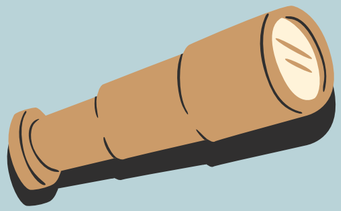
Grupa 17

GRA W STATKI

AGENDA

- Koncepcja gry
- Architektura
- Funckje i polimofrizm



KONCEPCJA GRY

Czym są “Statki” ?

Rozpoczynasz grę ustawiając lub automatycznie system ustawia twoją plansze z statkami na 10x10

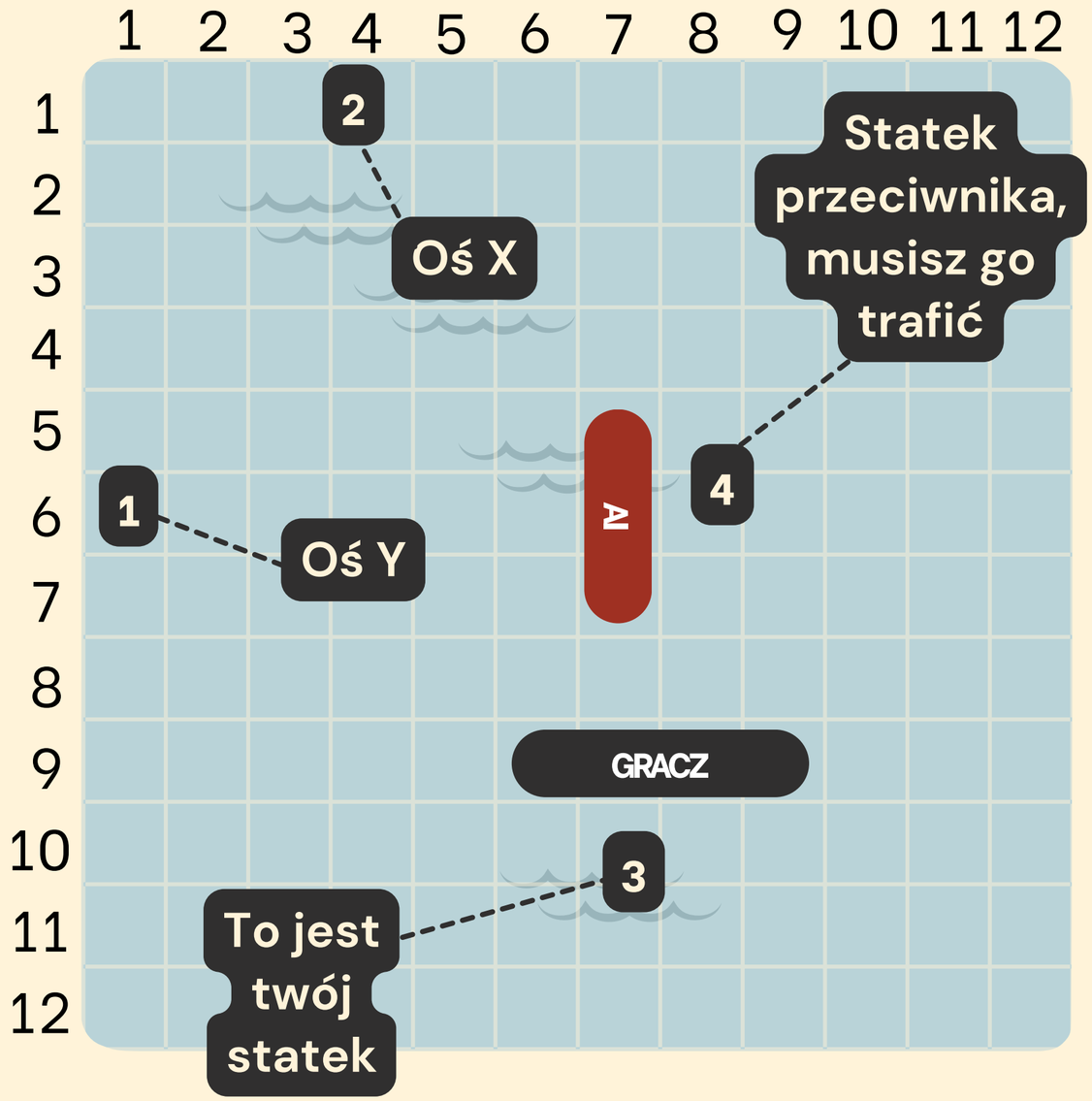
Twoim celem jest zatopić wszystkie statki, które przeciwnik ma rozstawione np. Destroyer posiada 2 pola

Strzeż się, ponieważ AI jest podchwytliwy, a różne wydarzenia sprawiają, że gra jest ciekawsza!

