

Programowanie obiektowe – raport projekt 2 Java

W poniższym dokumencie opiszę wykonaną podczas projektu pt. „Wirtualny świat” w ramach przedmiotu Programowanie obiektowe.

Drugi projekt wykonałam w języku Java, z wykorzystaniem biblioteki Swing. Była to moja pierwsza przygoda w tym języku z tą biblioteką, także początki były ciekawe. Po zaznajomieniu się z komponentami biblioteki wybrałam kilka i później użyłam ich w projekcie.

Między innymi używałam:

KeyListener i ActionListener oraz JFrame, JMenu, JMenuItem, JButton, JPanel, JLabel i Image.

Zrealizowałam wymagania na 4 punkty, więc gra posiada:

- Implementacja świata gry i jego wizualizacji.
- Implementacja wszystkich obowiązkowych gatunków zwierząt.
- Implementacja wszystkich gatunków roślin.
- Implementacja Człowieka poruszanego za pomocą strzałek na klawiaturze.
- Implementacja specjalnej umiejętności Człowieka. (szybkość antylopy)
- Implementacja możliwości zapisania do pliku i wczytania z pliku stanu wirtualnego świata.
- Implementacja możliwości dodawania organizmów do świata gry. Naciśnięcie na wolne pole powinno dać możliwość dodania każdego z istniejących w świecie organizmów.

Analogicznie do Laboratorium 4 i 3, pliki projektu podzieliłam tematycznie na paczki: rośliny i zwierzęta, które zawierają odpowiednio pliki .java - roślin i zwierząt. Ustawiłam też ikonę gry. Wszystkie właściwości graficzne świata zawarte są w klasie i jednocześnie pliku SwiatSwing.java.



Gra rozpoczyna się powitaniem nowego gracza (JLabel):

Witaj :)



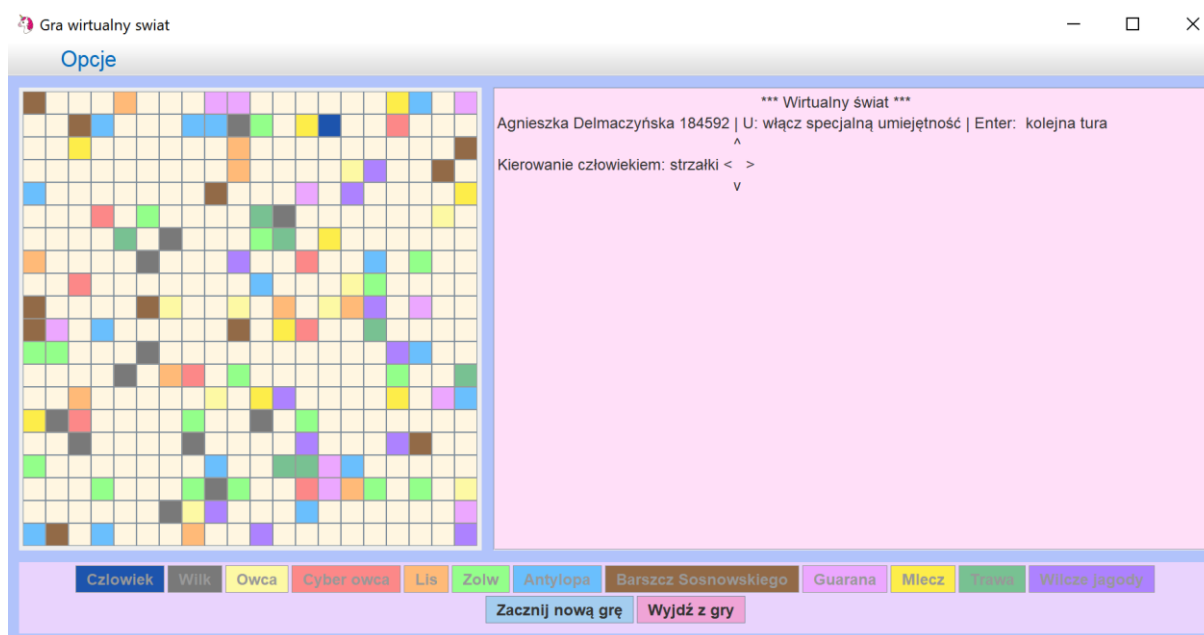
Witaj w grze Wirtualny świat! Zaczynamy?



Następnie pokazuje się okno gry z rozwijanym menu (JMenu i JMenuitem):



Wybieramy „nowa gra”



Wyświetla się stała plansza o rozmiarach 20x20 i organizmy do niej dodane (w tym człowiek). Instrukcja obsługi klawiszy oraz informacja o numerze tury jest umieszczona w panelu po prawej stronie u góry. W dolnej sekcji znajduje się legenda oznaczeń organizmów z reprezentującymi je kolorami.

