
# Wprowadzenie do Javy

## Część I

Michał Idzik
AGH IEiT Informatyka

# Czym jest Java?

# Dlaczego Java?

# Java

- Język wysokiego poziomu
- Programowanie zorientowane obiektowo
- Prosta składnia
- Przenośność
- Szeroki zestaw narzędzi w bibliotece standardowej

# *“Od dębu do kawy na abonament”*

## Krótka historia Javy

**1991**
Sun tworzy projekt *Green*.
Jeden kod, wiele urządzeń - nowy język programowania: *Oak*

**1996**
Java 1.0 dostępna publicznie, wsparcie w przeglądarkach

**1998**
Java 2

**2009**
Firma *Oracle* wykupuje *Sun*

**2011**
Java 7

**2014**
**Java 8 (LTS)**

**2017**
Java 9

**2018**
Java 10, **Java 11 (LTS)**

**2019**
Java 12, Java 13

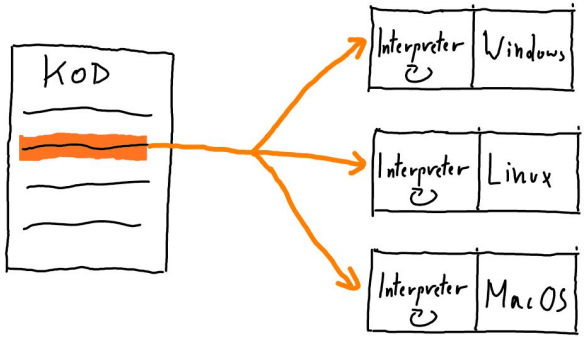
- Nowy, 6-miesięczny cykl wydawniczy
- Oracle JDK płatne w zastosowaniach komercyjnych
- Darmowe OpenJDK jako alternatywa, ale bez darmowego *Long Term Support*

# Java
# Architektura

# Kompilatory i interpretery

## Interpreter

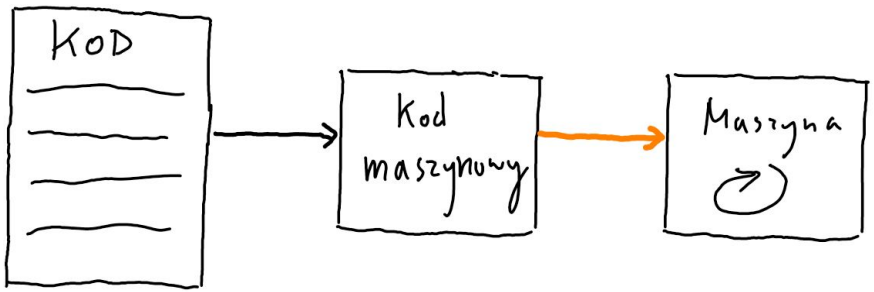
- Analizuje kod źródłowy
- **Wykonuje kod bezpośrednio** w trakcie działania
- Wolny, duży koszt zarządzania zasobami
- Duża elastyczność, wygodny do testów i procesu debugowania



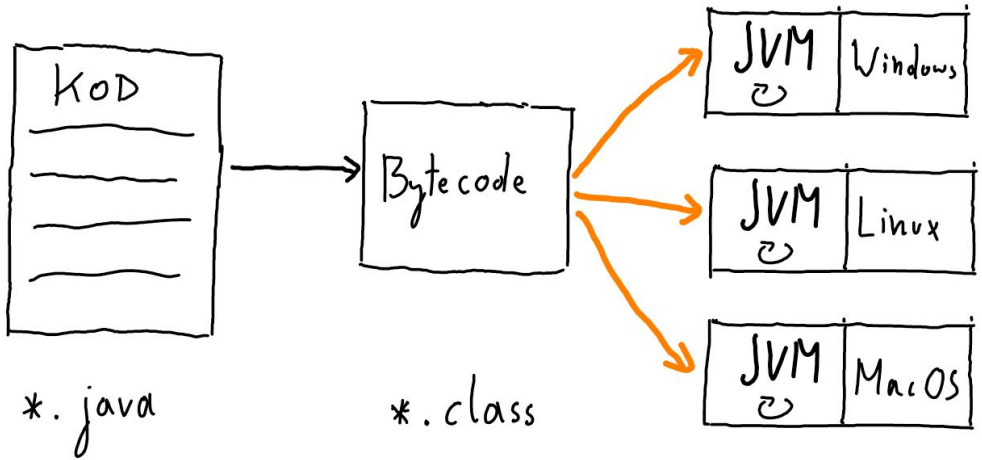
# Kompilatory i interpretery

## Kompilator

- Tłumaczy kod do postaci zapisanej w innym języku (najczęściej: kod maszynowy)
- **Nie wykonuje** kodu źródłowego
- Może dokonywać optymalizacji



