



Podstawy programowania w Javie

Trener: Krzysztof Pawłowski

Gdańsk, 16 grudnia 2017 roku

# Java - wprowadzenie

- Aktualna wersja to Java 9
- Aplikacje pisane w Javie są kompilowane do bytecode, a następnie uruchamiane na maszynie wirtualnej (JVM – Java Virtual Machine)
- Jest to język obiektowy
- Automatyczne ładowanie klas, automatyczne zarządzanie pamięcią
- Aplikacja może być uruchomiona na dowolnej maszynie, na której jest dostępna JVM (Linux, Solaris, Windows, Mac, itd.)

# Java - wprowadzenie

- Dystrybucje Javy
  - Java SE – Java Standard Edition – podstawowa dystrybucja Javy
  - Java EE – Java Enterprise Edition – do wytwarzania aplikacji webowych
  - Java ME – Java Micro Edition – do wytwarzania aplikacji na urządzenia mobilne, aktualnie ma coraz mniejsze znaczenie

# Java - składnia

*nazwa pliku HelloWorld.java*

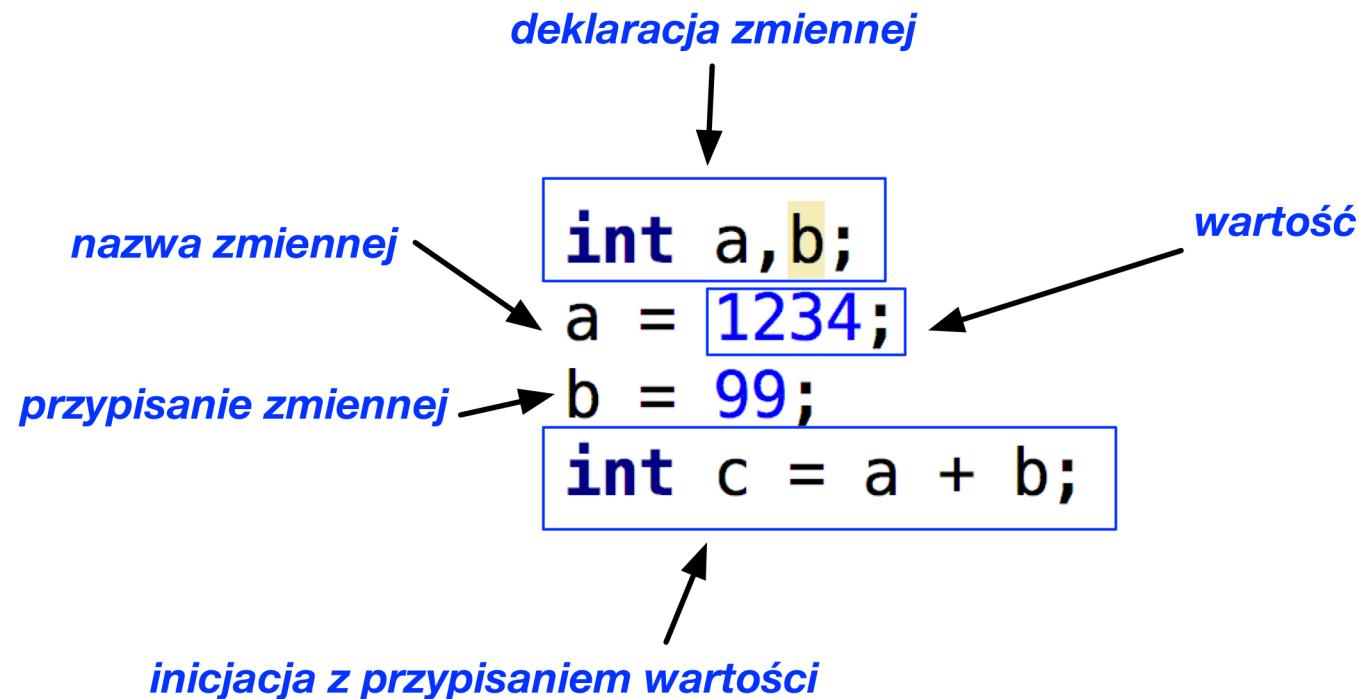
```
public class HelloWorld {  
    public static void main(String... args) {  
        System.out.println("Hello, world");  
    }  
}
```