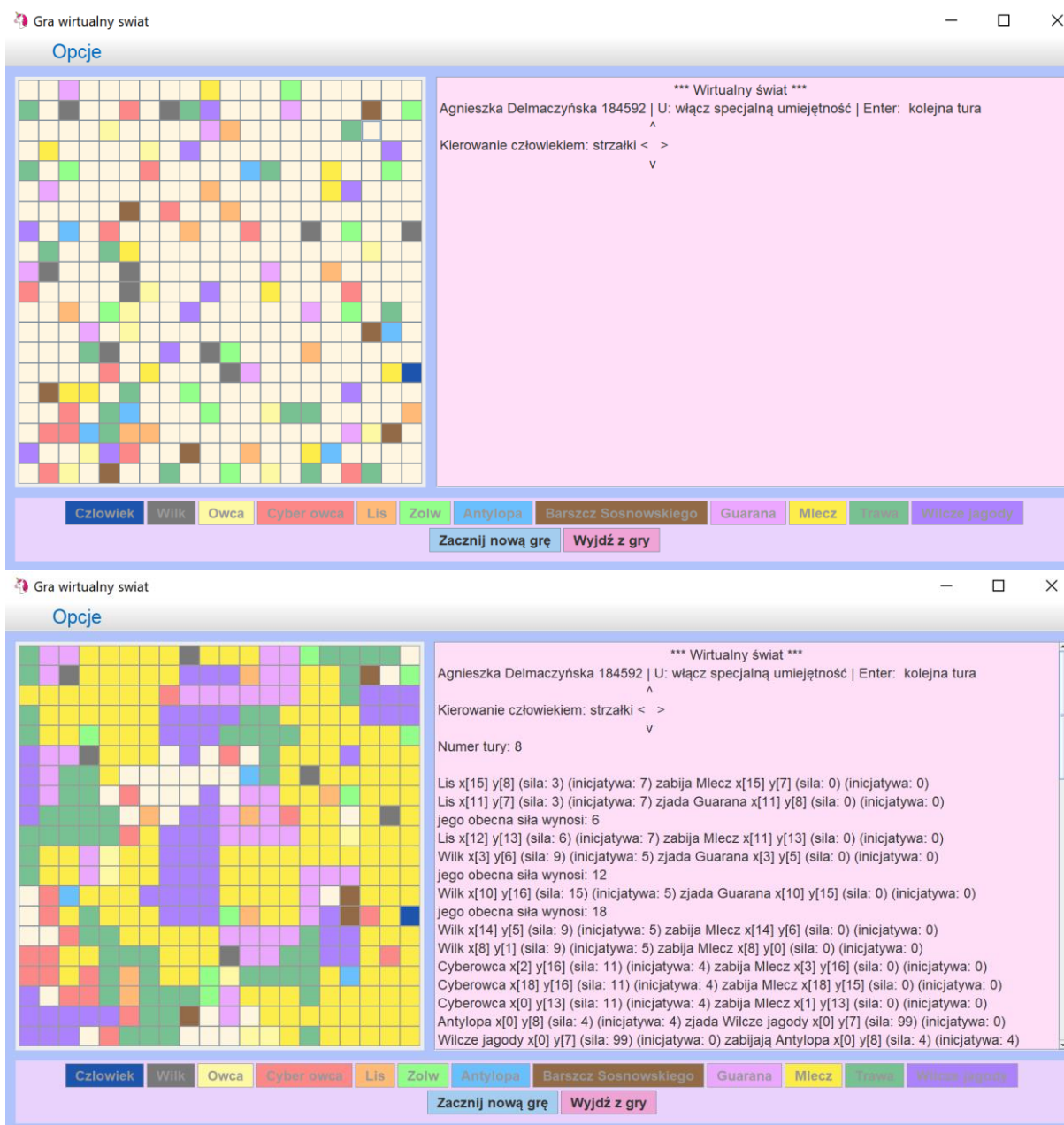
Są to JButtons, nie włączone i nie funkcyjne. Poniżej znajdują się dwa funkcyjne butony: „Zacznij nową grę” i „Wyjdź z gry”. Po kliknięciu na nie, przy użyciu funkcji ActionListener zostaje wykonana odpowiednia akcja. „Wyjdź z gry” zamyka całkowicie okienko z grą. Jest to

co prawda powtórzenie funkcji z rozwijanego menu, ale ustawienie tych przycisków „pod ręką” jest wygodniejsze w użyciu niż rozwijanie za każdym razem JMenu listy.