# JVM - Java Virtual Machine




- **Kompilacja** do kodu pośredniego (*bytecode*)
- **Interpretacja** *bytecode* w emulatorze maszyny wirtualnej Javy (JVM)

# JVM - Java Virtual Machine

- Przenośność: *“Write Once, Run Everywhere”*
- Optymalizacje: JIT
- Inne języki mogą być wykonywane na JVM:
  - Scala
  - Clojure
  - Kotlin
  - Jython (Python), JRuby (Ruby), itp.

# Java Development Kit (JDK)

javac : plik**.java** → plik**.class** (kompilacja)

java : plik**.class** → JVM (interpretacja)

plik.java

```
public class HelloJava {


    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

# Java
# Składnia

# Z czego składa się program?

“Algorytmy + Struktury Danych = Programy”

- N. Wirth

Instrukcje sterujące

Dane (zmienne, stałe)
o różnych typach

# Typy danych w Javie

Wbudowane **typy proste (prymitywne)**:

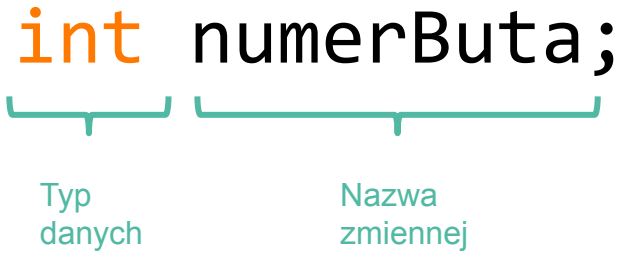
- int - liczba całkowita
- long - liczba całkowita (większy zakres)
- double - liczba zmiennoprzecinkowa
- boolean - wartość logiczna
- char - pojedynczy znak

```
12 -10 0

0L 9999999999L

3.14 1.0 2e10

true false

'a' 'z' 'ę'
```

# Typy danych w Javie

## Zmienna

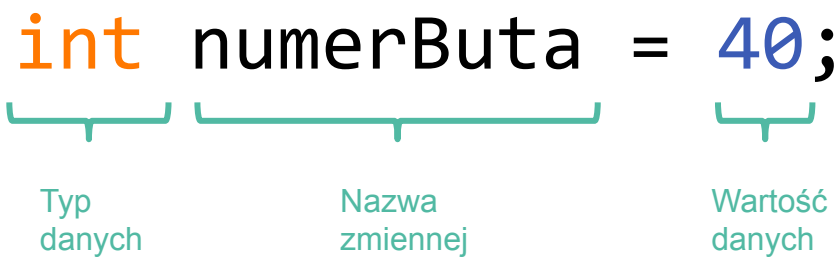
- Reprezentuje dane określonego typu



# Typy danych w Javie

## Zmienna

- Reprezentuje dane określonego typu
- Przechowuje wartość danych



# Typy danych w Javie

## Zmienna

- Reprezentuje dane określonego typu
- Przechowuje wartość danych
- Wartość może się zmieniać

```
int numerButa = 40;

numerButa = 41;
```

# Typy danych w Javie


Co można zrobić na zmiennej typu prostego?

- Przypisać wartość

```
int numer = 149;
```

*“Od teraz zmienna* **numer** *ma wartość 149”*

**UWAGA**: Znak równości nie oznacza porównania!

# Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość
- Skopiować do niej wartość innej zmiennej

```
double twojWynik = 3.14;

double mojWynik = twojWynik;
```

*“Od teraz zmienna* **mojWynik** *ma wartość 3.14”*

# Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość
- Skopiować do niej wartość innej zmiennej
- Użyć na niej operatora (np. **arytmetycznego**)

```
int rok = 2016;

rok = rok + 1;
```

*“Od teraz zmienna* **rok** *ma wartość 2017”*

# Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość
- Skopiować do niej wartość innej zmiennej
- Użyć na niej operatora (np. arytmetycznego, **logicznego**)