*nazwa*

*główna metoda*

*ciało metody*

# Java - składnia



# Java - składnia

*wyrażenie warunkowe*

```
if (x > y){   gdy niespełniony powyższy warunek
    t = 0;
} else if (x < y){
    t = 1;
} else {          gdy żaden warunek nie jest spełniony
    t = -1;
}
```

# Java - składnia

```
warunek kontynuacji pętli
↓
while (x < t){
    x = 2*x;
}
      ↑
      ciało pętli
```

# Java - składnia

*deklaracja i inicjacja zmiennej sterującej pętlą*

*warunek kontynuacji pętli*

*inkrementacja*

*cało pętli*

```
int b = 1;
for (int i = 0; i < t; i++) {
    System.out.println(i + " " + b);
    b = 2 * b;
}
```

# Java - składnia

```
do {  
    x = 2.0 * Math.random() - 1.0;  
    y = 2.0 * Math.random() - 1.0;  
} while (Math.sqrt(x * x + y * y) > 1);
```



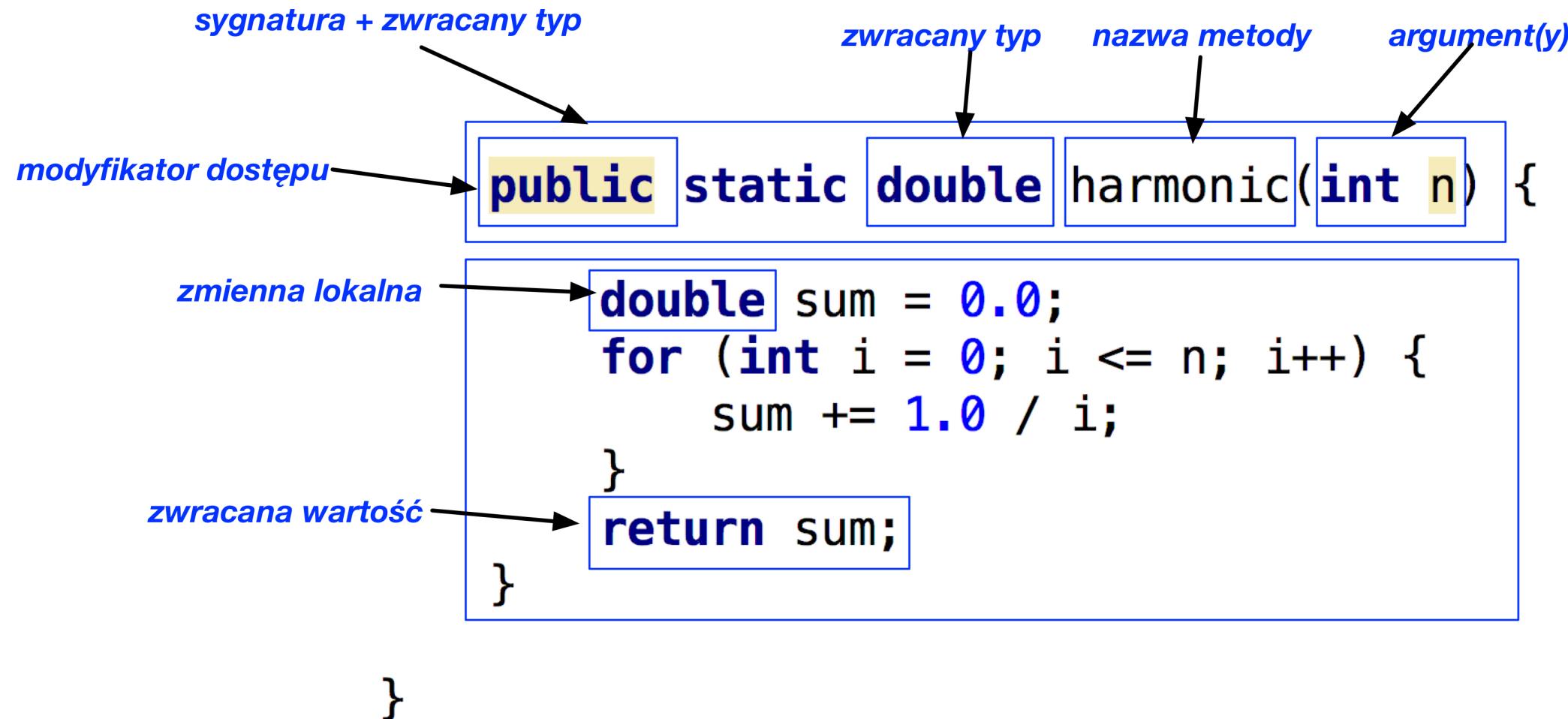
*warunek wyjścia z pętli*

# Java - składnia

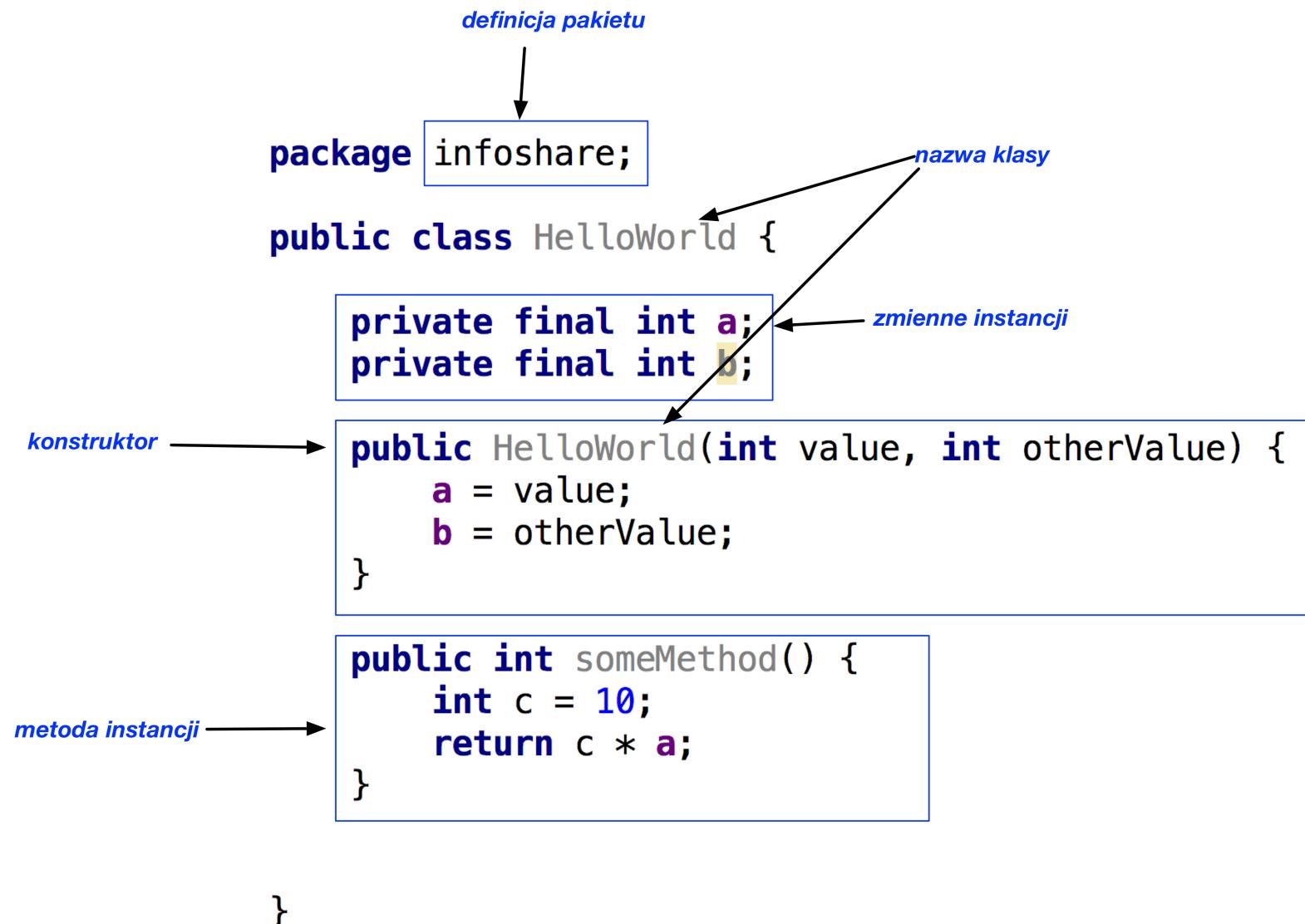
wartość, dla której sprawdzamy którą linię kodu wywołać

```
switch (t) {  
    case 0:  
        System.out.println(0);  
        break;  
    case 1:  
        System.out.println(1);  
        break;  
    default:  
        System.out.println("inny");  
}
```