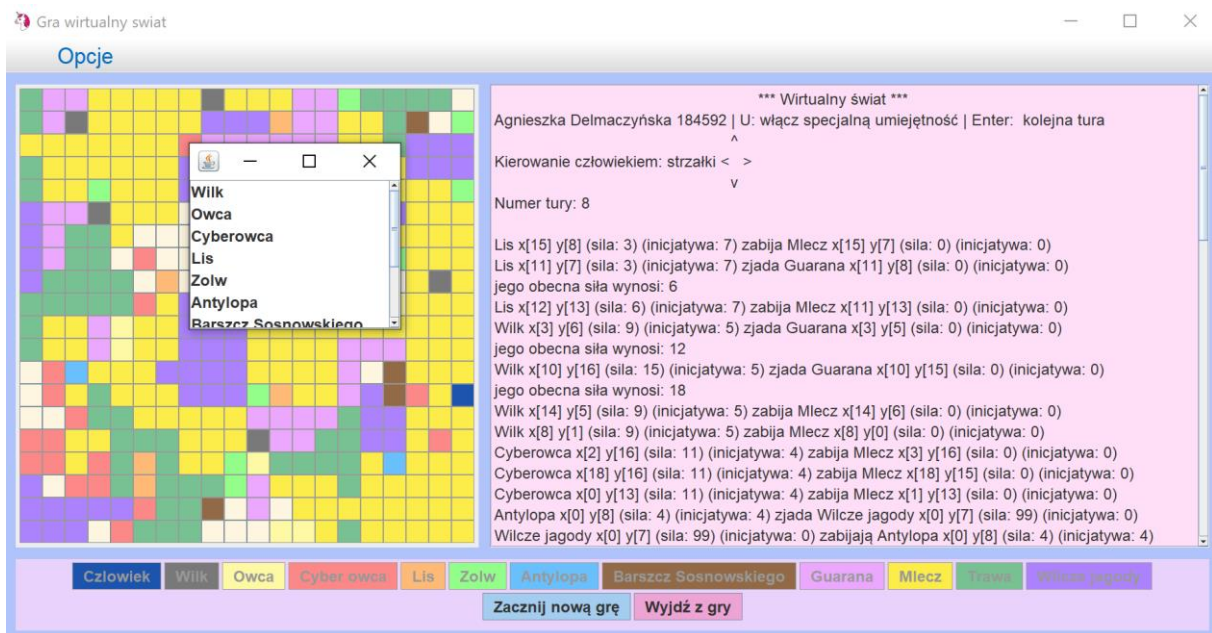
Miałam problemy z dodawaniem komponentów biblioteki Swing w odpowiednie miejsca w oknie gry, ale często powody problemu były dość śmieszne, na przykład nie implementowałam ActionListenera lub nie overrideowałam actionPerformed lub nie dodawałam do panelu lub ramki gry komponentu.

W przeciwieństwie do pierwszego projektu w C++, wyświetlanie informacji o bieżących akcjach na planszy musiało się odbyć nie w konsoli, ale w graficznym okienku gry. Zwykłe wyświetlanie tekstu w konsoli i printfowanie na każdym razem tekstu zastąpiłam tworząc klasę „Informator.java”, który w wygodniejszy sposób przekazuje daną informację do komponentu biblioteki Swing i wypisuje na ekran.

Przykładowy przebieg gry i prezentacja informacji o dokonywanych akcjach na świecie:



Po kliknięciu na wolne pole pojawia się możliwość dodania ręcznie organizmu do planszy (JList + JScrollPane, żeby nie wyświetlać od razu całej listy i nie zasłaniać przez to planszy). Można dodać każdy organizm z wyjątkiem człowieka, który jest tylko jeden. Gdy człowiek umiera, gra się kończy. Klawisz Enter również nie działa i nie przechodzi do kolejnej tury.



Sporym problemem, który napotkałam był domyślny rozmiar czcionki – bardzo mały, wręcz maciupki. Udało mi się rozwiązać ten problem za pomocą `setFont`, ale wiem, że to rozwiązanie nie jest najprawdopodobniej optymalne, ponieważ w każdym miejscu wyświetlania tekstu w komponentach biblioteki Swing musiałam dodać taką linijkę,

```
np.:jList.setFont(jList.getFont().deriveFont(24.0f));
```

Kod i screeny przykładowych klas

W projekcie utworzyłam klasę Świat (`Swiat`) i ŚwiatSwing (`SwiatSwing`) zarządzającą rozgrywką i organizmami. Zawiera ona m.in. metody, np: • `wykonaj_ture()` • `rysuj_swiat()` oraz pola.

Utworzyłam również abstrakcyjną klasę `Organizm`.

Jej podstawowe pola (`protected`):

- siła
- inicjatywa
- położenie, czyli klasa `Koordynaty` z dwoma polami, przechowującymi położenie `x` i `y`
- świat - referencja do świata w którym znajduje się organizm.

Podstawowe metody klasy `Organizm`:

- `akcja()` → określa zachowanie organizmu w trakcie tury,

- kolizja() → określa zachowanie organizmu w trakcie kontaktu/zderzenia z innym organizmem,
- rysowanie() → powoduje narysowanie symbolicznej reprezentacji organizmu. Override'uję je w klasach Zwierze i Roslina.

Klasa Organizm jest abstrakcyjna, dziedziczą po niej dwie kolejne klasy czyli Zwierze i Roslina. Odkryłam, że istnieje coś takiego jak enum i użyłam go kilkakrotnie, bardzo wygodnie się go używa. Uniknęłam tym sposobem wielu powtórzeń w kodzie.

Poniżej wstawiam część pliku Organizm.java:

```

1 package PO_projekt_2;
2 import java.awt.*;
3 import java.util.Random;
4
5 public abstract class Organizm
6 {
7     private Color kolor;
8     private Swiat swiat;
9     private final koordynaty koordynaty;
10    private TypOrganizmu typOrganizmu;
11    private int sila;
12    private int inicjatywa;
13    private int tura_urodzenia;
14    private boolean czy_zyje;
15    protected final boolean[] kierunek;
16    private boolean czy_sie_rozmnazal;
17
18    public abstract void akcja();
19    public abstract void kolizja(Organizm other);
20    public abstract String typ_organizmu_to_string();
21    public abstract boolean czy_zwierze();
22
23    public Organizm(Swiat swiat, TypOrganizmu typOrganizmu, int sila, int inicjatywa, Organizm.Koordynaty koordynaty,
24    {
25        this.swiat = swiat;
26        this.typOrganizmu = typOrganizmu;
27        this.sila = sila;
28        this.inicjatywa = inicjatywa;
29        this.koordynaty = koordynaty;
30        this.tura_urodzenia = tura_urodzenia;
31    }
  