```
int wagaZiemniakow = 2;

wagaZiemniakow > 1
```

*“Czy ziemniaki ważą więcej niż 1kg?”*

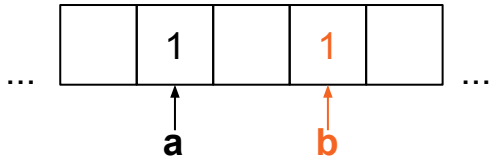
```
wagaZiemniakow == 2
```

*“Czy ziemniaki ważą dokładnie 2kg?”*

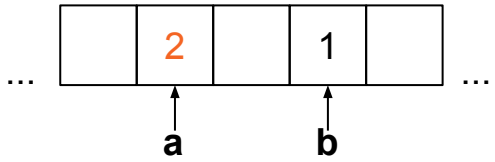
# Typy danych w Javie

```
int a = 1;
```


```
int b = a;
```




```
a = a + 1;
```




W zmiennych typów prostych przypisujemy **wartości**.

# Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość
- Skopiować do niej wartość innej zmiennej
- Użyć na niej operatora (np. arytmetycznego)
- Użyć jako argumentu funkcji (metody)


```
int kwota = 1000;

System.out.println(kwota);
```

# Typy danych w Javie

- Java to język **statycznie typowany**!
- Każda zmienna musi explicite deklarować swój typ
- Kompilator sprawdza poprawność operacji pod kątem typu danych

**Python** (dynamiczne typowanie):

```
liczba = 100
```

**Java** (statyczne typowanie):

```
int liczba = 100;
```

# Typy danych w Javie

✔ `int a = 40;`

✘ `int b = 2.5;` → `int b = (int)2.5;`

✔ `double c = 3;` (bezstratne rzutowanie)

✘ `boolean d = 1;` → `boolean d = true;`

✔ `boolean e = a == 40;` (operator == zwraca **true** lub **false**)

# Typy danych w Javie

Typy złożone = **Klasy**

- składają się z zestawu atrybutów i funkcji (metod)
- pozwalają definiować własne typy danych
- wprowadzają programowanie obiektowe!

- ...program napisany w Javie składa się z zestawu wzajemnie powiązanych klas

# Typy danych w Javie

**Klasy** (przykłady):

- `String`
- `System`
- `HelloJava`

  …
- `Samochód`
- `Instrument`

# Typy danych w Javie

Klasa `String`

- Reprezentuje łańcuch znaków (napis)

```
String inwokacja = "Litwo! Ojczyzno moja!";
```



*Wartość napisu musi być umieszczona między " "*

# Typy danych w Javie

Klasa `String`

- Reprezentuje łańcuch znaków (napis)
- Oferuje zestaw operacji (metod) na napisach

```
String inwokacja = "Litwo! Ojczyzno moja!";
int rozmiar = inwokacja.length();
```

*Wywołanie operacji (metody) oznaczamy kropką*

# Typy danych w Javie

Klasa `String`

- Reprezentuje łańcuch znaków (napis)
- Oferuje zestaw operacji (metod) na napisach

```
String inwokacja = "Litwo! Ojczyzno moja!";
int rozmiar = inwokacja.length();
String kraj = inwokacja.substring(0, 5);
```

# Typy danych w Javie

Specjalny typ złożony : **Tablica**

- Ciąg elementów danego typu

```
int[] wynikiLotto;
```

*Tablica wartości typu całkowitego*

# Typy danych w Javie

Specjalny typ złożony : **Tablica**

- Ciąg elementów danego typu
- Posiada określony rozmiar

