości oraz nieoptymalne rozwiązania.

Interfejs

Przebieg

Sterowanie w grze polega na wprowadzeniu odpowiednich komend do wiersza poleceń. Po włączeniu programu graczom ukazuje się menu główne.

```
Witaj w grze EmaKu... znaczy Rummikub!
Dostepne opcje:
c - sterowanie/controls
r - zasady/rules
p - graj/play
h - historia/history
e - wyjscie/exit
```

menu główne

Przykłady wyboru danej opcji

```
h


~~~HISTORIA ZWYCIEZCOW~~~

1~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. Gracz: Ema
1. Gracz: Kasia
3. Gracz: Marek

2~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. Gracz: Ema
2. Gracz: Kasia
3. Gracz: Marek

3~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1. Gracz: Ema
1. Gracz: Kasia
1. Gracz: Marek
```

```
r

Zasady dostepne na: https://rummikub.pl/instrukcje/
```

```
e - wyjscie/exit
c

sterowanie
Komenda wylozenia: <nr ciagu>-<idetyfikator zetonu> np.  3-01B  |  2-11G  |  11-J  |  9-JK  |  11-2R
Komenda wylozenia z JK: np. <2-JK> i potem watosc jaka ma przyjac np. 11R.
Komenda nowy - new: <n>-<idetyfikator zetonu> np.  n-1G  |  n-11B
Komenda wez - take: <t><nr ciagu>-<nr zetonu (pierwszy, drugi)> np.  -(1)-[2R][JK][4R]  t1-2  = bierzemy JK
Komenda dziel ciag - divide: <d><nr ciagu>-<nr zetonu po ktorym dzielimy> np.  -(1)-[2R][JK][4R][5R][6R] d1-3 = -(1)-[2R
][JK][4R] -(2)-[5R][6R]
Komenda poloncz ciagi - merge: <m><nr ciagu>-<nr ciagu z ktorym lonczymy> np.  -(1)-[2B][JK][4B] -(2)-[5B][6B][7B] m1-2
= -(1)-[2B][JK][4B][5B][6B][7B]
Mozna laczyc komendy za pomoca ; np. n-11g;3-11y;3-11r;d4-1 lub 1-jk;1-jk;1-jk
```

```
e - wyjscie/exit
p
Wybierz ilosc graczy z przedzialu 2-4
3
Wpisz nazwe pierwszego gracza. Mozesz wpisac litere <n> jesli nie chcesz nadawac znaw graczom.
Ema
Wpisz nazwe graczar nr. 2
Kasia
Wpisz nazwe graczar nr. 3
Marek

sterowanie
Komenda wylozenia: <nr ciagu>-<idetyfikator zetonu> np.  3-01B  |  2-11G  |  11-J  |  9-JK  |  11-2R
Komenda wylozenia z JK: np. <2-JK> i potem watosc jaka ma przyjac np. 11R.
Komenda nowy - new: <n>-<idetyfikator zetonu> np.  n-1G  |  n-11B
Komenda wez - take: <t><nr ciagu>-<nr zetonu (pierwszy, drugi)> np.  -(1)-[2R][JK][4R]  t1-2  = bierzemy JK
Komenda dziel ciag - divide: <d><nr ciagu>-<nr zetonu po ktorym dzielimy> np.  -(1)-[2R][JK][4R][5R][6R] d1-3 = -(1)-[2R
][JK][4R] -(2)-[5R][6R]
Komenda poloncz ciagi - merge: <m><nr ciagu>-<nr ciagu z ktorym lonczymy> np.  -(1)-[2B][JK][4B] -(2)-[5B][6B][7B] m1-2
= -(1)-[2B][JK][4B][5B][6B][7B]
Mozna laczyc komendy za pomoca ; np. n-11g;3-11y;3-11r;d4-1 lub 1-jk;1-jk;1-jk