```

```

36 public Swiat get_swiat() { return swiat; }
37 public void setSwiat(Swiat swiat) { this.swiat = swiat; }
38
39 public Koordynaty get_koordynaty() { return koordynaty; }
40
41 public TypOrganizmu get_typ_organizmu() { return typOrganizmu; }
42 public void setTypOrganizmu(TypOrganizmu typOrganizmu) { this.typOrganizmu = typOrganizmu; }
43
44 // getters i setters do prywatnych pól: sila, inicjatywa, tura_urodzenia, czy_zyje, czy_sie_rozmnazal, kolor
45
46 public int get_sila() { return sila; }
47 public void set_sila(int sila) { this.sila = sila; }
48
49 public int get_inicjatywa() { return inicjatywa; }
50 public void setInicjatywa(int inicjatywa) { this.inicjatywa = inicjatywa; }
51
52 public int get_tura_urodzenia() { return tura_urodzenia; }
53 public void set_tura_urodzenia(int tura_urodzenia) { this.tura_urodzenia = tura_urodzenia; }
54
55 public boolean get_czy_zyje() { return czy_zyje; }
56 public void set_czy_zyje(boolean czy_zyje) { this.czy_zyje = czy_zyje; }
57
58 public boolean get_czy_sie_rozmnazal() { return czy_sie_rozmnazal; }
59 public void set_czy_sie_rozmnazal(boolean czy_sie_rozmnazal) { this.czy_sie_rozmnazal = czy_sie_rozmnazal; }
60
61 public Color get_kolor() { return kolor; }
62 public void set_kolor(Color kolor) { this.kolor = kolor; }
63
64 public enum Kierunek // enum wszystkich kierunków, w tym braku kierunku, gdy wszystkie są zajęte
65 {
66
  
```

```

public enum Kierunek // enum wszystkich kierunków, w tym braku kierunku, gdy wszystkie są za...
{
    PRAWO(0),
    LEWO(1),
    GORA(2),
    DOL(3),
    BRAK_KIERUNKU(4);
    private final int kierunek;
    Kierunek(int kierunek) { this.kierunek = kierunek; }
    public int getValue() { return kierunek; }
}

public enum TypOrganizmu // enum wszystkich typów organizmów, które występują na świecie
{
    CZLOWIEK,
    WILK,
    OWCA,
    LIS,
    ZOLW,
    ANTYLOPA,
    CYBER_OWCA,
    TRAMA,
    MLECZ,
    GUARANA,
    WILCZE_JAGODY,
    BARSZCZ_SOSNOWSKIEGO
}

static TypOrganizmu daj_dowolny_organizm() // losuje dowolny typ organizmu, użyte przy dodawaniu organizmów do planu
{
}

```

```

static TypOrganizmu daj_dowolny_organizm() // losuje dowolny typ organizmu, użyte przy dodawaniu
{
    Random rand = new Random();
    int tmp = rand.nextInt(11);
    switch(tmp)
    {
        case 0: return TypOrganizmu.WILK;
        case 1: return TypOrganizmu.OWCA;
        case 2: return TypOrganizmu.CYBER_OWCA;
        case 3: return TypOrganizmu.LIS;
        case 4: return TypOrganizmu.ZOLW;
        case 5: return TypOrganizmu.ANTYLOPA;
        case 6: return TypOrganizmu.TRAMA;
        case 7: return TypOrganizmu.MLECZ;
        case 8: return TypOrganizmu.GUARANA;
        case 9: return TypOrganizmu.WILCZE_JAGODY;
        case 10: return TypOrganizmu.BARSZCZ_SOSNOWSKIEGO;
    }
    return TypOrganizmu.WILK;
}

public String OrganizmToString() // informacje wyświetlane o każdym organizmie: pozycja, siła, inicjatywa
{
    return (typ_organizmu_to_string() + " x[" + koordynaty.getX() + " y[" + koordynaty.getY() + " (siła: " + siła);
}

public boolean supermoc(Organizm wykonujacy_akcje, Organizm atakowany) // supermoc nadpisywana w poszczególnych org
{
    return false;
}

```

W klasie Zwierze zaimplementowałam wspólne dla wszystkich/większości zwierząt zachowania, przede wszystkim:

- podstawową formę ruchu w metodzie akcja() → każde typowe zwierze w swojej turze przesuwa się na wybrane losowo, sąsiednie pole,
- rozmnażanie w ramach metody kolizja() → przy kolizji z organizmem tego samego gatunku nie dochodzi do walki, oba zwierzęta pozostają na swoich miejscach, koło nich pojawia się trzecie zwierze, tego samego gatunku.