```
int[] wynikiLotto = new int[6];
```

*Tablica wartości typu całkowitego*

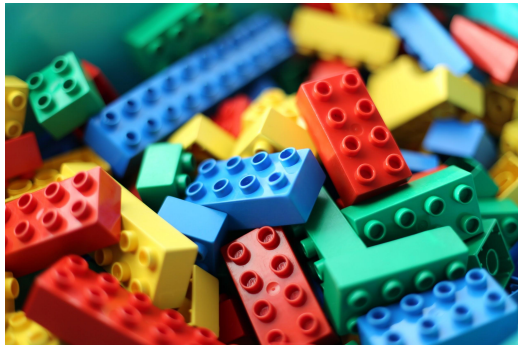
# Typy danych w Javie

Zmienna →

Tablica →

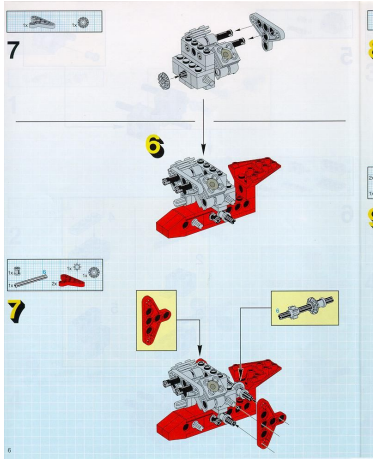
# Instrukcje sterujące

- Definicje zmiennych
- Operatory
- Wywołania funkcji/metod
- Instrukcje warunkowe
- Pętle



Dane różnych typów

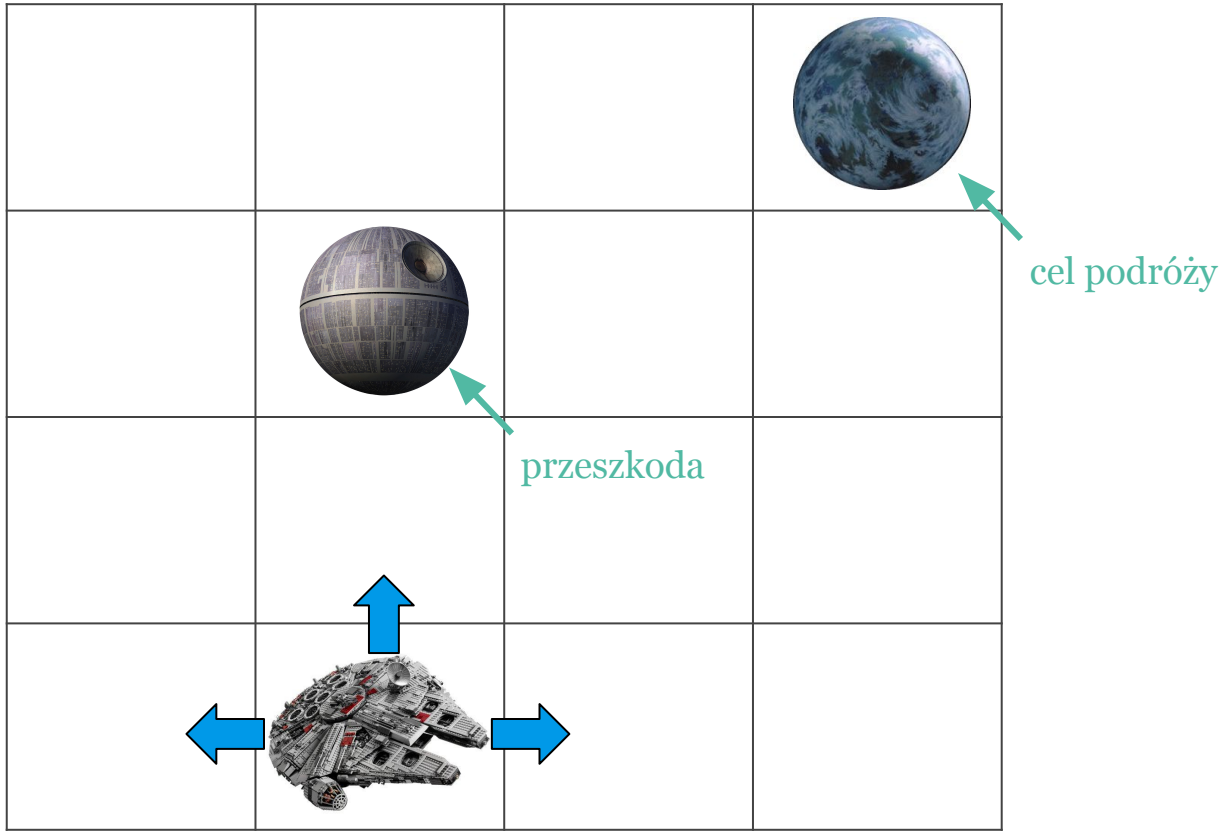
+




**Instrukcje sterujące**

→


Program


cel podróży
przeszkoda

# Definicje zmiennych