Pierwsze wylozenie musi miec wspolna wartosc zetonow >=30


~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-
~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1B] [1Y] [2R] [3G] [3B] [4B] [5R] [6R] [6B] [6R] [8G] [10B] [11B] [12Y] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ema twoj ruch.
Wpisz komende ruchu:
```

Przykład sekwencji komend dla rozpoczęcia rozgrywki

Sterowanie

W każdym momencie rozgrywki gracze mogą się poddać za pomocą komendy „surr”. Wszyscy gracze, którzy się poddali, zostaną umieszczeni na ostatnim miejscu niezależnie od tego, w której turze się poddali.
Aby zakończyć turę należy wpisać „f”.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1B] [1Y] [2R] [3G] [3B] [4B] [5R] [6R] [6B] [6R] [8G] [10B] [11B] [12Y] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ema twoj ruch.
Wpisz komende ruchu:
f
Wylozenie jest niepoprawne! Ciagniesz zeton z worka! -->[8Y]
Pierwsze wylozenie musi miec wspolna wartosc zetonow >=30
```

Tura zakończona niepowodzeniem

Wyłóż

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Marek:
{ [2R] [3Y] [3R] [4Y] [5G] [6B] [7B] [8G] [9Y] [10Y] [12G] [13B] [13R] [13G] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Marek twoj ruch.
Wpisz komende ruchu:
1-13B;1-13G;1-13r



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[13B][13G][13R]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Marek:
{ [2R] [3Y] [3R] [4Y] [5G] [6B] [7B] [8G] [9Y] [10Y] [12G] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Marek twoj ruch.
Wpisz komende ruchu:
f
~~Poprawne pierwsze wylozenie~~
```

Tura zakończona powodzeniem (dla pierwszego wyłożenia wartość żetonów musi być większa bąć równa 30)

Wyłóż JK

Należy wpisać dokładną wartość, jaką ma przyjąć Joker. Program nie domyśla się, czym ma być Joker, aby ciąg był poprawny.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
-(3)-[9Y][10Y]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1B] [1Y] [1G] [2R] [2B] [3G] [3B] [4B] [5R] [5Y] [6R] [6B] [6R] [8G] [8Y] [10B] [11B] [12Y] [13G] }
Kasia:
{ [1R] [1G] [2Y] [2G] [3Y] [5R] [6Y] [7G] [8B] [9G] [9R] [9R] [9Y] [9G] [10R] [11B] [11G] [12B] [13Y] }
Marek:
{ [2R] [3R] [5G] [6B] [7B] [8G] [12G] [JK] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Marek twoj ruch.
Wpisz komende ruchu:
3-jk
Jaka wartosc chcesz nadac zetonowi? Wzor komendy --> 2B --> 11G
11y


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
-(3)-[9Y][10Y][JK11Y]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1B] [1Y] [1G] [2R] [2B] [3G] [3B] [4B] [5R] [5Y] [6R] [6B] [6R] [8G] [8Y] [10B] [11B] [12Y] [13G] }
Kasia:
{ [1R] [1G] [2Y] [2G] [3Y] [5R] [6Y] [7G] [8B] [9G] [9R] [9R] [9Y] [9G] [10R] [11B] [11G] [12B] [13Y] }
Marek:
{ [2R] [3R] [5G] [6B] [7B] [8G] [12G] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Marek twoj ruch.
Wpisz komende ruchu:
```

Poprawne zastosowanie Jokera

Nowy

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1B] [1Y] [1G] [2R] [2B] [3G] [3B] [4B] [5R] [5Y] [6R] [6B] [6R] [8G] [8Y] [10B] [11B] [12Y] [13G] }
Kasia:
{ [1R] [1G] [2Y] [2G] [3Y] [5R] [6Y] [7G] [8B] [9G] [9R] [9R] [9Y] [9G] [10R] [11B] [11G] [12B] [13Y] }
Marek:
{ [2R] [3R] [5G] [6B] [7B] [8G] [9Y] [10Y] [12G] [JK] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Marek twoj ruch.
Wpisz komende ruchu:
n-9y;3-10y