- `supermoc()` → specjalne zachowanie zwierzęcia podczas ataku

```

projekt_2.0 | src | PO_projekt_2 | Zwierze
Organizm.java | BarszczSosnowskiego.java | Zwierze.java | SwiatSwing.java | icon.bmp | main.java | swiat.txt | CyberOwca.java
1 package PO_projekt_2;
2 import java.util.Random;
3
4 public abstract class Zwierze extends Organizm
5 {
6     public Zwierze( Swiat swiat, TypOrganizmu typOrganizmu, int sila, int inicjatywa, Koordynaty koordynaty, int tura_urodzenia)
7     {
8         super(swiat, typOrganizmu, sila, inicjatywa, koordynaty, tura_urodzenia);
9     }
10
11     @Override
12     public boolean czy_zwierze()
13     {
14         return true;
15     }
16
17     @Override
18     public void akcja()
19     {
20         Koordynaty wylosowana_pozycja = planowanie_przyszlego_ruchu();
21         if (get_swiat().czy_miejsce_zajete(wylosowana_pozycja))
22         {
23             if (get_swiat().co_na_polu(wylosowana_pozycja) != this) kolizja(get_swiat().co_na_polu(wylosowana_pozycja));
24             else if (get_swiat().co_na_polu(wylosowana_pozycja) != this) wykonaj_ruch(wylosowana_pozycja);
25         }
26         else if (get_swiat().co_na_polu(wylosowana_pozycja) != this) wykonaj_ruch(wylosowana_pozycja);
27     }
28
29     @Override
30     public void kolizja(Organizm other)
31     {
32         System.out.println(other.OrganizmToString());
33         if (get_typ_organizmu() == other.get_typ_organizmu())

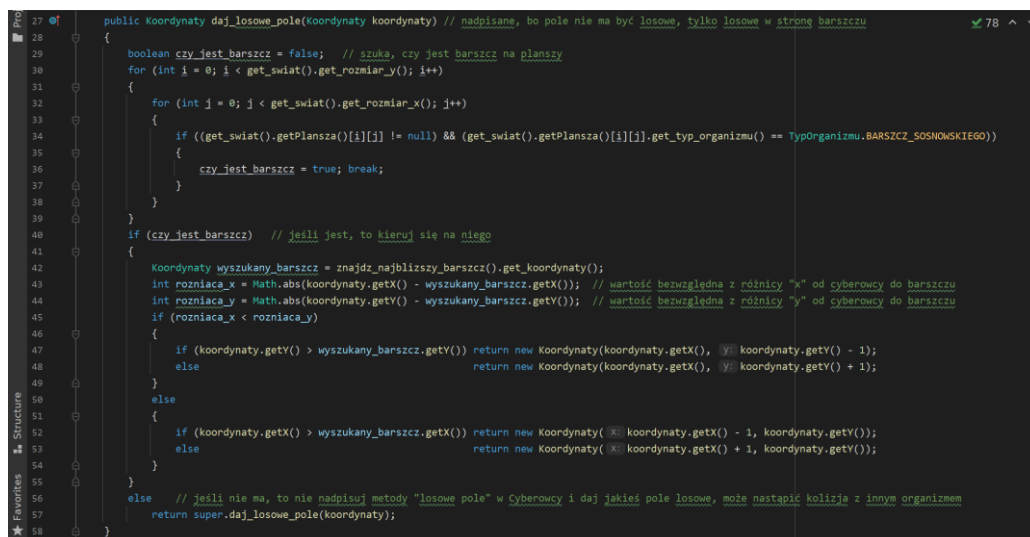
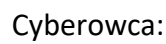
```

```

projekt_2.0 | src | PO_projekt_2 | Zwierze
Organizm.java | BarszczSosnowskiego.java | Zwierze.java | SwiatSwing.java | icon.bmp | main.java | swiat.txt | CyberOwca.java
28
29     @Override
30     public void kolizja(Organizm other)
31     {
32         System.out.println(other.OrganizmToString());
33         if (get_typ_organizmu() == other.get_typ_organizmu())
34         {
35             Random rand = new Random();
36             if (rand.nextInt(100) < 50)
37                 rozmazanie(other);
38         }
39         else
40         {
41             if (supermoc( wykonujacy_akcje this, other)) return;
42             if (other.supermoc( wykonujacy_akcje this, other)) return;
43             if (get_sila() >= other.get_sila())
44             {
45                 Informator.dodaj_informacje( komentarz other.OrganizmToString() + " zabija " + other.OrganizmToString());
46                 get_swiat().usun_organizm ze swiata(other);
47                 wykonaj_ruch(other.get_koordynaty());
48             }
49             else
50             {
51                 Informator.dodaj_informacje( komentarz other.OrganizmToString() + " zabija " + OrganizmToString());
52                 get_swiat().usun_organizm ze swiata(this);
53             }
54         }
55     }
56
57     private void rozmazanie(Organizm other)
58     {
59         if (this.get_czy_sie_rozmazal() || other.get_czy_sie_rozmazal()) return;

```

Poniżej załączam przykładowy jednego ze zwierząt.



```

60 private BarszczSosnowskiego znajdz_najblizszy_barszcz()
61 {
62     BarszczSosnowskiego najblizszy_barszcz = null;
63     int najkrotsza_droga = get_swiat().get_rozmiar_x() + get_swiat().get_rozmiar_y(); // suma x i y, czyli max odległość, jaka może być po przekątnej
64     for (int i = 0; i < get_swiat().get_rozmiar_y(); i++)
65     {
66         for (int j = 0; j < get_swiat().get_rozmiar_x(); j++)
67         {
68             Organizm organizm = get_swiat().getPlansza()[i][j];
69             if ((organizm != null) && (organizm.get_typ_organizmu() == TypOrganizmu.BARSZCZ_SOSNOWSKIEGO))
70             {
71                 int rozniaca_x = Math.abs(get_koordynaty().getX() - organizm.get_koordynaty().getX());
72                 int rozniaca_y = Math.abs(get_koordynaty().getY() - organizm.get_koordynaty().getY());
73                 int nowa_potencjalna_najkrotsza_droga = rozniaca_x + rozniaca_y;
74                 if (najkrotsza_droga > nowa_potencjalna_najkrotsza_droga) // jeśli znaleziono krótszą drogę do innego barszcza
75                 {
76                     najkrotsza_droga = nowa_potencjalna_najkrotsza_droga;
77                     najblizszy_barszcz = (BarszczSosnowskiego) organizm; // ustaw nowy barszcz jako najblizszy
78                 }
79             }
80         }
81     }
82     return najblizszy_barszcz;
83 }
84 }

