**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:

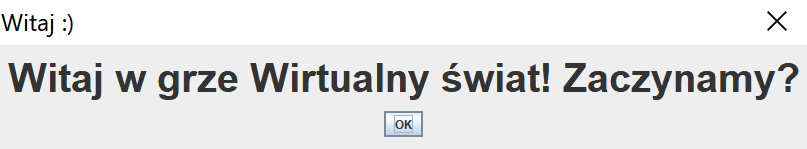
KeyListener i ActionListener oraz JFrame, JMenu, JMenuItem, JButton, JPanel, JLabel i Image.

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):



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



Wybieramy „nowa gra”



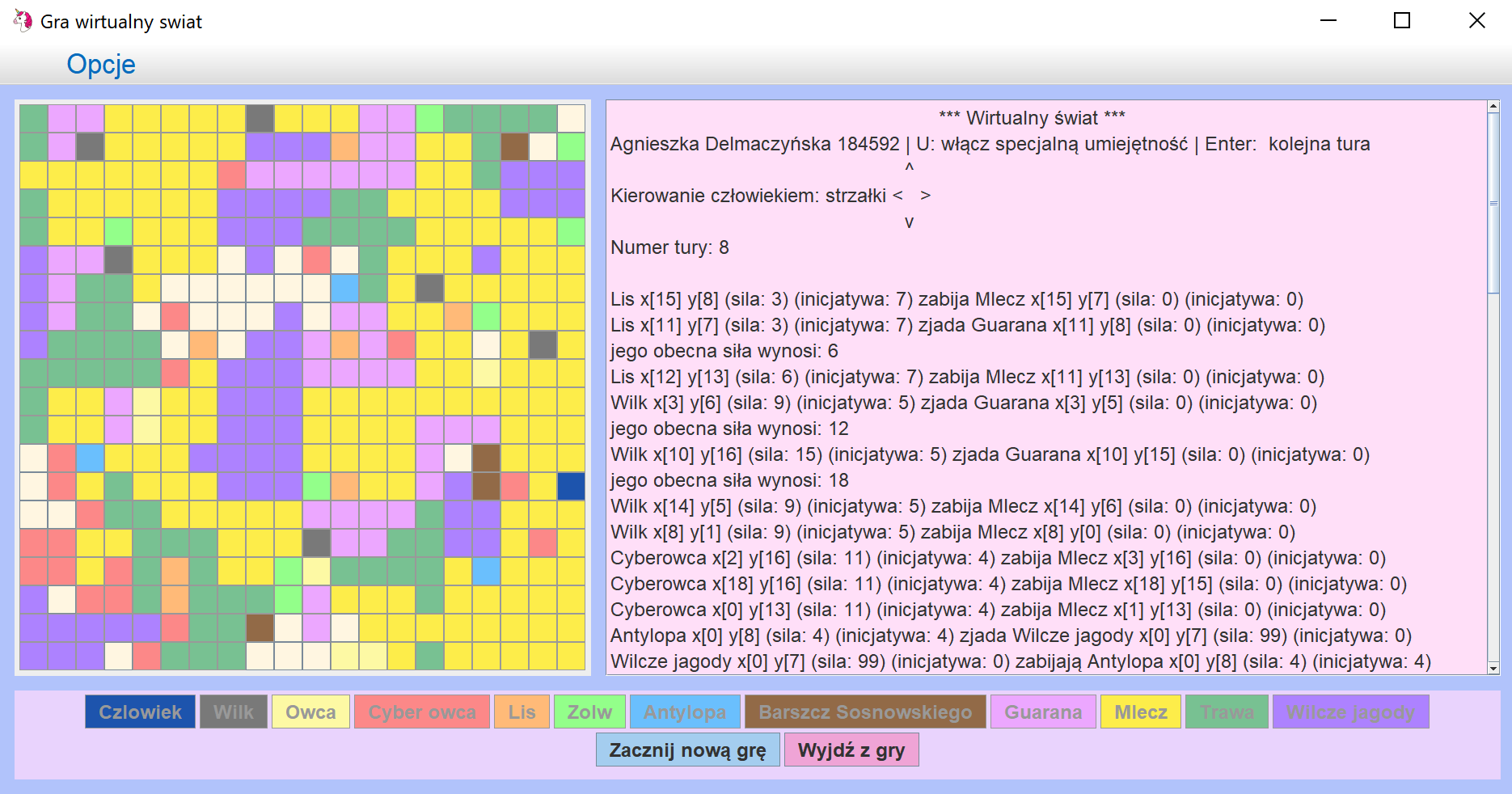
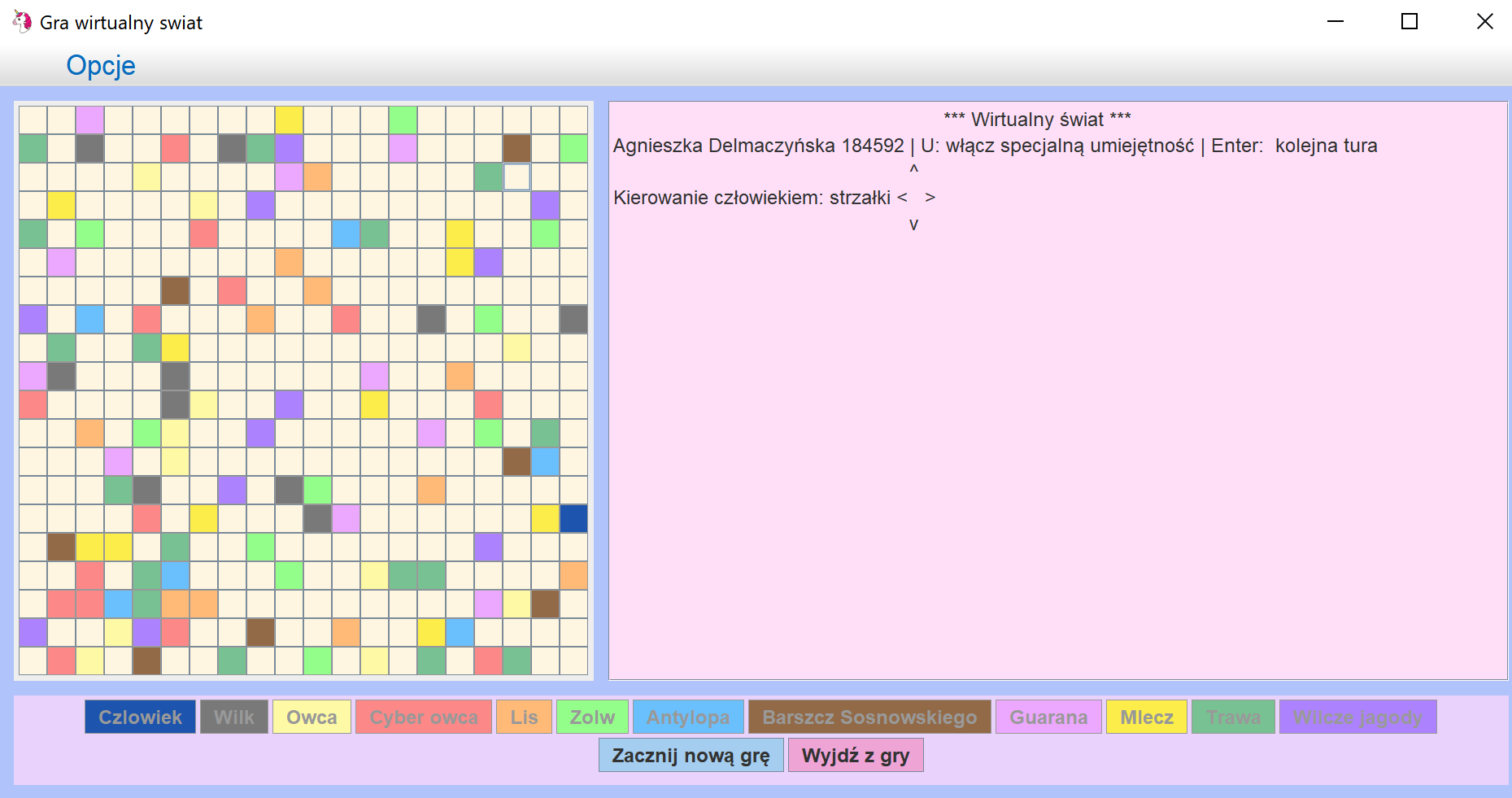
Wyświetla się stała plansza o rozmiarach 20x20 i organizmy do niej dodane (w tym człowiek). Instrukcja obsługi klawiszy oraz informacja o numerze tury jest umieszczona w panelu po prawej stronie u góry. W dolnej sekcji znajduje się legenda oznaczeń organizmów z reprezentującymi je kolorami.

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.