~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
-(3)-[9Y][10Y]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1B] [1Y] [1G] [2R] [2B] [3G] [3B] [4B] [5R] [5Y] [6R] [6B] [6R] [8G] [8Y] [10B] [11B] [12Y] [13G] }
Kasia:
{ [1R] [1G] [2Y] [2G] [3Y] [5R] [6Y] [7G] [8B] [9G] [9R] [9R] [9Y] [9G] [10R] [11B] [11G] [12B] [13Y] }
Marek:
{ [2R] [3R] [5G] [6B] [7B] [8G] [12G] [JK] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Marek twoj ruch.
Wpisz komende ruchu:
```

Weź

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
-(3)-[9Y][10Y][JK11Y]
-(4)-[10B][11B][12B]
-(5)-[1B][2B][3B][4B]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1Y] [1G] [2R] [3G] [5R] [5Y] [6R] [6B] [6R] [8G] [8Y] [12Y] [13G] }
Kasia:
{ [1R] [1G] [2Y] [2G] [3Y] [5R] [6Y] [7G] [7R] [8B] [9G] [9R] [9R] [9Y] [9G] [10R] [11B] [11G] [12B] [12Y] [13Y] }
Marek:
{ [2R] [3R] [4R] [5G] [6B] [7B] [8G] [11R] [12G] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ema twoj ruch.
Wpisz komende ruchu:
t3-3


~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
-(3)-[9Y][10Y]
-(4)-[10B][11B][12B]
-(5)-[1B][2B][3B][4B]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1Y] [1G] [2R] [3G] [5R] [5Y] [6R] [6B] [6R] [8G] [8Y] [12Y] [13G] } | { [JK] }
Kasia:
{ [1R] [1G] [2Y] [2G] [3Y] [5R] [6Y] [7G] [7R] [8B] [9G] [9R] [9R] [9Y] [9G] [10R] [11B] [11G] [12B] [12Y] [13Y] }
Marek:
{ [2R] [3R] [4R] [5G] [6B] [7B] [8G] [11R] [12G] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ema twoj ruch.
Wpisz komende ruchu:
```

Dziel

Nowy ciąg powstały przez podział zawsze zostaje ostatnim ciągiem.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
-(3)-[8Y][9Y][10Y]
-(4)-[10B][11B][12B][13B]
-(5)-[1B][2B][3B][4B][JK5B][6B][7B]
-(6)-[2R][3R][4R][5R][6R]
-(7)-[9R][9G][9B]
-(8)-[9Y][9G][9R]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1Y] [1G] [2R] [3G] [5Y] [5B] [6B] [6R] [7Y] [8G] [12Y] [13G] }
Kasia:
{ [1R] [1G] [2Y] [2G] [3Y] [5R] [6Y] [7G] [7R] [8B] [10R] [11B] [11G] [11G] [12B] [12Y] [13Y] }
Marek:
{ [5G] [8G] [11R] [12G] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Ema twoj ruch.
Wpisz komende ruchu:
d5-4


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
-(3)-[8Y][9Y][10Y]
-(4)-[10B][11B][12B][13B]
-(5)-[1B][2B][3B][4B]
-(6)-[2R][3R][4R][5R][6R]
-(7)-[9R][9G][9B]
-(8)-[9Y][9G][9R]
-(9)-[JK5B][6B][7B]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Połącz

```
-(6)-[5R][6R][JK7R]
-(7)-[2R][3R][4R]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tabliczki graczy:
Ema:
{ [1Y] [1G] [2R] [3G] [5Y] [6B] [6R] [8G] [12Y] [13G] }
Kasia:
{ [1R] [1G] [2Y] [2G] [3Y] [5R] [6Y] [7G] [7R] [8B] [9G] [9R] [9R] [9Y] [9G] [10R] [11B] [11G] [11G] [12B] [12Y] [13Y] }

Marek:
{ [5G] [6B] [7B] [8G] [11R] [12G] }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Marek twoj ruch.
Wpisz komende ruchu:
m6-7



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-(1)-[3Y][4Y][5Y]
-(2)-[13B][13G][13R]
-(3)-[8Y][9Y][10Y]
-(4)-[10B][11B][12B]
-(5)-[1B][2B][3B][4B]
-(6)-[2R][3R][4R][5R][6R][JK7R]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Dodatkowe informacje

Przed każdym wypisaniem planszy następuje sortowanie ciągów oraz tabliczek graczy. Wyjątkiem jest sytuacja, gdy używamy komendy „weź”. Wtedy wzięty żeton, który poprzednio znajdował się w tabliczce gracza, zostanie ustawiony na końcu.
W przypadku wpisania niepoprawnej komendy, program poinformuje o tym gracza oraz zasugeruje skorzystanie z komendy „h”.