```

Stworzyłam również klasę Człowiek, która dziedziczy po klasie Zwierze. Nie posiada on własnej inteligencji. Sterowany jest przez gracza strzałkami na klawiaturze, prawo, lewo, góra, dół. Posiada on umiejętność, Umiejetnosc jako klasa wewnątrz klasy Człowiek.

Człowiek jest tylko jeden i się nie rozmnaża. Jego specjalną umiejętnością jest szybkość antylopy (mój indeks: 184592, (ostatnia cyfra indeksu)mod 5 == 2, 2 to szybkość antylopy). Po włączeniu tej umiejętności klikając U, człowiek zyskuje umiejętność. Jej czas trwania wynosi 5 tur, a możliwość ponownej aktywacji specjalnej umiejętności wynosi 5 tur po wygaśnięciu uprzednio włączonej specjalnej umiejętności.

```

projekt_2.0 | src | PO_projekt_2 | zwierzeta | Czlowiek | szybkosc_antylopy
package PO_projekt_2.zwierzeta;
import PO_projekt_2.Swiat;
import PO_projekt_2.Informator;
import PO_projekt_2.Zwierze;
import java.awt.*;
import java.util.Random;

public class Czlowiek extends Zwierze
{
    private static final int CZLOWIEK_SILA = 5;
    private static final int CZLOWIEK_INICJATYWA = 4;
    private Kierunek kierunek_ruchu;
    private final Umiejetnosc umiejetnosc;

    public Czlowiek(Swiat swiat, Koordynaty koordynaty, int tura_urodzenia)
    {
        super(swiat, TypOrganizmu.CZLOWIEK, CZLOWIEK_SILA, CZLOWIEK_INICJATYWA, koordynaty, tura_urodzenia);
        kierunek_ruchu = Kierunek.BRAK_KIERUNKU;
        umiejetnosc = new Umiejetnosc();
        set_kolor(new Color(0, 29, 11, 84, 12, 173));
    }

    @Override
    public String typ_organizmu_to_string() { return "Człowiek"; }

    @Override
    public void akcja()
    {
        if (umiejetnosc.get_czy_umiejetnosc_aktywna())
        {
            Informator.dodaj_informacje( komentarz: OrganizmToString() + "\n 'Szybkość antylopy' jest aktywna (pozostały czas trwania: " +
                + umiejetnosc.get_czas_trwania_umiejetnosci() + " tur)");
            szybkosc_antylopy();
        }
    }
}

```

```

28
29
30 @Override
31 public void akcja()
32 {
33     if (umiejtnosc.get_czy_umiejtnosc_aktyna())
34     {
35         Informator.dodaj_informacje( komentarz: OrganizmToString() + ":\n 'Szybkość antylopy' jest aktywna (pozostały czas trwania: " +
36             + umiejtnosc.get_czas_trwania_umiejtnosci() + " tur)");
37         szybkosc_antylopy();
38     }
39     Koordynaty zaplanowane_przyszle_miejsce = planowanie_przyszlego_ruchu();
40     if (get_swiat().czy_miejsce_zajete(zaplanowane_przyszle_miejsce) && get_swiat().co_na_polu(zaplanowane_przyszle_miejsce) != this)
41         kolizja(get_swiat().co_na_polu(zaplanowane_przyszle_miejsce));
42     else if (get_swiat().co_na_polu(zaplanowane_przyszle_miejsce) != this)
43         wykonaj_ruch(zaplanowane_przyszle_miejsce);
44     kierunek_ruchu = Kierunek.BRAK_KIERUNKU;
45     umiejtnosc.uaktualnij();
46 }
47
48 protected void szybkosc_antylopy()
49 {
50     if (umiejtnosc.get_czas_trwania_umiejtnosci() >= 2)
51         przestaw_koordynaty();
52     else
53     {
54         Random rand = new Random();
55         int losuj = rand.nextInt(100);
56         if (losuj < 50) // szansa, że wylosuje <50 to 50%
57             przestaw_koordynaty();
58     }
59 }