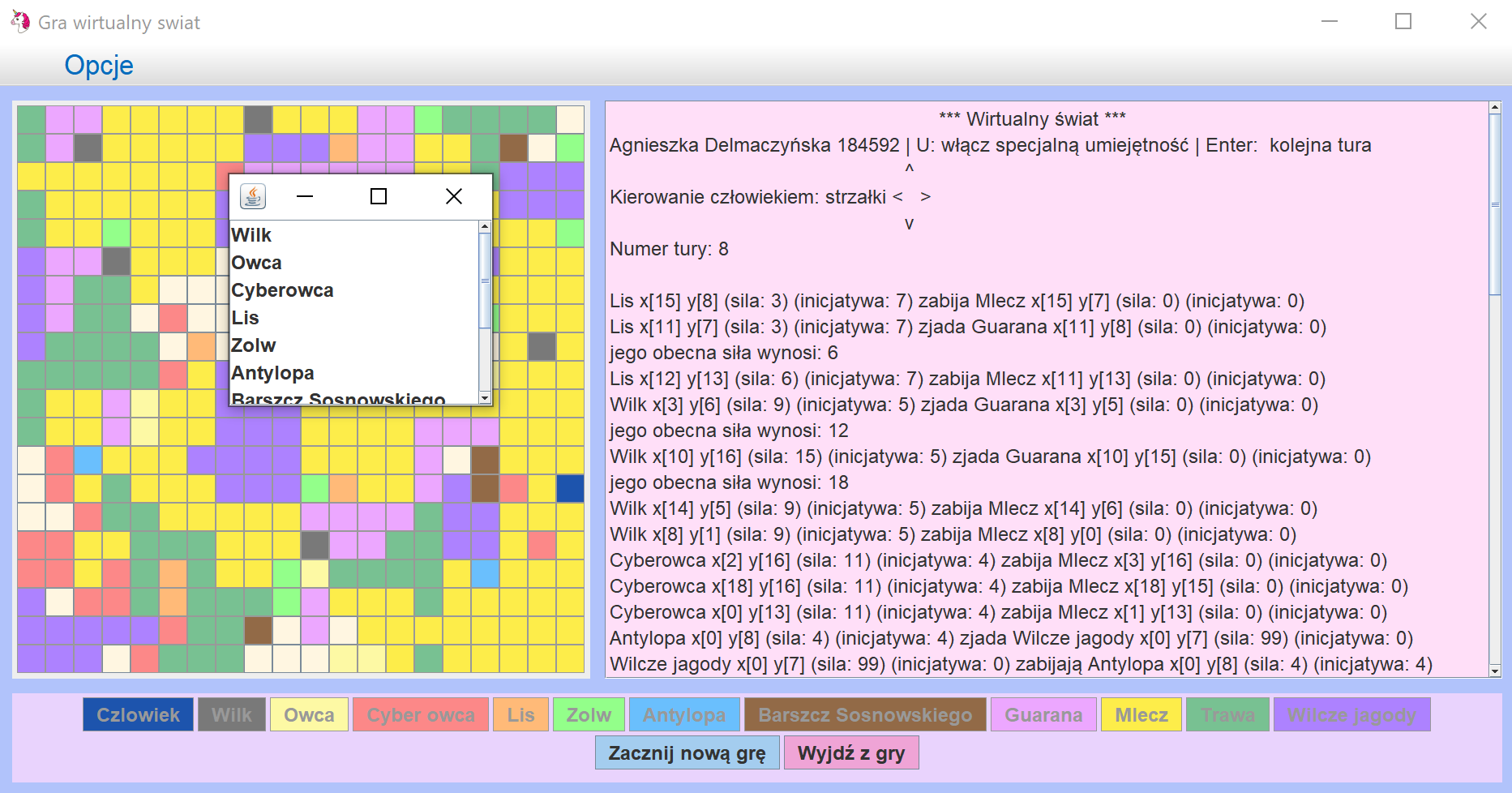
Miałam problemy z dodawaniem komponentów biblioteki Swing w odpowiednie miejsca w oknie gry, ale często powody problemu były dość śmieszne, na przykład nie implementowałam ActionListenera lub nie overridowałam ationPerformed lub nie dodawałam do panelu lub ramki gry komponentu.

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 pfintfowanie 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.

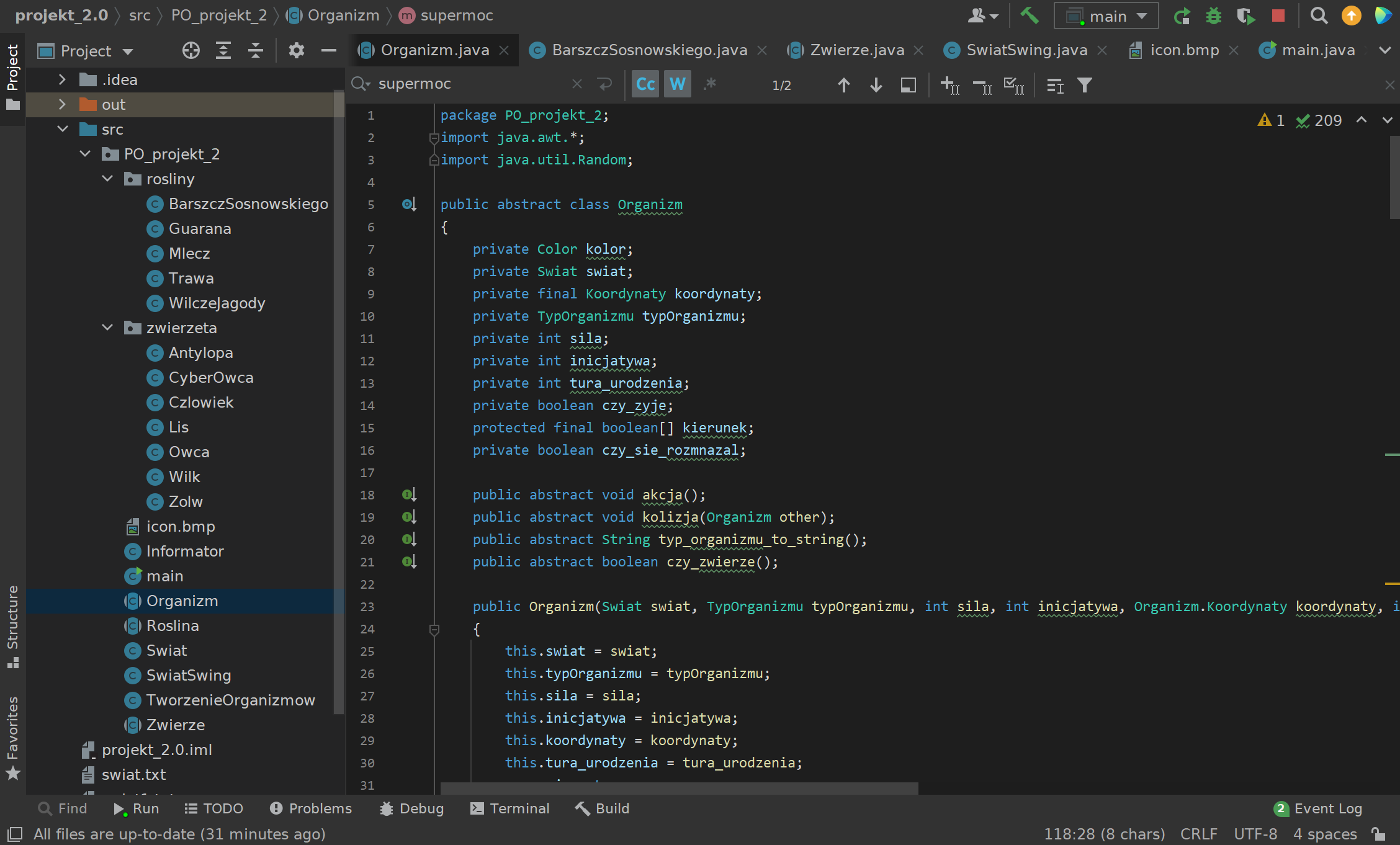
Podstawowe metody klasy Organizm:

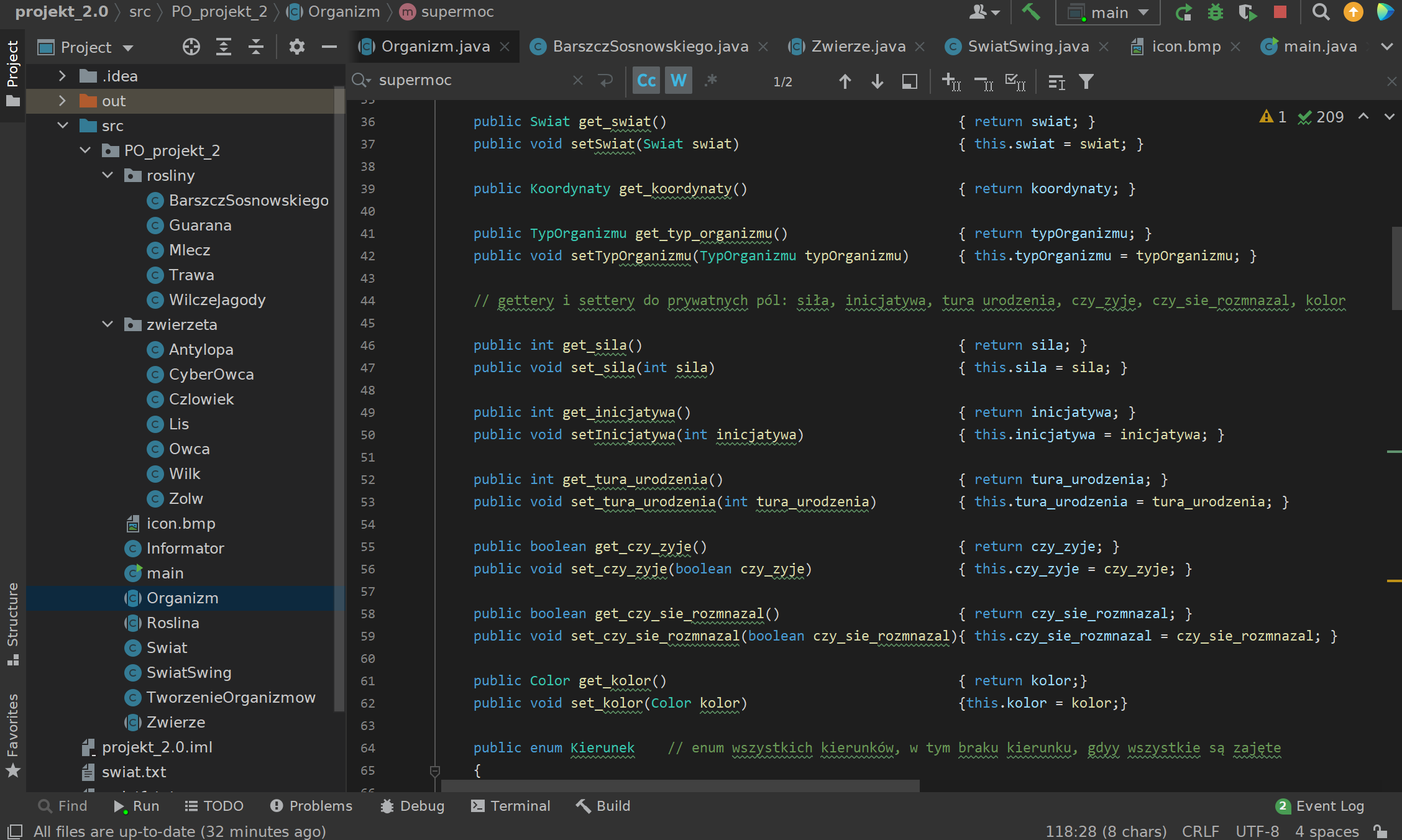
• akcja() → określa zachowanie organizmu w trakcie tury,

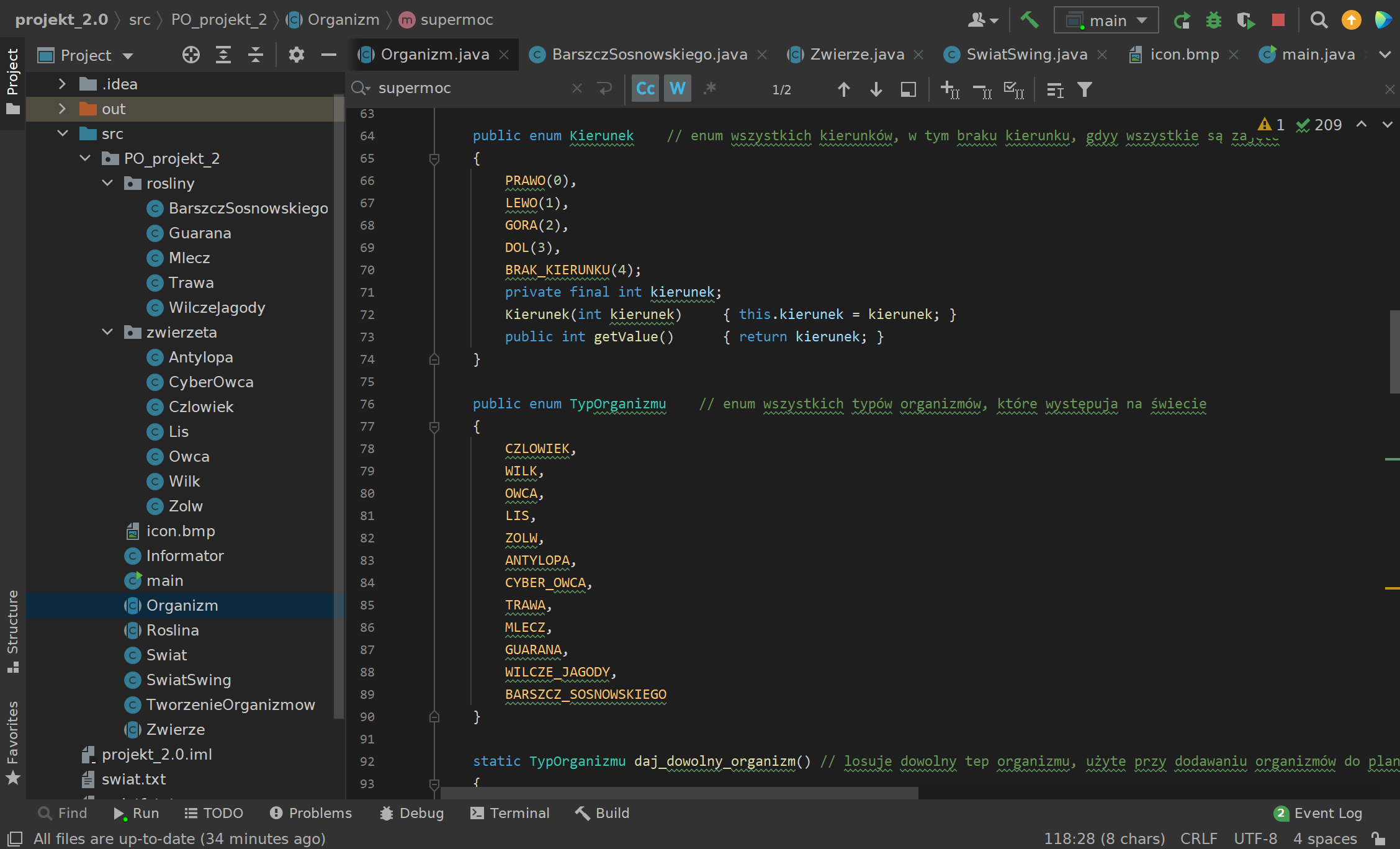
• kolizja() → określa zachowanie organizmu w trakcie kontaktu/zderzenia z innym organizmem,

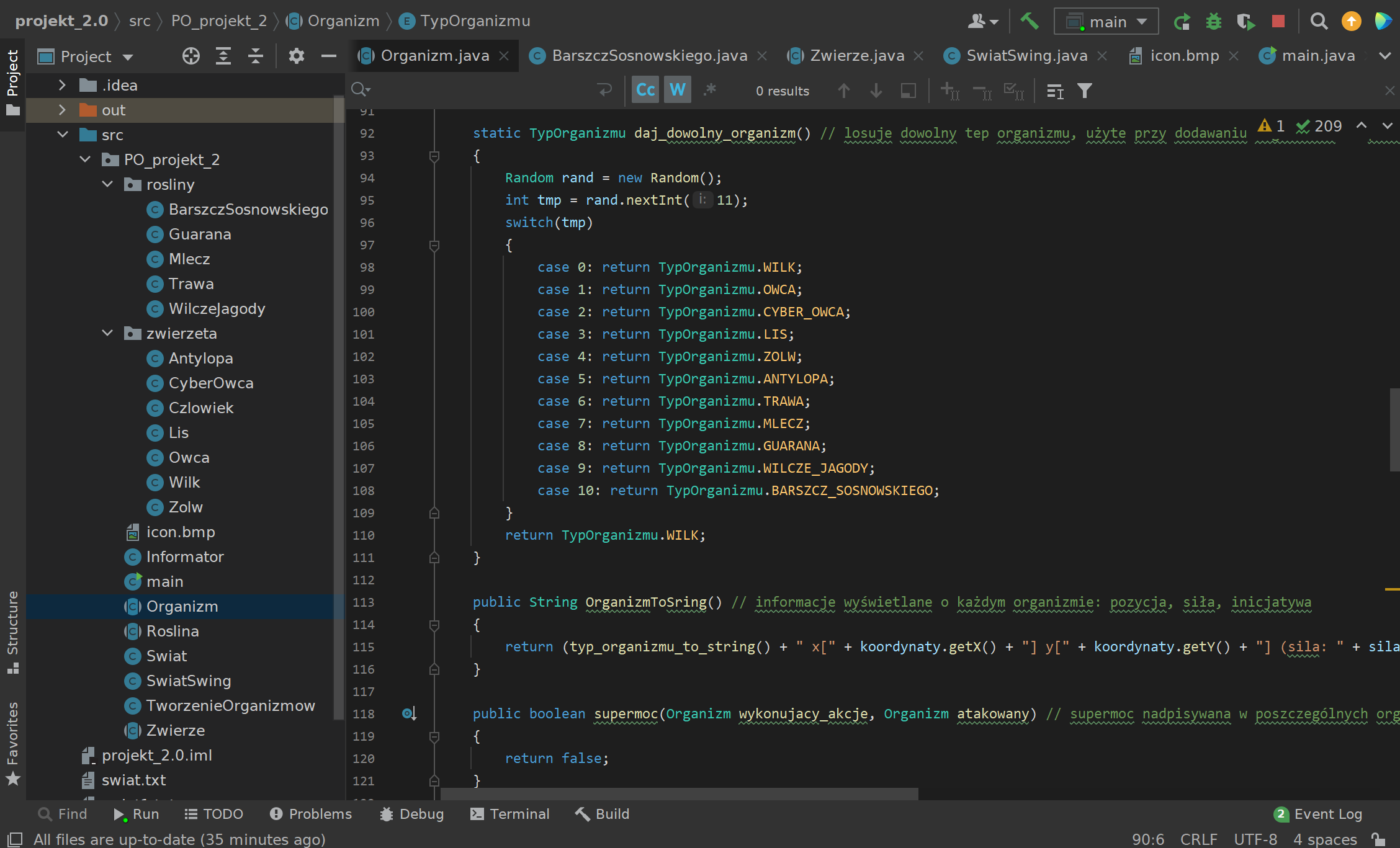
• rysowanie() → powoduje narysowanie symbolicznej reprezentacji organizmu. Override’uję je w klasach Zwierze i Roslina.