```
~~~TABELA ZWYCIEZCOW~~~

1. Gracz: Ema
2. Gracz: Marek
3. Gracz: Kasia


Witaj w grze EmaKu... znaczy Rummikub!
Dostepne opcje:
c - sterowanie/controls
r - zasady/rules
p - graj/play
h - historia/history
e - wyjscie/exit
```

Rezultat zakończonej rozgrywki.

Opis klas i struktury danych

Główna struktura danych

Na potrzeby wymagań projektu oraz treningu została stworzona własna lista jednokierunkowa o nazwie „ListaZetonow”*. Składa się ona z prywatnego pola ElementListy* oraz publicznych metod:

- void wyswietlListe();
- void dodajZeton(Zeton* zeton);
- void usunElement(Zeton* zeton);
- void polaczListy(const ListaZetonow& lista);
- ListaZetonow podzielListy(int indeks);
- Iterator begin() const;
- Iterator end() const;

Dodatkowo, klasa zawiera wewnętrzną klasę Iterator, która posiada prywatne pole „ElementListy*” oraz odpowiednie operatory:

- Iterator& operator++();
- bool operator!=(const Iterator& other) const;
- Zeton* operator*() const;

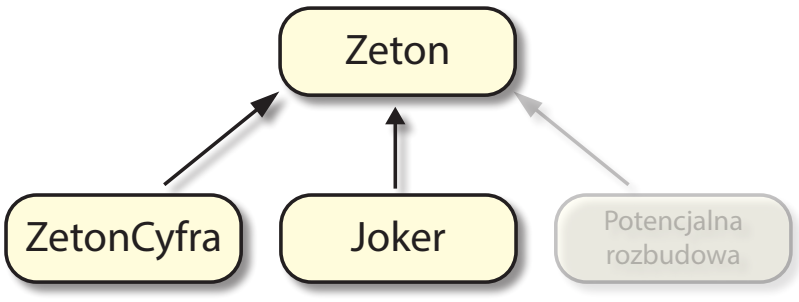
Klasy

Zeton

Klasa abstrakcyjna, zawierające metody i operatory potrzebne klasom dziedziczącym.
W tej wersji dziedziczyć będą tylko 2 klasy, ale taka implementacja pozwala na stosunkowo łatwą potencjalną rozbudowę planszówki np. do wersji Rummikub Twist 3W1, gdzie mamy więcej rodzajów jokerów.

ZetonCyfra

Przypadek szczególny żetonu, który reprezentuje jego domyślną postać czyli liczba i kolor. Klasa dziedziczy publicznie po klasie Zeton. Zawiera pola przechowujące int oraz char, metody i operatory potrzebne do wyświetlenia, porównanie czy uzyskania danych.



Joker

Przypadek szczególny żetonu, który może przyjąć postać dowolnego żetonu. Klasa dziedziczy publicznie po klasie Zeton. Zawiera dodatkowe pola tymczasowych danych o wartości i kolorze żetonu. Metody są podobne to tych z klasy ZetonCyfra, ale przystosowane do działania Jokera. Klasa zawiera również metodę do nadania tymczasowych wartości żetonu.

WorekNaZetony

Klasa zawiera pole vector<Zeton*> do którego wrzuca odpowiednią ilość danych wskaźników na żetony klas ZetonCyfra i Joker, potrzebny do rozgrywki. Żetony zostaną wymieszane, aby następnie mogły być wydane innym klasom.

CiagZetonow

Klasa zawierająca listę wskaźników na Zeton. Jest zaprzyjaźniona z klasą Plansza, aby ta mogła manipulować listą. Oprócz możliwości wyświetlenia zawartości posiada metodę sprawdzania poprawność ciągu, która jest wykorzystywane w końcowym sprawdzaniu ruchu.

**ListaZetonow została zaprojektowana wyłącznie do celów dydaktycznych. W ramach tego projektu zastosowano std::list.*

TabliczkaGracza
Klasa zawierająca jedną listę wskaźników na Zeton do przechowywania zawartości tabliczki oraz drugą dla podręcznych żetonów wyciągnięci z planszy. Pozostałe to numer gracza oraz wartość boolowska przechowująca informacją o konieczności wykonania pierwszego wyłożenia. Jest zaprzyjaźniona z klasą Plansza. Szczególna metoda tej klasy służy do sprawdzanie czy żeton wskazany komendą przez gracza jest w znajduje się w którejś z list.