# Java - składnia



# Java - składnia



# Zadanie wstępne

- Utworzenie i uruchomienie projektu java w IntelliJ z użyciem mavena i archetypu maven-archetype-quickstart.

# Zadanie wstępne - omówienie

- maven - narzędzie automatyzujące budowę oprogramowania na platformę Java
  - archetyp maven-archetype-quickstart tworzy podstawową aplikację
  - tworzy odpowiednią strukturę katalogów

# Zadanie wstępne - omówienie

- Klasa – podstawowy element składowy aplikacji
  - Definicja każdej klasy publicznej definiujemy w osobnym pliku o takiej samej nazwie jak klasa (klasy mogą być public, private, default, protected)
- Klasa to definicja pewnego rodzaju bytów, które posiadają zbiór metod i cech
  - Klasa to definicja, instancja klasy odzwierciedla konkretny byt
- Metoda public static void main() to główna metoda, od której rozpoczyna się uruchamianie programu przez JVM
  - Uruchamiając aplikację określamy z której klasy metodę main() chcemy wywołać

# Zadanie wstępne

- Metoda statyczna to taka metoda, którą możemy wywołać bezpośrednio na klasie, nie jest wymagane istnienie jej instancji
- Odczyt parametrów przekazanych do funkcji main() poprzez tablicę

# Zadanie 1

- Przekazanie argumentu do aplikacji
- Wyświetlenie argumentu na ekranie

## Zadanie 2

- napisz kod, który przypisze do zmiennej lokalnej "randomInt" typu int losową liczbę z przedziału 1-100 (java.util.Random)
- zrzutuj zmienną "randomInt" na zmienną "randomDouble" typu double
- zaokrąglij zmienną „randomDouble” korzystając z java.lang.Math, wynik zapisz w zmiennej „rounded” typu int
- porównaj zmienne "rounded" z "randomDouble", wypisz na ekran informację jeżeli są równe

## Zadanie 3

- Korzystając z kodu z poprzedniego zadania:
  - wydziel kod z metody main do metody o nazwie "checkPrecision"
  - wywołaj metodę checkPrecision w nieskończonej pętli
  - przy każdej iteracji wstrzymaj aplikację na 1 sekundę (Thread.sleep())
  - ustaw breakpoint na wywołaniu metody "checkPrecision"- uruchom program w trybie debug

## Zadanie 4

- Liczba doskonała – liczba naturalna, która jest sumą wszystkich swych dzielników właściwych. Najmniejszą liczbą doskonałą jest 6, ponieważ  $6 = 3 + 2 + 1$ . Następną jest 28 ( $28 = 14 + 7 + 4 + 2 + 1$ ),
- Dla zadanego N znaleźć wszystkie liczby doskonałe mniejsze niż N.

## Zadanie 5

- Napisz program, który dla zadanego N policzy ciąg Fibonacciego z ostatnim elementem mniejszym lub równym N
  - Np dla N = 12, wynikiem będzie: 1,1,2,3,5,8

## Zadanie 6

- Napisz program, który wypełni tablicę napisów imionami podanymi z klawiatury i wypisze:
  - najdłuższe imię
  - najkrótsze imię
  - \*powtarzające się imiona wraz z ich krotnościami

## Zadanie 7

- Zaimplementuj klasę Ułamek, umożliwiającą prostą obsługę ułamków zadanych jako para liczb całkowitych licznik i mianownik.
  - Pola tej klasy mogą być następujące: licznik i mianownik typu integer, wartość typu double, czescCalkowita typu int.
  - Metody powinny być następujące:
    - void pomnoz(Ułamek u)
    - void dodaj(Ułamek u) - podobnie,
    - void wyswietl() - wyświetla ułamek w postaci [licznik / mianownik]