```
int paliwo = 300;
```

*Definicja (stworzenie) zmiennej z przypisaniem wartości*

```
paliwo = 200;
```

*Przypisanie wartości do istniejącej zmiennej*

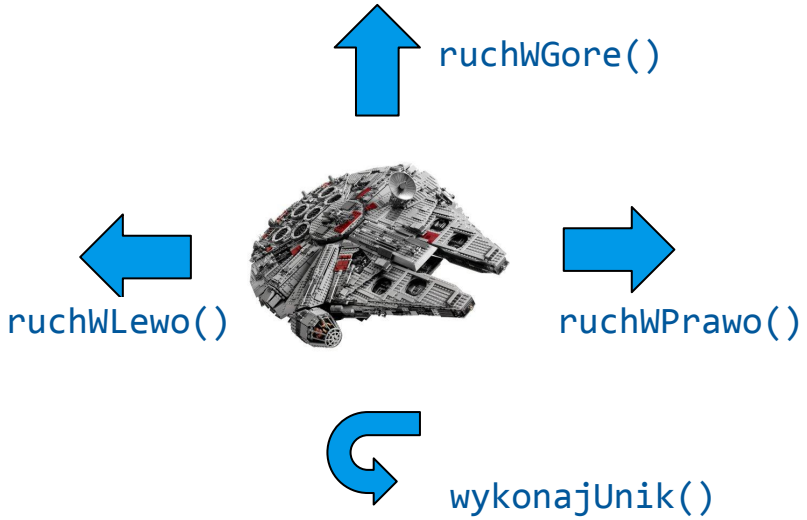
# Definicje zmiennych

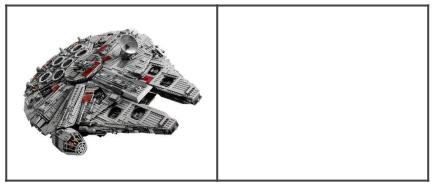
```
Statek sokolMillenium = new Statek();

Statek gwiazdaSmierci = new Statek();
```

*Definicje zmiennych klasy (typu złożonego)* `Statek`

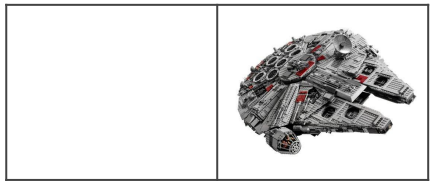
# Wywołania funkcji/metod

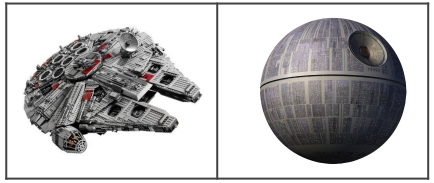
```
sokolMillenium.ruchWGore();

sokolMillenium.ruchWLewo();

sokolMillenium.ruchWPrawo();

sokolMillenium.wykonajUnik();

sokolMillenium.wrogWPoblizu();
```



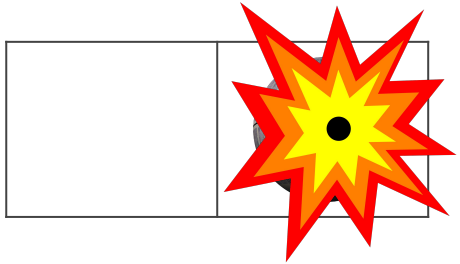


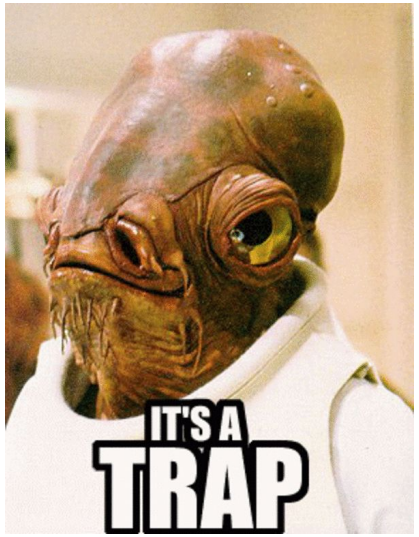
```
sokolMillenium.ruchWPrawo();
```



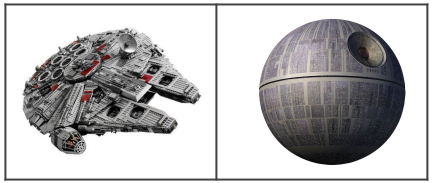