OdczytZapisDoPliku
Klasa zarządzająca odczytem i zapisem do pliku. Aktualnie wykorzystywana do zapisywania i przekazywania danych potrzebnych do wyświetlenia historii zwycięzców.

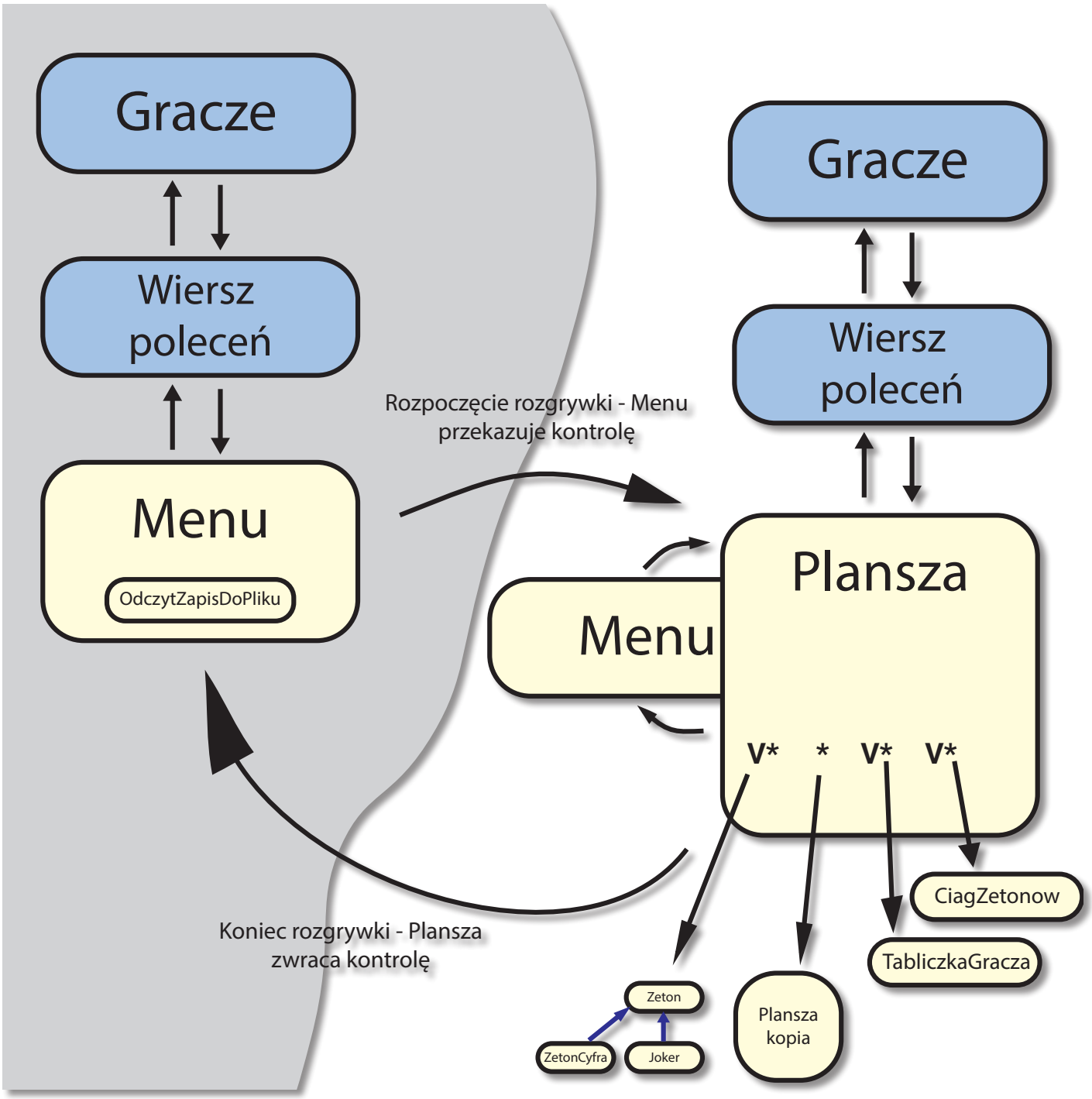
Menu
Klasa wyświetla powitalny interfejs oraz służy do rozpoczęcia jak i zakończenia rozgrywki. Instruuje gracza o sposobie sterowania oraz zasadach. Metody menu mogą zostać wywołane podczas rozgrywki w razie konieczność przypomnienia zasad lub sterowania za pomocą odpowiednich komend.