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

Poniżej wstawiam część pliku Organizm.java: 





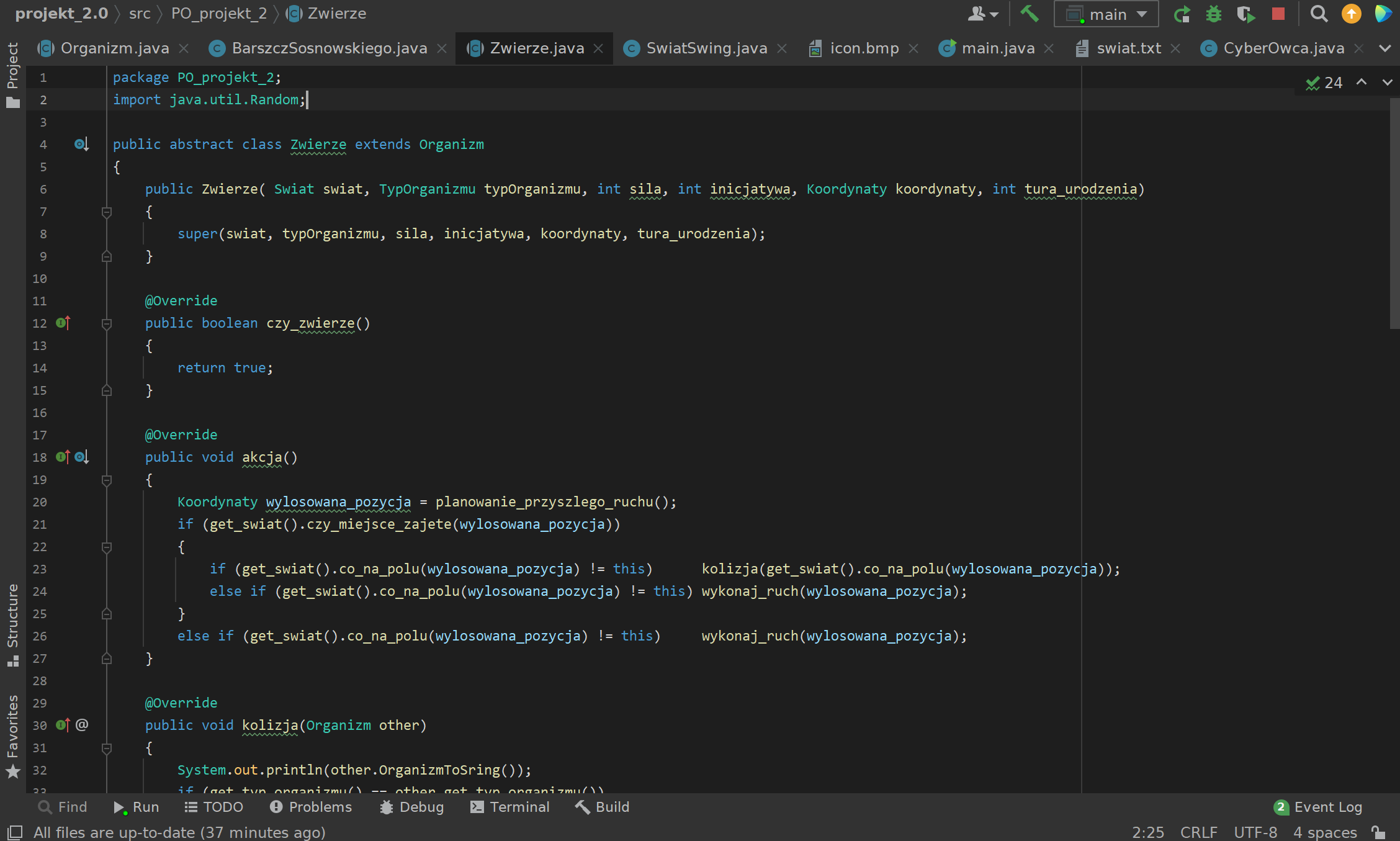


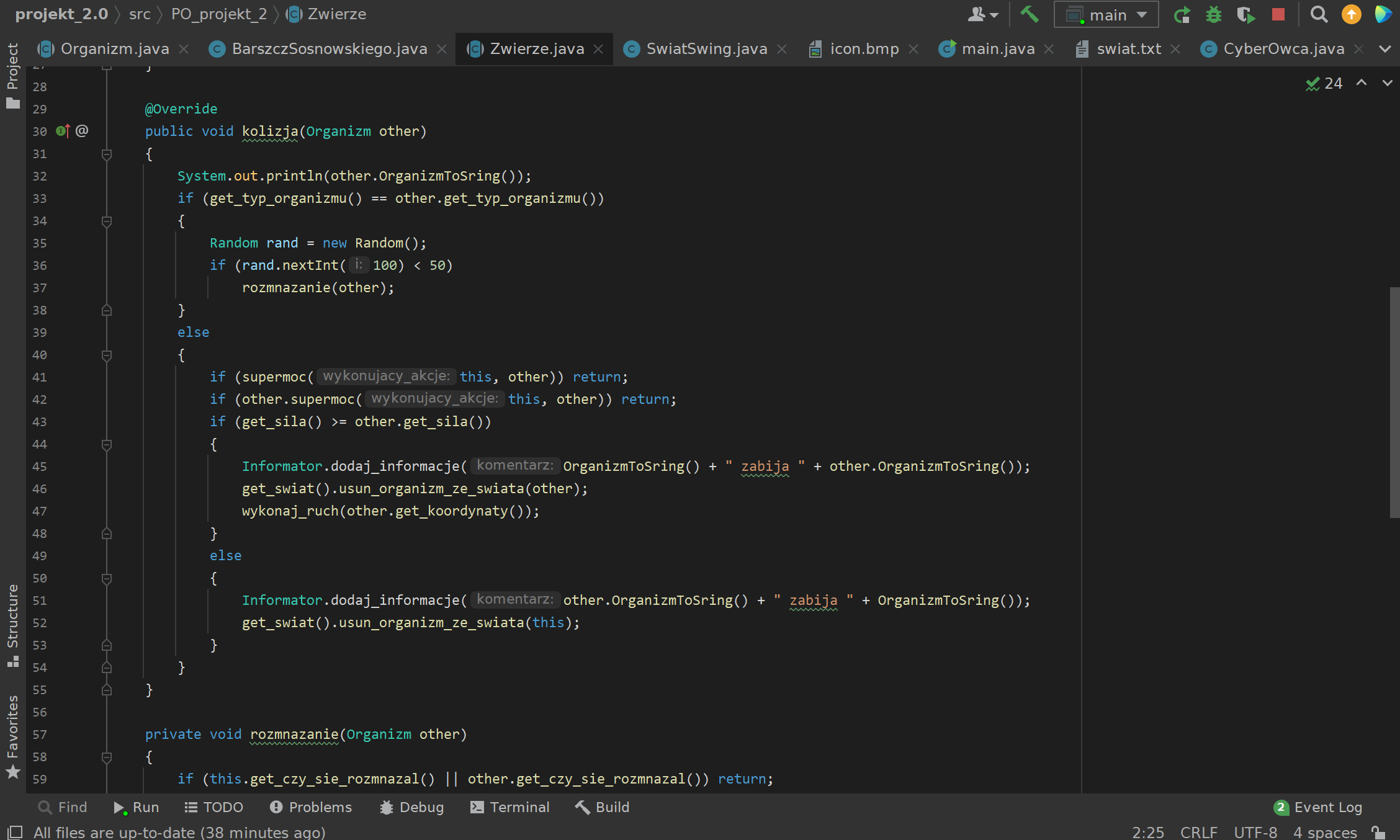
W klasie Zwierze zaimplementowałam wspólne dla wszystkich/większości zwierząt zachowania, przede wszystkim:

• podstawową formę ruchu w metodzie akcja() → każde typowe zwierze w swojej turze przesuwa się na wybrane losowo, sąsiednie pole,

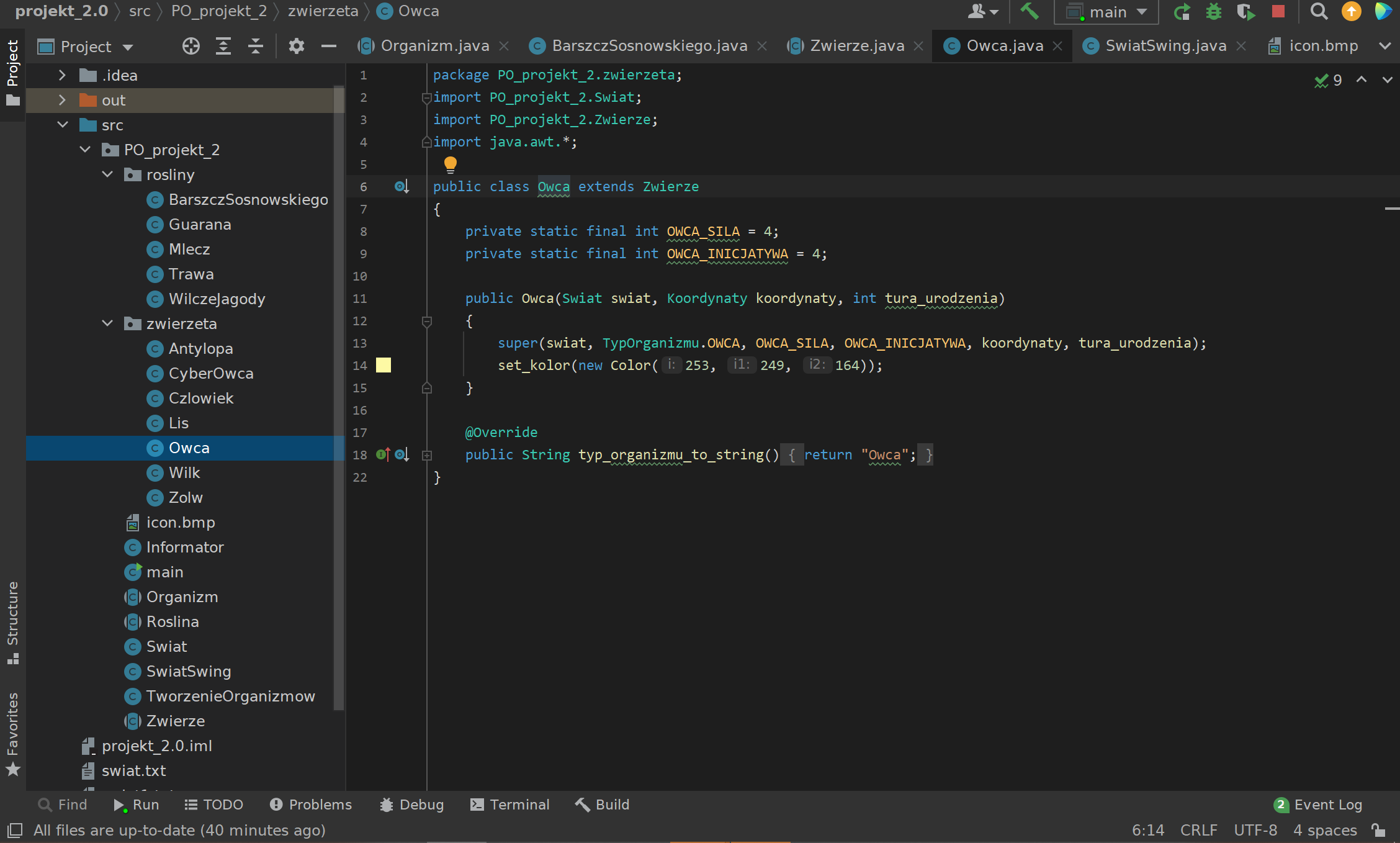
• rozmnażanie w ramach metody kolizja() → przy kolizji z organizmem tego samego gatunku nie dochodzi do walki, oba zwierzęta pozostają na swoich miejscach, koło nich pojawia się trzecie zwierze, tego samego gatunku.