```
sokolMillenium.ruchWPrawo();
```

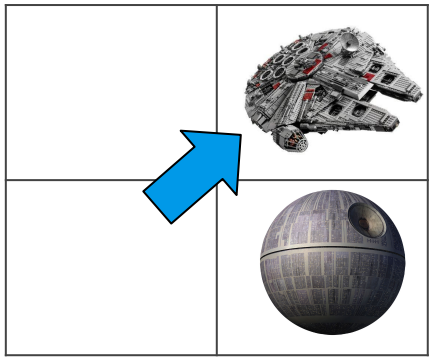




# Instrukcje warunkowe



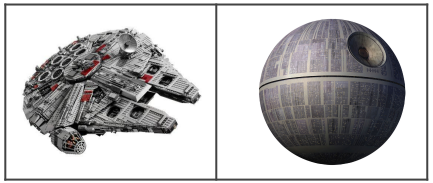
```
if (sokolMillenium.wrogWPoblizu()) {

    sokolMillenium.wykonajUnik();

}
```

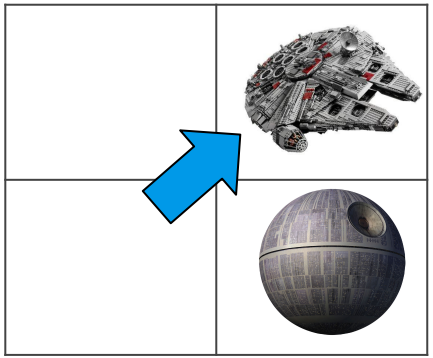


# Instrukcje warunkowe

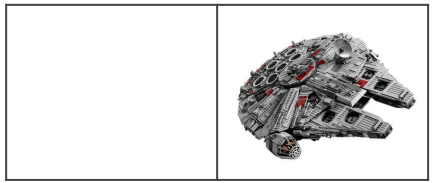
```
if (sokolMillenium.wrogWPoblizu()) {
    sokolMillenium.wykonajUnik();
} else {
    sokolMillenium.ruchWPrawo();
}
```

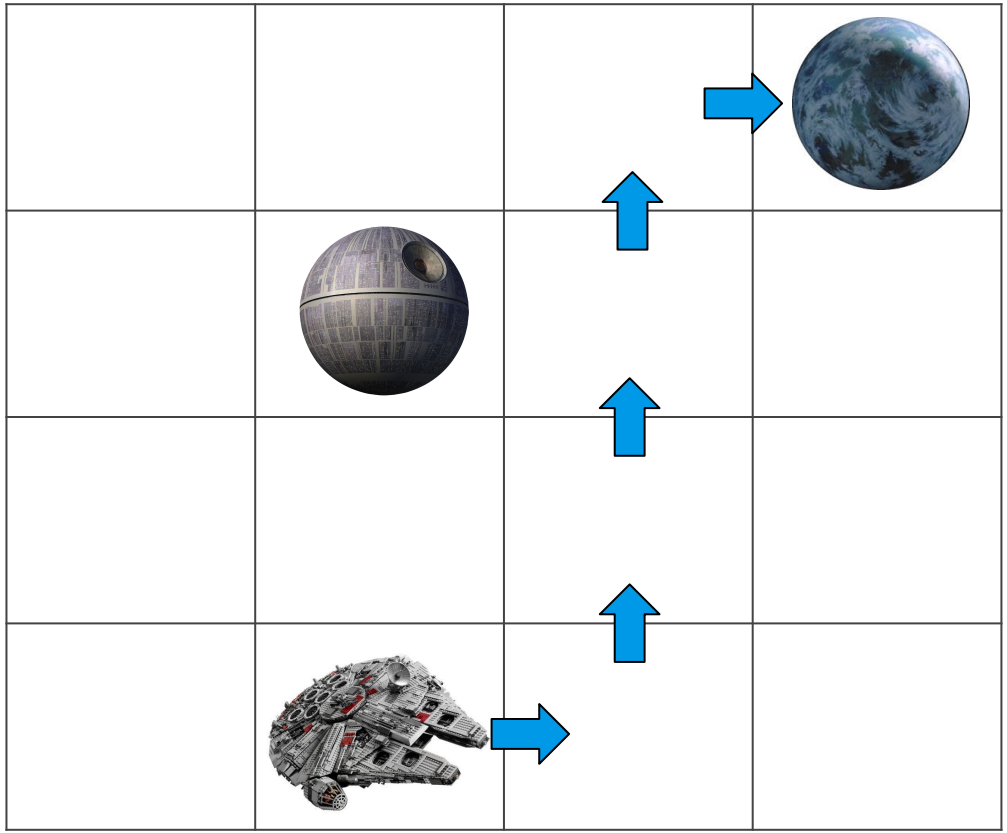
if





else