Plansza
Najbardziej rozbudowana oraz odpowiedzialna za przebieg klasa korzystająca lub zawierając właściwie wszystkie pozostałe klasy. Jej polami prywatnymi są:

- vector<CiagZetonow*> wszystkieCiagi - przechowujący wszystkie ciągi znajdujące się na planszy
- vector<TabliczkaGracza*> wszystkieTabGr - przechowujący tabliczki wszystkich graczy
- vector<Zeton*> resztaZetonowWWorku - reszta żetonów która pozostała w worku po wydaniu potrzebnych żetonów graczom. Są potrzebne w razie gdy dany gracz nie potrafi swoimi przydzielonymi żetonami wykonać żadnego ruchu
- string tabelaZwyciezcow - w którym będzie zapisywana kolejność graczy którzy pozbyli się zawartości ich tabliczek
- Plansza* kopiaZaposwaPlanszy - kopia która zostaje wczytana gdy dany gracz kończy turę, ale nie wykonał poprawnych ruchów. Poprawność została sprawdzona przez metodę klasy Plansza
- bool pierwszeWykonanieKomendy - potrzebne do poprawnego przebiegu ruchu gracza
- bool poprawnePierwszewylozenie - potrzebne do poprawnego przebiegu pierwszego wyłożenia

Klasa zawiera metody sprawdzające poprawność i logiczność komend graczy oraz metody potrzebne na poczet samej rozgrywki: dziel ciąg, łącz ciągi, nowy ciąg, weź żeton, wykonaj ruch. Klasa daje możliwość wywołania potrzebnych metod klasy menu.

Wizualizacja działania programu i współdziałania klas



Wnioski, napotkane problemy, uzyskana wiedza z projektu

Wraz z pisaniem projektu poznawałem nowe możliwości programowania obiektowego. Już gdzieś w połowie pisania zauważyłem, że kod wyprodukowany przy posiadaniu tylko podstawowej wiedzy jest całkiem średni. Teraz po zakończeniu pisania projektu wiem, że to, co pisałem w połowie, jest równie średnie. Na pierwszy rzut oka może to wydawać się czymś negatywnym, ale zważając, że projekt był pisany w celach dydaktycznych, można śmiało uznać to za sukces. Moja wiedza i umiejętności wzrosły na tyle, że potrafię dostrzec własne błędy.

Przekonałem się, że gdy piszemy program dla użytkownika, konieczne jest przeprowadzenie testu przez osobę z grupy odbiorczej oraz zebranie jej opinii.

Pomimo wstępnych obaw, rozgrywka w wierszu poleceń okazała się całkiem przyjemna zarówno dla mnie, jak i dla mojej testerki.

Większość problemów napotkanych podczas pisania projektu wynikała z niewiedzy lub jakości kodu napisanego w momencie, gdy moja biegłość w programowaniu obiektowym była jeszcze dosyć niska.

Wzrosła również moja znajomość i przychylność do korzystania z Git'a. Powodem była konieczność pracy na różnych urządzeniach, raz w domu, raz w akademiku. Wartość Git'a ujawniła się szczególnie, gdy miałem już działający kod, ale chciałem go rozbudować lub zoptymalizować. Po wykonanej rozbudowie program zachowywał się niepoprawnie, ale zamiast spędzać czas na debugowaniu kodu, mogłem bez problemu wrócić do poprzedniego commita sprzed rozbudowy gdzie wszystko działało i spróbować ponownie.

Pomimo ukończenia programu, planuję podczas wakacji pracować nad jego rozbudową oraz optymalizacją.

Jestem przekonany, że przyszłe projekty, dzięki zdobytej wiedzy i umiejętnościom, będą wykonane lepiej, sprawniej i dokładniej przy podobnym nakładzie czasu.