# Odczyt danych z pliku

- Można odczytać wszystkie linie za pomocą metody `Files.readAllLines(path)`
- W efekcie powstaje lista linii (Listy dokładniej będą przedstawione później)
- Chcemy każdą z linii wyświetlić na ekranie, korzystając z funkcji `foreach()`

## Zadanie 8

- Odczytaj linie z pliku
- Wyświetl linie z pliku

# Parsowanie tekstu - split

- String split dzieli ciąg znaków na tablicę elementów
- Tablica to struktura danych przechowująca sekwencję zmiennych pewnego typu
  - String[] elements = new String[5]
  - int[] numbers = new int[16]

## Zadanie 9

- Wyświetlenie z pliku tylko pierwszej kolumny

# Parsowanie tekstu – Regular expressions

- klasa Pattern – skompilowana reprezentacja wyrażenia regularnego
- klasa Matcher – obiekt, który interpretuje tekst wejściowy poprzez pattern

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

<https://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html>

## Zadanie 10

- Dany jest tekst: „ala ma kota i kot ma ale aLe KoTa nie widziano.”
  - napisz kod, który zwróci pozycję początkową i końcową dla każdego wystąpienia słowa "ma" w tekście
  - napisz kod, który zwróci pozycję początkową i końcową dla każdego wystąpienia słowa "ala" lub "kot" w tekście bez uwzględnienia wielkości znaków.

## Zadanie 11

- Wykorzystując grupowanie z klasy Matcher, stwórz wzorzec, który wyciągnie z daty w formacie DD/MM/YYYY dzień, miesiąc oraz rok

# Typy danych

- java jest językiem silnie typowanym, każda wartość ma określony typ
- istnieją dwa rodzaje typów
  - Proste
  - Obiektowe

# Typy danych – typy proste

- char – pojedynczy znak
- boolean – typ logiczny
- Typy numeryczne
  - byte
  - short
  - int
  - long
  - float
  - double

# Typy danych – typy obiektowe

- String – do przechowywania tekstów
- BigDecimal – do przechowywania liczb o ścisłe określonej precyzji i zawiera metody do wykonywanie operacji arytmetycznych o ścisłe określonym sposobie zaokrąglania wyniku

## Zadanie 12

- Przypisz wpisy z pierwszego wiersza do zmiennych o odpowiednim typie

# POJO

- Plain Old Java Objects
- Zawiera definicje pól, metody do pobierania i ustawiania wartości

## Zadanie 13

- Stwórz POJO dla wiersza z pliku

# Coding style

- Aby kod był czytelny i zrozumiały należy stosować się do pewnych zasad:
  - odpowiednia kolejność różnych elementów w klasie
  - odpowiednie nazwy
  - podział na klasy pod względem spełnianych funkcji
  - krótkie klasy i metody
  - podział na pakiety

## Zadanie 14

- Stworzenie serwisu do czytania plików
  - podział na odpowiednie klasy
  - podział na pakiety
  - wydzielenie serwisu / komponentu do czytania plików
  - serwis wypisuje na ekran liczbę wierszy i zwraca listę POJO odwzorowujących wczytany plik

# Kolekcje

- Kolekcja to zbiór obiektów zwanych elementami, duża wydajność kolekcji dzięki odpowiednim strukturom danych i algorytmom
- Podstawowe typy kolekcji
  - zbiory (Set) - unikatowość elementów
  - listy (List) - uporządkowanie elementów
  - kolejki (Queue) – kolejkowanie elementów
  - mapy (Map) – tablica klucz-wartość

# Generics

- rozszerzenie języka Java pozwalające na parametryzowanie typów
- tworzenie szablonów kodu, zawierających typy parametryzowane
- typy generyczne sprawdzane są podczas komplikacji,
- informacja o typach parametryzowanych jest usuwana zdefinicji klasy w runtime, typy parametryzowane zamieniane są na typy ogólne

# Sortowanie - TreeSet

- TreeSet to struktura, która automatycznie przechowuje elementy posortowane (sortowanie naturalne lub poprzez komparator)
- Element, który wstawiamy do TreeSet musi implementować interfejs Comparable<TypElementu>

## Zadanie 15

- Serwis utworzony w poprzednim zadaniu powinien zwracać listę posortowaną rosnącą, wg nazwiska

Dziękuję za uwagę