```
sokolMillenium.ruchWPrawo();

sokolMillenium.ruchWGore();
sokolMillenium.ruchWGore();
sokolMillenium.ruchWGore();

sokolMillenium.ruchWPrawo();
```

# Co jeśli droga w górę prowadzi przez całą galaktykę?

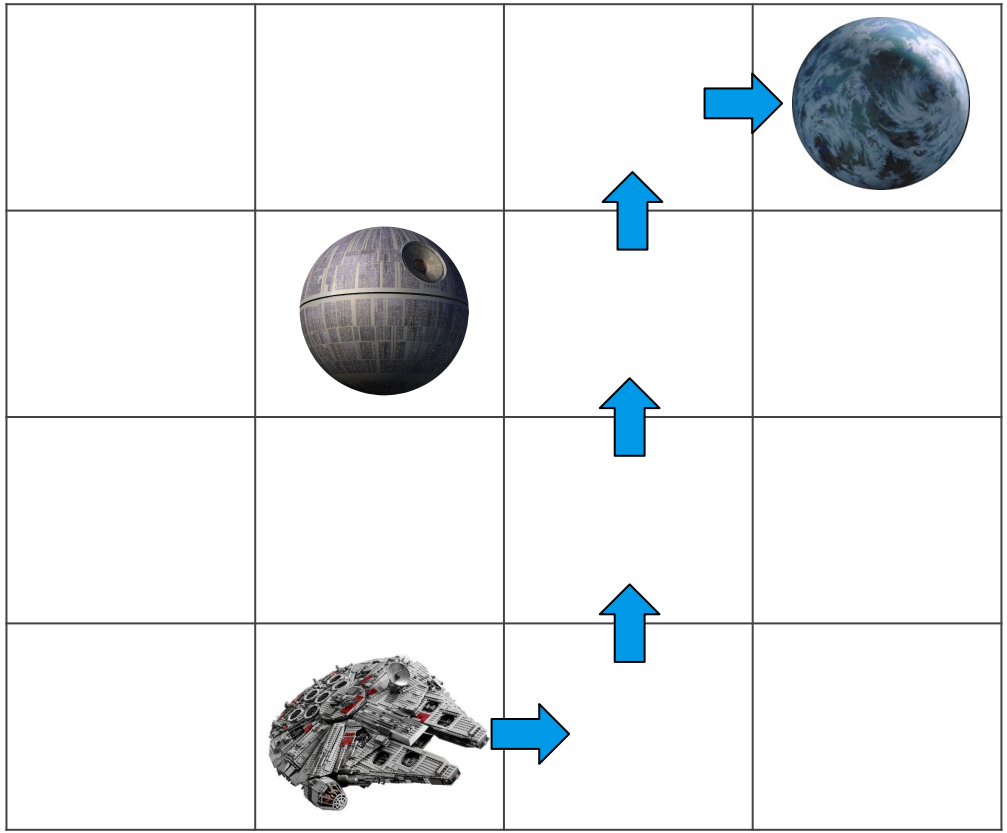
Napiszemy `sokolMillenium.ruchWGore()` 100 razy? 10000 razy?

# Pętle

```
for (int krok = 1; krok <= 100; krok++) {
    sokolMillenium.ruchWGore();
}
```

# Pętle

```
int krok = 1;

while (krok <= 1000) {
    sokolMillenium.ruchWGore();
    krok += 1;
}
```



```
sokolMillenium.ruchWPrawo();

for (int krok = 1; krok <= 3; krok++) {
    sokolMillenium.ruchWGore();
}

sokolMillenium.ruchWPrawo();
```