• supermoc() → specjalne zachowanie zwierzęcia podczas ataku

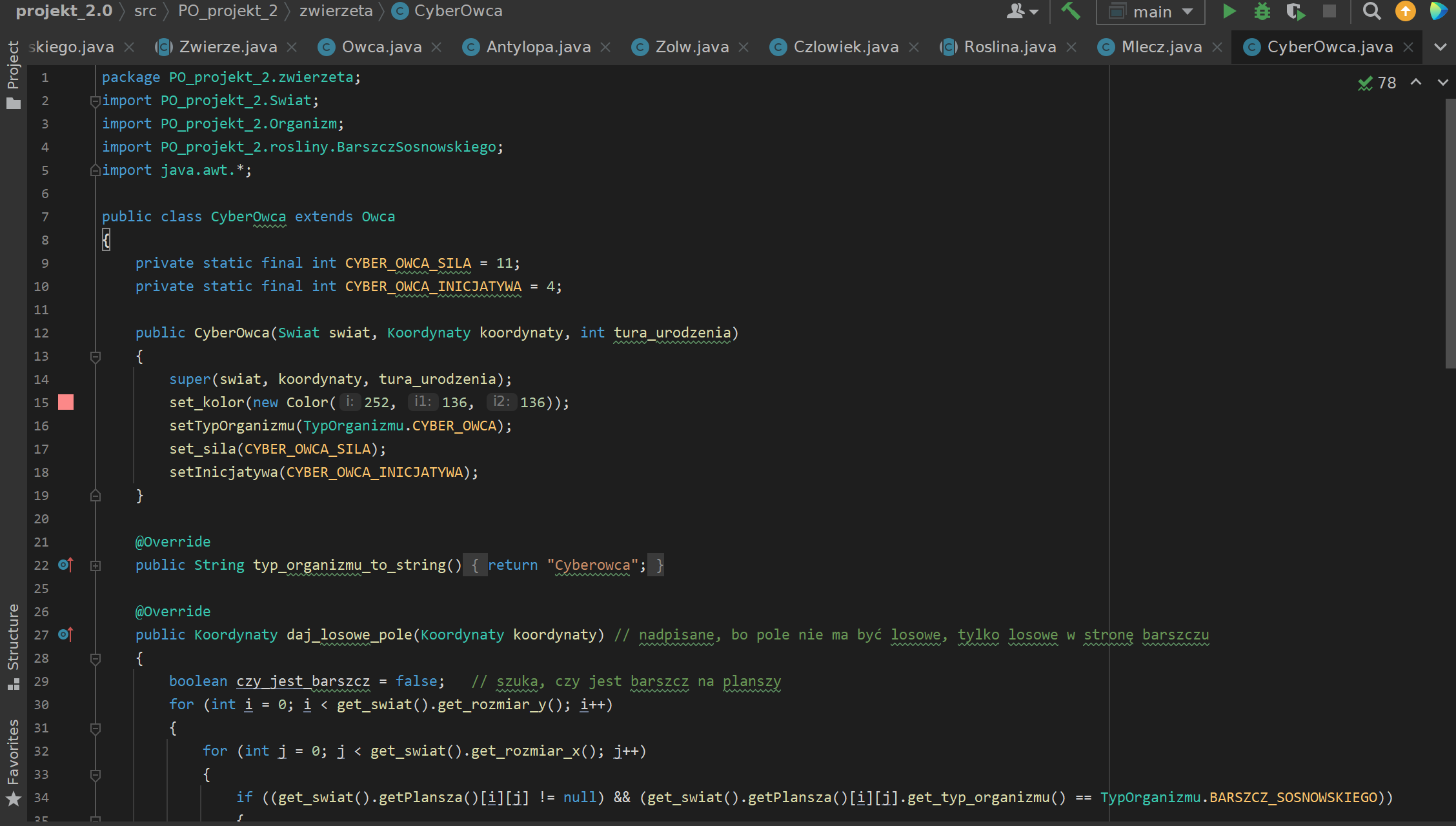


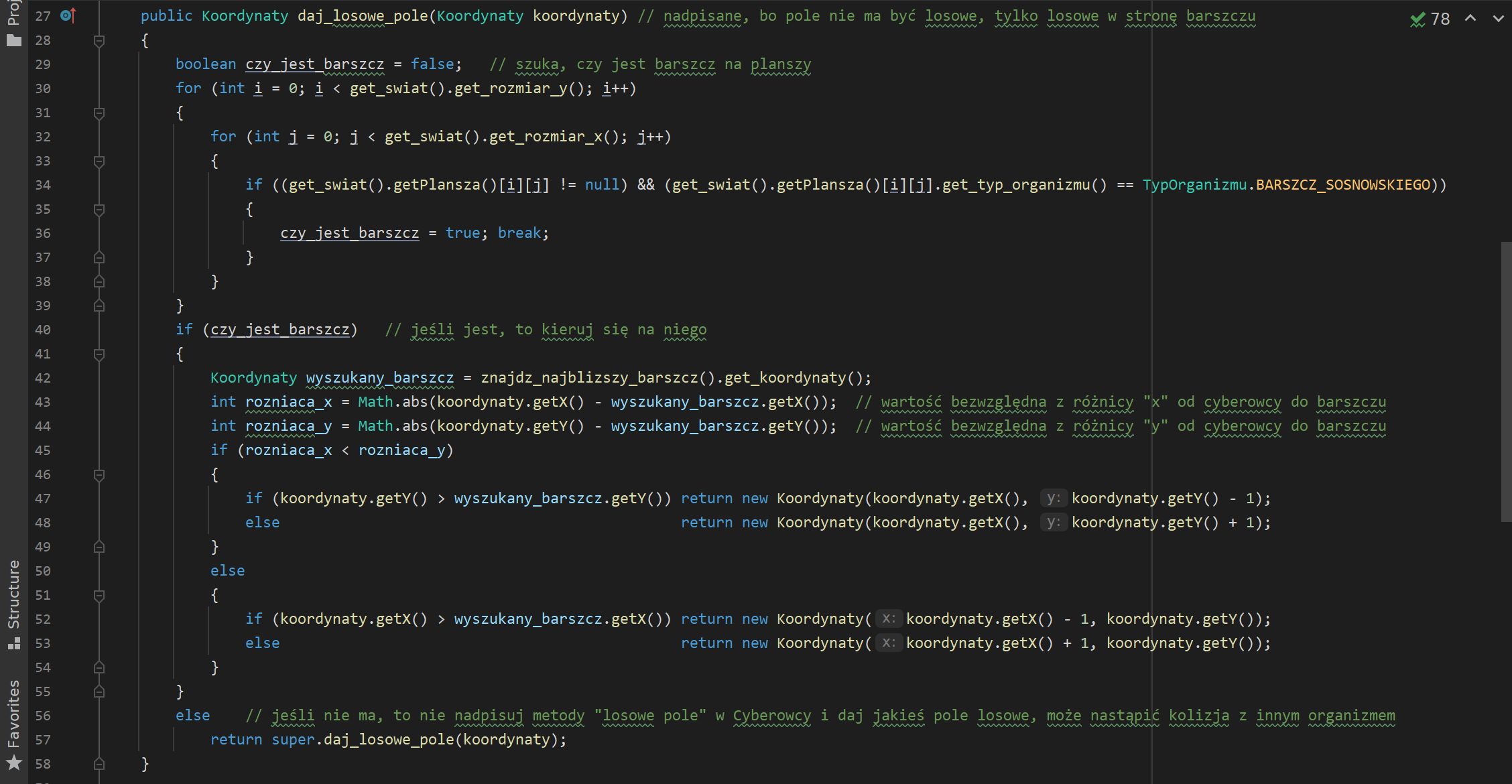


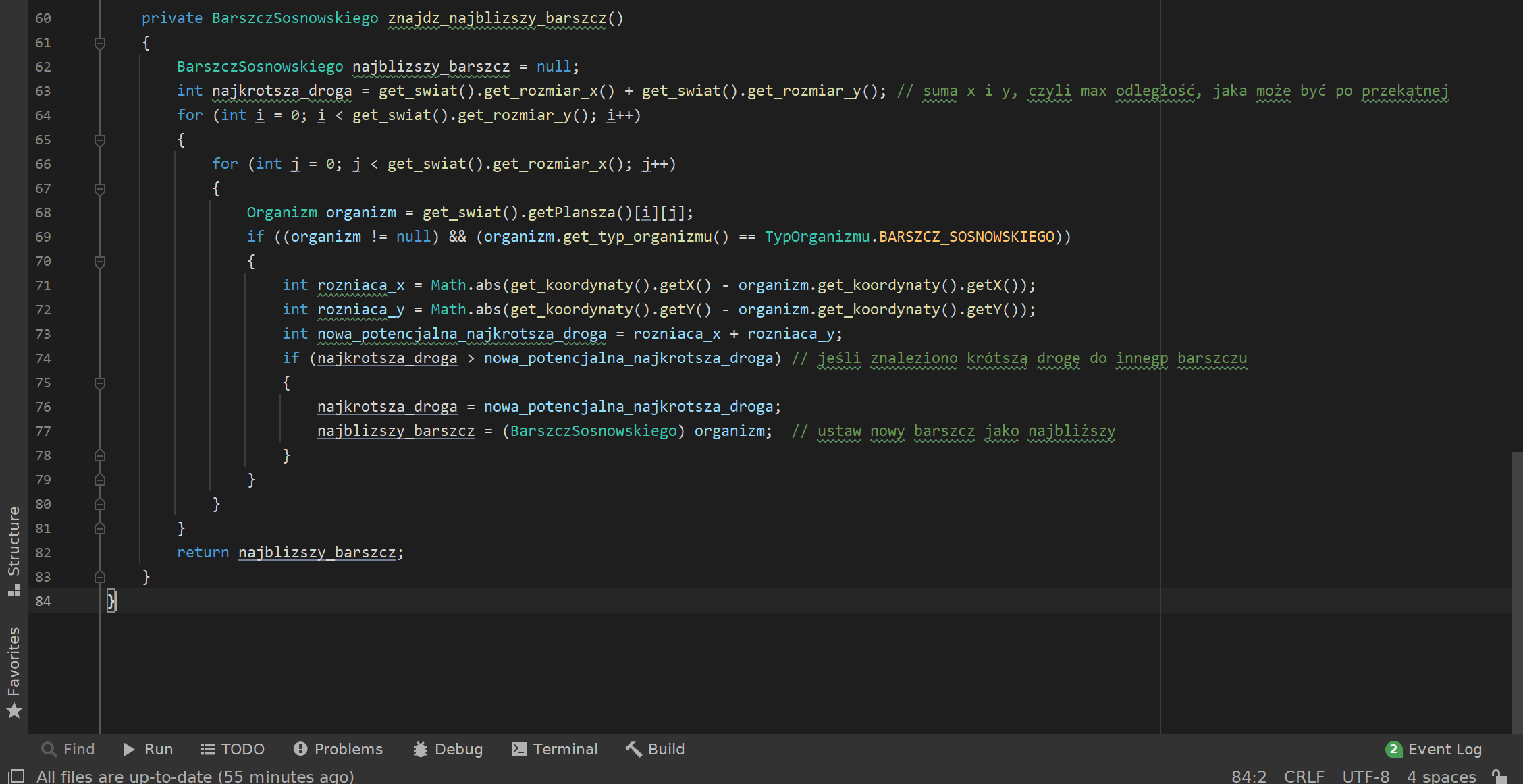
Poniżej załączam przykładowy jednego ze zwierząt.



Cyberowca:

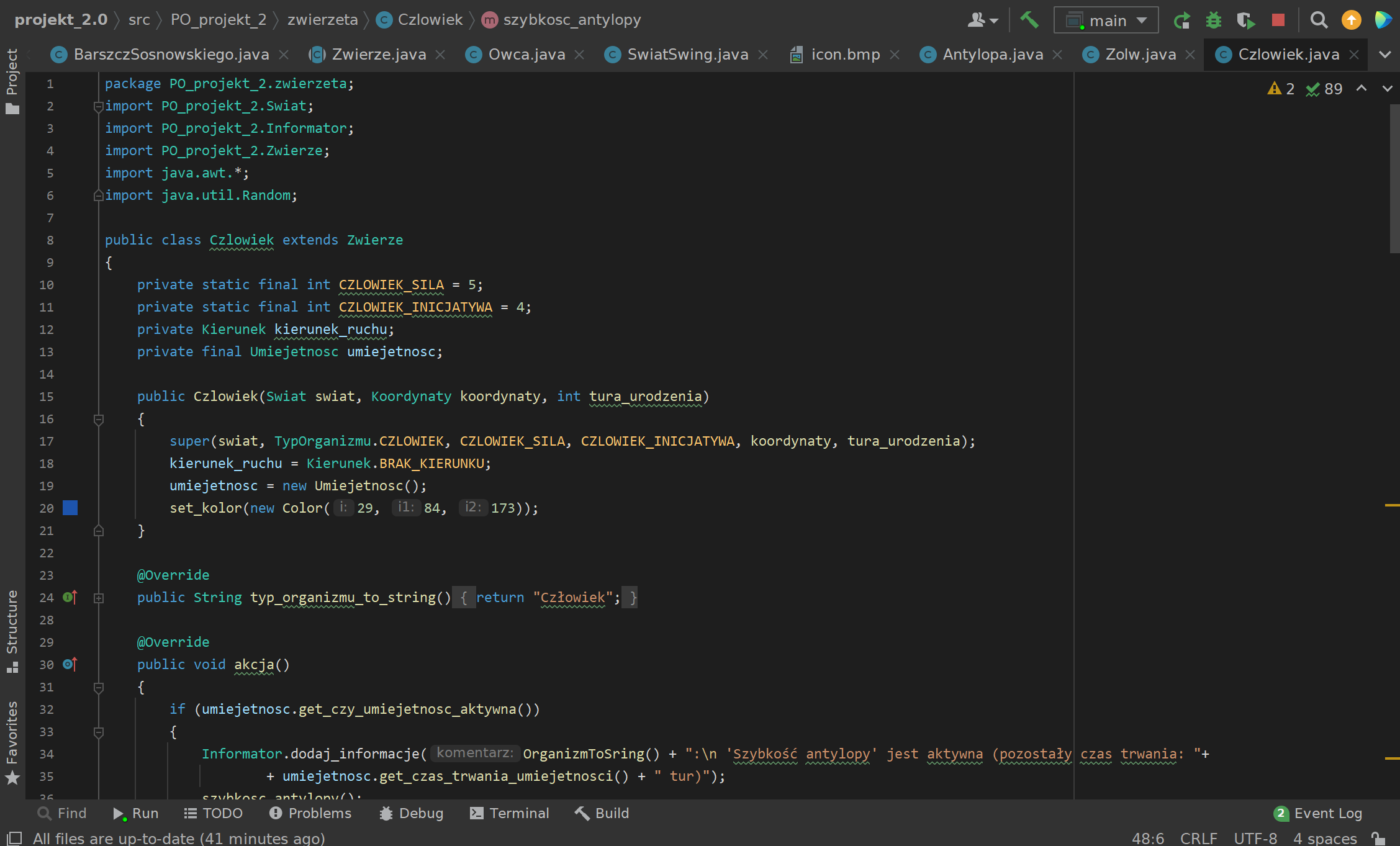


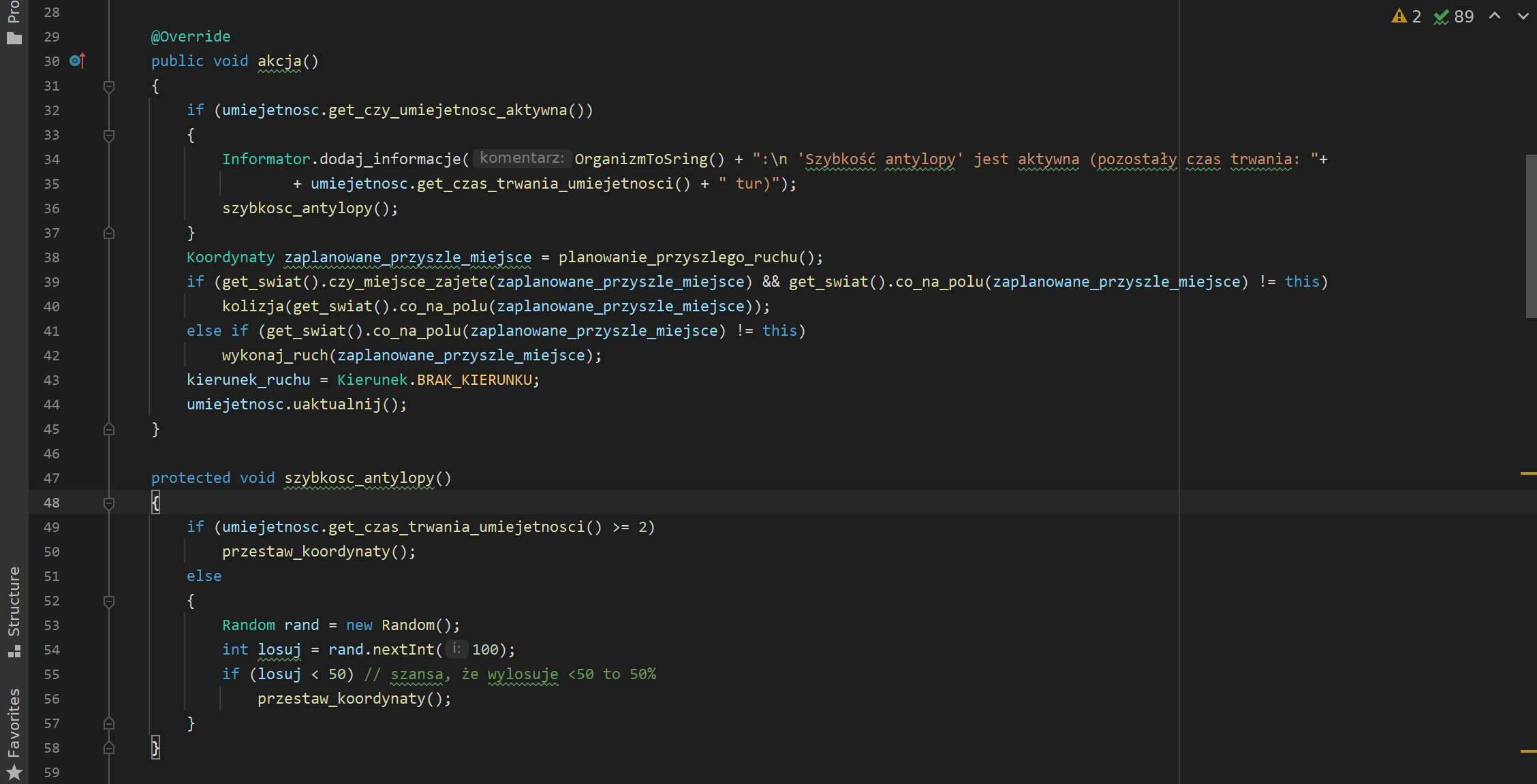




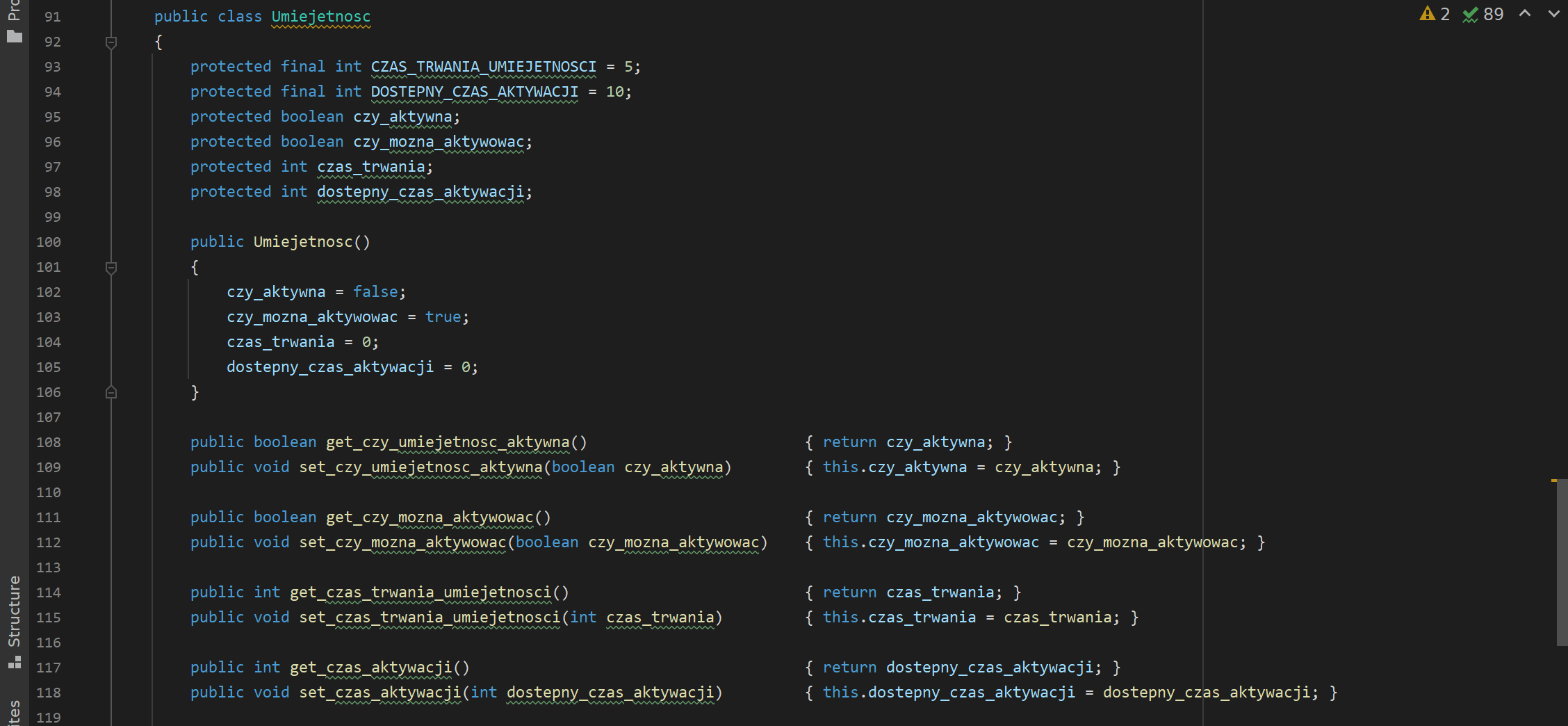
Stworzyłam również klasę Czlowiek, która dziedziczy po klasie Zwierze. Nie posiada on własnej inteligencji. Sterowany jest przez gracza strzałkami na klawiaturze, prawo, lewo, góra, dół. Posiada on umiejętność, Umiejetnosc jako klasa wewnątrz klasy Czlowiek.