RODZAJE STATKÓW

Przed rozpoczęciem gry musimy ustawić nasze statki. Najlepiej tak, żeby przeciwnik ich nigdy nie znalazł... czyli zupełnie odwrotnie niż my ustawiamy.

Pola	Nazwa	
1	Okręt podwodny	4x
2	Niszczyciel	3x
3	Krążownik	2x
4	Pancernik	1x



ZASADY GRY

W trakcie gry będziesz musiał wybierać pola, które chcesz „zatopić” — wpisując współrzędne X (kolumna) i Y (wiersz).

💥 Jeśli trafisz statek — gratulacje, celny strzał!
💧 Jeśli nie — niestety pudło.

🔹 Tura gracza:
- Podajesz współrzędne, np. 5 3
- Program informuje: **Trafiony / Pudło**
- Plansza się aktualizuje

🔹 Tura AI:
- AI... no cóż... myśli (czasem aż za dobrze) i też strzela
- Jeśli trafi — powtarza ruch, aż zatopi statek

ARCHITEKTURA

Jak działa projekt ?

```java
public class Main {  Matthias3721 +1 *
    private static final Logger LOGGER = Logger.getLogger(Main.class.getName());

    public static void main(String[] args) {  Matthias3721 +1 *
        LOGGER.info( msg: "Starting Battleship game");

        Scanner scanner = new Scanner(System.in);
        System.out.print("Podaj swój login: ");
        String username = scanner.nextLine();

        MainMenu.show(username);
    }
}
```

1

```java
public class MainMenu {  1 usage  new *
    public static void show(String username) {  1 usage  new *
        Scanner scanner = new Scanner(System.in);
        while (true) {
            System.out.println("\n== MENU GŁÓWNE ==");
            System.out.println("[1] Rozpocznij grę");
            System.out.println("[2] Zobacz statystyki");
            System.out.println("[3] Twórcy gry");
            System.out.println("[4] Wyjście");
            System.out.print("Wybierz opcję: ");

            String input = scanner.nextLine();

            switch (input) {
                case "1":
                    Game game = new Game(username);
                    game.start();
                    break;
                case "2":
                    GameStats stats = StatsLoader.loadStats(username);
                    System.out.println(stats != null ? stats : "Brak statystyk dla użytkownika.");
                    break;
                case "3":
                    System.out.println("Twórcy: Oskar Świątek, Bartek Szołdrowski, Mateusz Więcek, 2025");
                    break;
                case "4":
                    System.out.println("Do zobaczenia, " + username + "!");
                    return;
                default:
                    System.out.println("Nieprawidłowy wybór. Spróbuj ponownie.");
            }
        }
    }
}
```

2

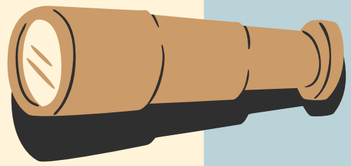
3

```java
    public void start() {  1 usage  Oskar Świątek +2 *
        Scanner scanner = new Scanner(System.in);
        int choice = -1;
        System.out.println("Welcome to Battleship!\n");

        while (true) {
            System.out.println("Wybierz jak chcesz generować plansze");
            System.out.println("[0] Automatyczne generowanie\n[1] Manualne generowanie");

            if (scanner.hasNextInt()) {
                choice = scanner.nextInt();
                if (choice == 0 || choice == 1) {
                    break;
```