```

```

59
60 protected void przestaw_koordynaty()
61 {
62     get_swiat().getPlansza()[get_koordynaty().getY()][get_koordynaty().getX()] = null;
63     if ((kierunek_ruchu == Kierunek.PRAWO) && (this.get_koordynaty().getX() != 20 - 1))
64         this.get_koordynaty().setX(this.get_koordynaty().getX() + 1);
65     else if ((kierunek_ruchu == Kierunek.LEWO) && (this.get_koordynaty().getX() != 0))
66         this.get_koordynaty().setX(this.get_koordynaty().getX() - 1);
67     if ((kierunek_ruchu == Kierunek.GORA) && (this.get_koordynaty().getY() != 0))
68         this.get_koordynaty().setY(this.get_koordynaty().getY() - 1);
69     else if ((kierunek_ruchu == Kierunek.DOL) && (this.get_koordynaty().getY() != 20 - 1))
70         this.get_koordynaty().setY(this.get_koordynaty().getY() + 1);
71 }
72
73 @Override
74 protected Koordynaty planowanie_przyszlego_ruchu()
75 {
76     int poz_x = get_koordynaty().getX();
77     int poz_y = get_koordynaty().getY();
78     daj_losowe_pole(get_koordynaty());
79     if ((kierunek_ruchu == Kierunek.BRAK_KIERUNKU) || kierunek_zablokowany(kierunek_ruchu))
80         return get_koordynaty();
81     else
82     {
83         if (kierunek_ruchu == Kierunek.PRAWO) return new Koordynaty( X poz_x + 1, poz_y);
84         if (kierunek_ruchu == Kierunek.LEWO) return new Koordynaty( X poz_x - 1, poz_y);
85         if (kierunek_ruchu == Kierunek.GORA) return new Koordynaty(poz_x, Y poz_y - 1);
86         if (kierunek_ruchu == Kierunek.DOL) return new Koordynaty(poz_x, Y poz_y + 1);
87         else return new Koordynaty(poz_x, poz_y);
88     }
89 }
90

```

```

91
92 public class Umiejtnosc
93 {
94     protected final int CZAS_TRWANIA_UMIEJTNOSCI = 5;
95     protected final int DOSTEPNY_CZAS_AKTYWACJI = 10;
96     protected boolean czy_aktyna;
97     protected boolean czy_mozna_aktwowac;
98     protected int czas_trwania;
99     protected int dostepny_czas_aktwacji;
100
101     public Umiejtnosc()
102     {
103         czy_aktyna = false;
104         czy_mozna_aktwowac = true;
105         czas_trwania = 0;
106         dostepny_czas_aktwacji = 0;
107     }
108
109     public boolean get_czy_umiejtnosc_aktyna() { return czy_aktyna; }
110     public void set_czy_umiejtnosc_aktyna(boolean czy_aktyna) { this.czy_aktyna = czy_aktyna; }
111
112     public boolean get_czy_mozna_aktwowac() { return czy_mozna_aktwowac; }
113     public void set_czy_mozna_aktwowac(boolean czy_mozna_aktwowac) { this.czy_mozna_aktwowac = czy_mozna_aktwowac; }
114
115     public int get_czas_trwania_umiejtnosci() { return czas_trwania; }
116     public void set_czas_trwania_umiejtnosci(int czas_trwania) { this.czas_trwania = czas_trwania; }
117
118     public int get_czas_aktwacji() { return dostepny_czas_aktwacji; }
119     public void set_czas_aktwacji(int dostepny_czas_aktwacji) { this.dostepny_czas_aktwacji = dostepny_czas_aktwacji; }

```

```

120 public void uaktualnij()
121 {
122     if (dostepny_czas_aktywacji > 0) dostepny_czas_aktywacji--;
123     if (czas_trwania > 0) czas_trwania--;
124     if (czas_trwania == 0) dezaktywuj_umiejetnosc();
125     if (dostepny_czas_aktywacji == 0) czy_mozna_aktwowac = true;
126 }
127
128 public void aktywuj_umiejetnosc()
129 {
130     if (dostepny_czas_aktywacji == 0)
131     {
132         czy_aktywna = true;
133         czy_mozna_aktwowac = false;
134         dostepny_czas_aktywacji = DOSTEPNY_CZAS_AKTYWACJI;
135         czas_trwania = CZAS_TRWANIA_UMIEJETNOSCI;
136     }
137     else if (dostepny_czas_aktywacji > 0)
138     {
139         Informator.dodaj_informacje( (komentarz: "\nUmiejetnosc 'Szybkosc antylopy' bedzie mozna wlaczyc po " +
140             dostepny_czas_aktywacji + " turach\n");
141     }
142 }
143 public void dezaktywuj_umiejetnosc() { czy_aktywna = false; }
144
145 public Kierunek get_kierunek() { return kierunek_ruchu; }
146 public void set_kierunek(Kierunek kierunek_ruchu) { this.kierunek_ruchu = kierunek_ruchu; }
147 public Umiejetnosc get_umiejetnosc() { return umiejetnosc; }
148
149 }

```

Zaimplementowałam 5 klas zwierząt, dziedziczących po Zwierze i Cyberowcę: Owca, Cyberowca, Wilk, Lis, Zolw i Antylopa.

W klasie Roślina zaimplementowałam wspólne dla wszystkich/większości roślin zachowania, przede wszystkim:

- symulacja rozprzestrzeniania się rośliny w metodzie akcja() → z pewnym prawdopodobieństwem każda z roślin może „zasiać” nową roślinę tego samego gatunku na losowym, sąsiednim polu. Wszystkie rośliny mają zerową inicjatywę
- supermoc() → specjalne zachowanie rośliny podczas ataku