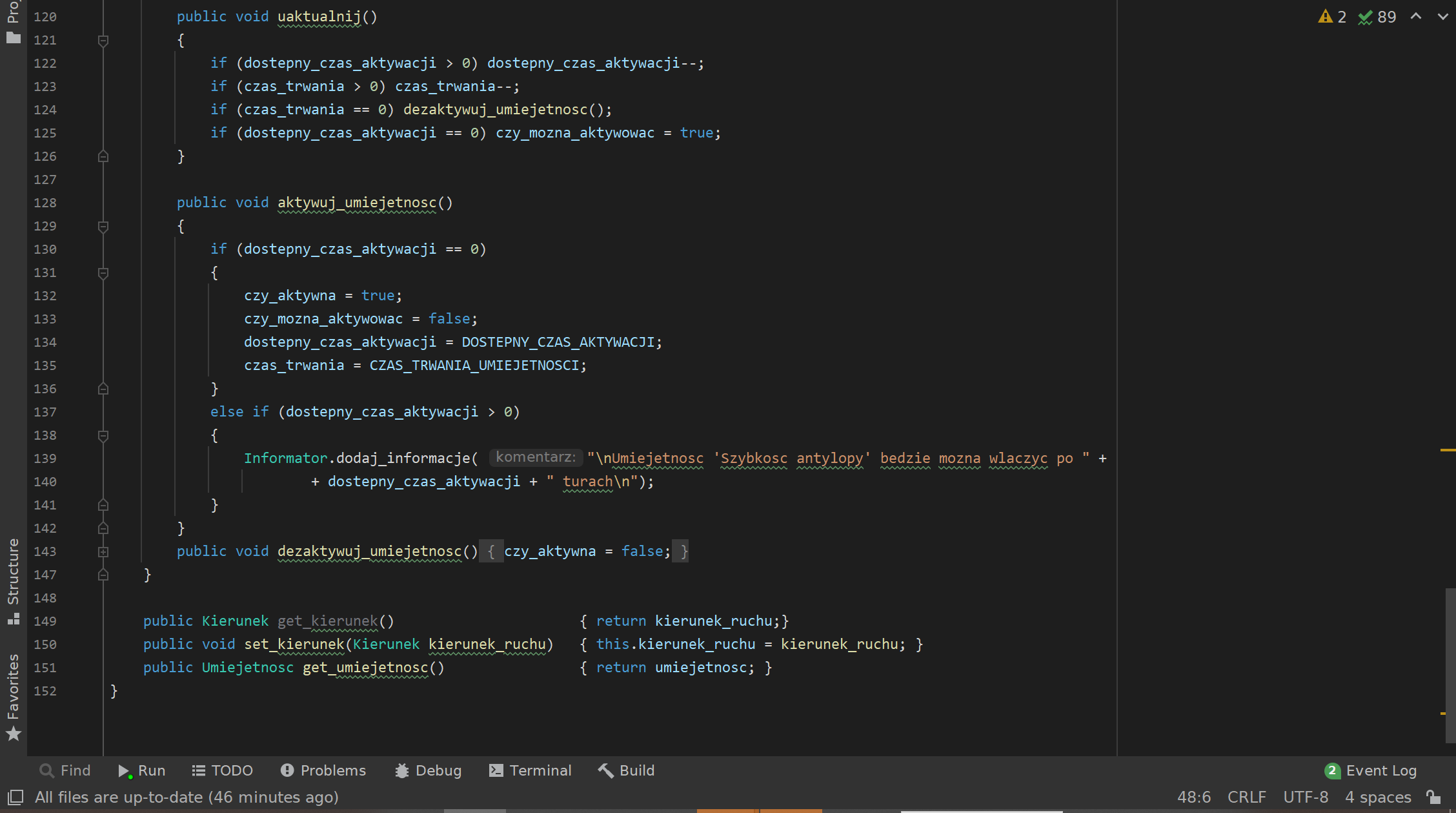
Człowiek jest tylko jeden i się nie rozmnaża. Jego specjalną umiejętnością jest szybkość antylopy (mój indeks: 184592, (ostatnia cyfra indeksu)mod 5 == 2, 2 to szybkość antylopy). Po włączeniu tej umiejętności klikając U, człowiek zyskuje umiejętność. Jej czas trwania wynosi 5 tur, a możliwość ponownej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











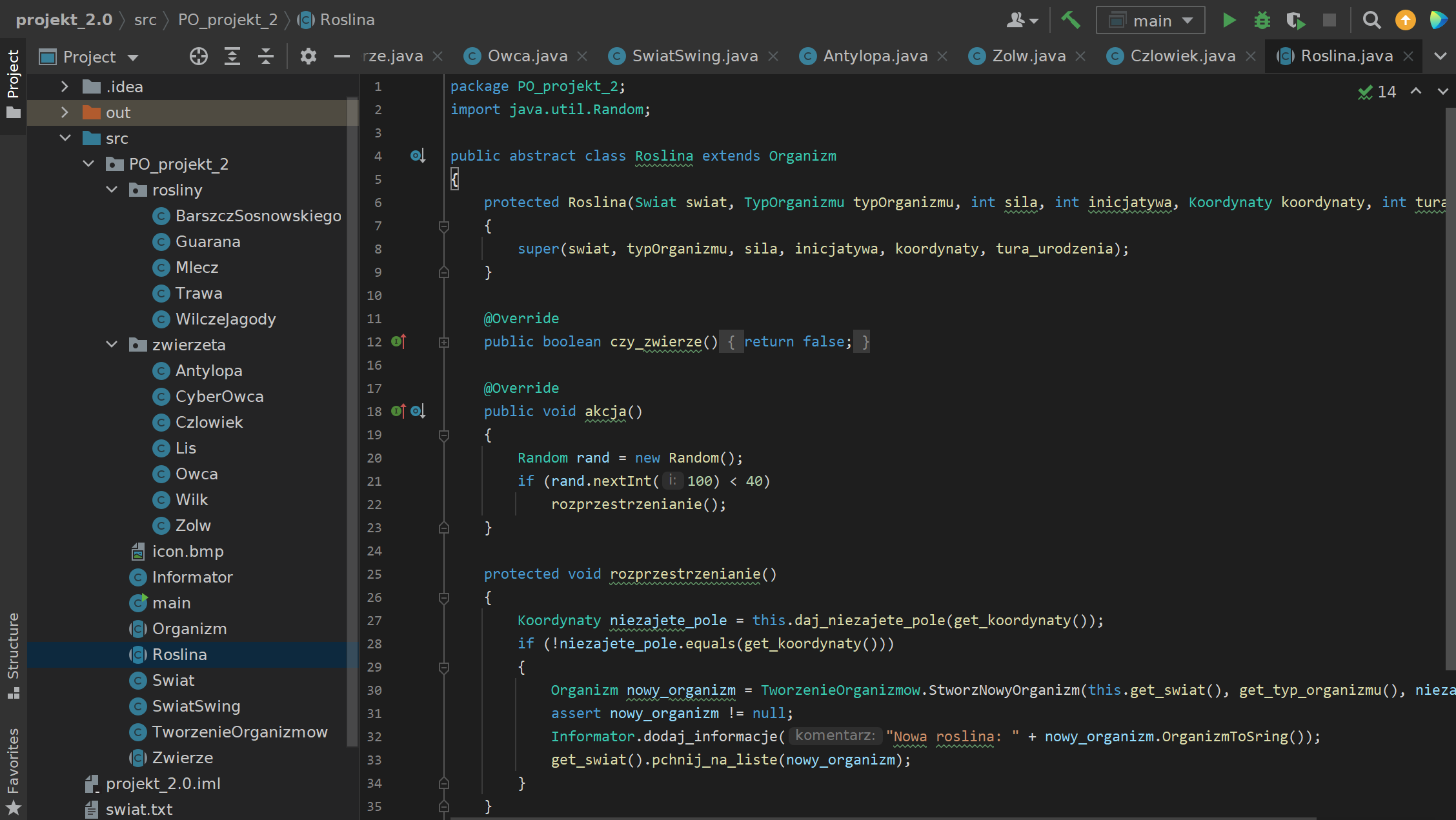
Zaimplementowałam 5 klas zwierząt, dziedziczących po Zwierze i Cyberowcę: Owca, Cyberowca, Wilk, Lis, Zolw i Antylopa.

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 Roslina: trawa, mlecz, guarana, wilcze jagody i barszcz Sosnowskiego. Roslina.java



Przykładowa roślina: 