OBSŁUGA WYJĄTKÓW

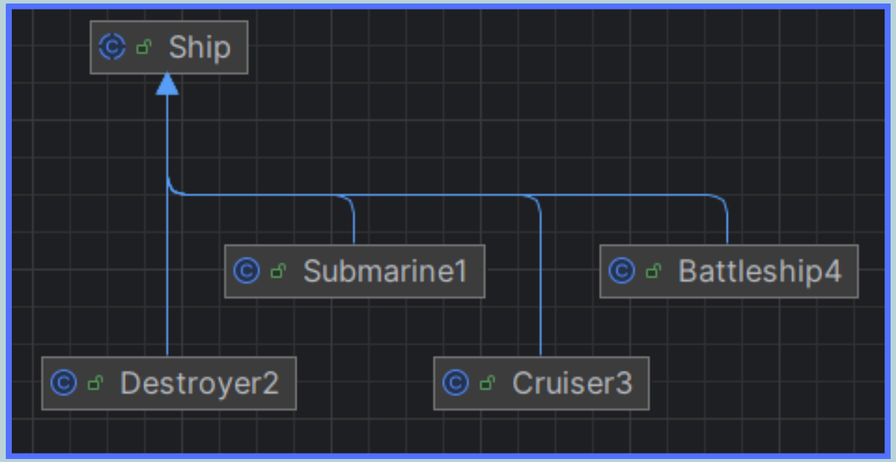


Podczas gry mogą zdarzyć się nieprawidłowe sytuacje — np. gracz wpisze współrzędne poza planszą albo strzeli drugi raz w to samo pole.

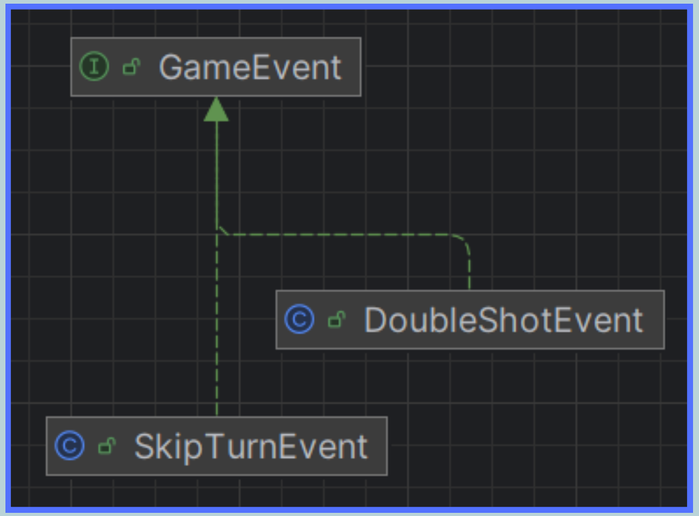
```java
// Glowny konstruktur z walidacja
public Coordinate(int x, int y) throws InvalidPositionException {  Matthias3721 +1
    if (x < 1 || x > 10 || y < 1 || y > 10)
        throw new InvalidPositionException("Out of bounds: " + x + "," + y);
    this.x = x;
    this.y = y;
}
```

POLIMORFIZM

Polimorfizm to cecha programowania obiektowego, która pozwala używać różnych klas w ten sam sposób.



*Dla Ship



*Dla Events

EVENTY

W trakcie wojny nigdy nie wiadomo, co się wydarzy..

SkipTurnEvent

Gracz lub AI traci turę – zupełnie jakby zagapił się przy lunecie!

```
[EVENT] Burza magnetyczna: AI traci turę!
Your's turn.
Enter X:
```

DoubleShotEvent

Gracz lub AI strzela dwa razy w tej rundzie – czas na ofensywę!

```
[EVENT] Podwójny strzał: Gracz odda dwa strzały!
Your's turn.
Enter X:
```

PRZECHOWYWANIE DANYCH

```xml
<?xml version="1.0" encoding="UTF-8"?>
<java version="21.0.7" class="java.beans.XMLDecoder">
 <object class="pl.battleship.stats.GameStats">
  <void property="hits">
   <int>19</int>
  </void>
  <void property="shotsFired">
   <int>96</int>
  </void>
  <void property="username">
   <string>test</string>
  </void>
 </object>
</java>
```

Statystyki

W naszym projekcie istnieje przechowywanie informacje gracza o ilości strzałów, wygranach itp.

Dzięki zastosowaniu formatu **XML** możemy w prosty sposób zapisywać i przechowywać statystyki gracza, takie jak liczba strzałów, trafień czy wynik rozgrywki.

DZIĘKUJĘ!

Zapraszam do gry

Twórcy

Oskar Świątek

Bartłomiej Szołdrowski

Mateusz Więcek

DOSTĘPNE W GITHUB

Pobierz już teraz:

GitHub

Kliknij przycisk **Pobierz**,
aby przejść na stronę.