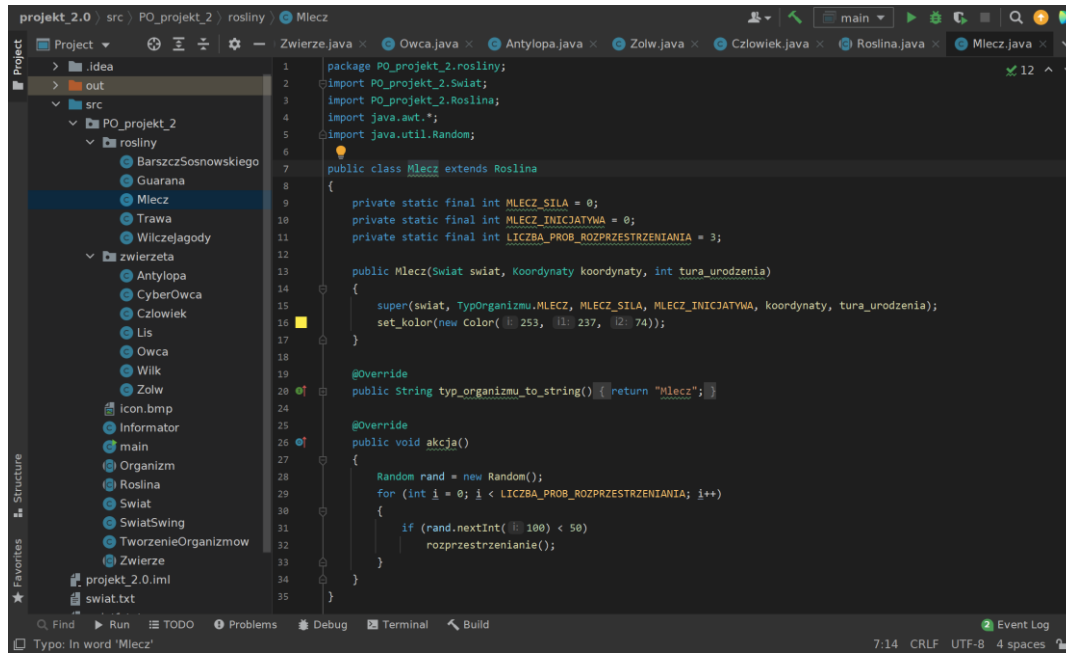
Zaimplementowałam 5 klas roślin, dziedziczących po Roślina: trawa, mlecz, guarana, wilcze jagody i barszcz Sosnowskiego. Roślina.java

```

1 package PO_projekt_2;
2 import java.util.Random;
3
4 public abstract class Roślina extends Organizm
5 {
6     protected Roślina(Swiat swiat, TypOrganizmu typOrganizmu, int sila, int inicjatywa, Koordynaty koordynaty, int tura_urodzenia)
7     {
8         super(swiat, typOrganizmu, sila, inicjatywa, koordynaty, tura_urodzenia);
9     }
10
11     @Override
12     public boolean czy_zwierze() { return false; }
13
14     @Override
15     public void akcja()
16     {
17         Random rand = new Random();
18         if (rand.nextInt(100) < 40)
19             rozprzestrzenianie();
20     }
21
22     protected void rozprzestrzenianie()
23     {
24         Koordynaty niezajete_pole = this.daj_niezajete_pole(get_koordynaty());
25         if (!niezajete_pole.equals(get_koordynaty()))
26         {
27             Organizm nowy_organizm = TworzenieOrganizmu.StworzNowyOrganizm(this.get_swiat(), get_typ_organizmu(), niezajete_pole);
28             assert nowy_organizm != null;
29             Informator.dodaj_informacje( (komentarz: "Nowa roślina: " + nowy_organizm.OrganizmToString());
30             get_swiat().pchnij_na_liste(nowy_organizm);
31         }
32     }
33 }

```

Przykładowa roślina:



The screenshot shows an IDE window with the following components:

- Project Structure:** A tree view on the left showing the project hierarchy. The 'rośliny' package is expanded, showing classes like 'Mlecz', 'Trawa', and 'Wilczjagody'. The 'zwierzeta' package is also visible.
- Code Editor:** The main area displays the source code for 'Mlecz.java'. The code is as follows:

```
1 package PO_projekt_2.rosliny;
2 import PO_projekt_2.Swiat;
3 import PO_projekt_2.Roslina;
4 import java.awt.*;
5 import java.util.Random;
6
7 public class Mlecz extends Roslina
8 {
9     private static final int MLECZ_SILA = 0;
10    private static final int MLECZ_INICJATYWA = 0;
11    private static final int LICZBA_PROB_ROZPRZESTRZENIANIA = 3;
12
13    public Mlecz(Swiat swiat, Koordynaty koordynaty, int tura_urodzenia)
14    {
15        super(swiat, TypOrganizmu.MLECZ, MLECZ_SILA, MLECZ_INICJATYWA, koordynaty, tura_urodzenia);
16        set_kolor(new Color(0, 253, 11, 237));
17    }
18
19    @Override
20    public String typ_organizmu_to_string() { return "Mlecz"; }
21
22    @Override
23    public void akcja()
24    {
25        Random rand = new Random();
26        for (int i = 0; i < LICZBA_PROB_ROZPRZESTRZENIANIA; i++)
27        {
28            if (rand.nextInt(100) < 50)
29            {
30                rozprzestrzanie();
31            }
32        }
33    }
34 }
35
```
- Bottom Panel:** Includes tabs for 'Find', 'Run', 'TODO', 'Problems', 'Debug', 'Terminal', and 'Build'. The status bar at the bottom shows '7:14 CRLF UTF-8 4 spaces'.