



Wyższa Szkoła Bankowa
Gdańsk Gdynia

Analiza danych w języku Java I

Zajęcia 6 - Zbiór narzędzi

Anna Pakeizer

Konstruktor

- Konstruktor to specjalna metoda wywoływana podczas tworzenia obiektu.
- Nazwa konstruktora jest zawsze taka sama, jak nazwa klasy, w której się znajduje.
- Konstruktor nie zwraca żadnej wartości.

Przykład 1 - Konstruktor

```
class ABC {  
  
    public static void main(String[] args) {  
        // Zima z = new Zima();  
        // Zima z = new Zima("Idzie wiosna!");  
    }  
}  
  
class Zima {  
    Zima() {  
        System.out.println("Witaj Zimo!");  
    }  
    Zima(String komunikat) {  
        System.out.println(komunikat);  
    }  
}
```

this

Co zwraca słowo kluczowe *this*?

Zwraca referencję do obiektu w metodzie niestatycznej, na rzecz którego (ta metoda) została wywołana.

Przykład 2 - *this*

```
class ABC {  
  
    public static void main(String[] args) {  
        Czlowiek cz = new Czlowiek(5);  
  
        System.out.println(cz.wiek);  
    }  
}  
  
class Czlowiek {  
    int wiek;  
    Czlowiek(int wiek) {  
  
        this.wiek = wiek;  
    }  
}
```

```
1  class ABC {
2
3      ▶ Run | 🐛 Debug
4      public static void main(String[] args) {
5          Czlowiek cz = new Czlowiek(5);
6          System.out.println(cz.wiek);
7      }
8  }
9
10
11  class Czlowiek {
12      int wiek;
13      Czlowiek(int wiek) {
14
15          this.wiek = wiek;
16      }
17  }
18
19
```

The image shows an IDE window with two tabs: 'Untitled-1' and 'ABC.java'. The 'ABC.java' tab is active, displaying the code for two classes. The first class, 'ABC', has a 'main' method that creates a 'Czlowiek' object and prints its 'wiek' attribute. The second class, 'Czlowiek', has a constructor that sets the 'wiek' attribute. Red arrows indicate the flow of data: one from the '5' in the constructor call to the 'wiek' parameter, and another from the 'cz.wiek' in the print statement to the 'this.wiek' in the constructor. A green arrow points from the 'Czlowiek' class name to the class definition.

Kompozycja

Kompozycja polega na tworzeniu nowych klas z obiektów już istniejących klas.

Przykład 3 - kompozycja

```
public class ABC {
    public static void main(String[] args) {
        Samochod sam = new Samochod();
        System.out.println(sam.toString());
        sam.uruchom();
        System.out.println(sam.toString());
        sam.zgas();
        System.out.println(sam.toString());
    }
}

class Silnik {
    boolean uruchomiony;
}

class Samochod { // klasa
    private Silnik s = new Silnik(); // obiekt
    public void uruchom() {
        s.uruchomiony = true;
    }
    public void zgas() {
        s.uruchomiony = false;
    }
    public String toString() {
        return String.valueOf(s.uruchomiony);
    }
}
```

Dziedziczenie

Dziedziczenie polega na tworzeniu nowej klasy z istniejącej klasy uzupełnionej o dodatkowy kod bez modyfikacji istniejącego kodu.

Przykład 4 - dziedziczenie

```
public class ABC {  
    public static void main(String[] args) {  
        Kot k = new Kot();  
        k.dajGlos();  
        k.spij();  
    }  
}  
  
class Zwierz {  
    public void dajGlos() {  
    }  
    public void spij() {  
        System.out.println("Spi...");  
    }  
}  
  
class Pies extends Zwierz {  
    public void dajGlos() {  
        System.out.println("Hau hau");  
    }  
}  
  
class Kot extends Zwierz {  
    public void dajGlos() {  
        System.out.println("Miau");  
    }  
}
```

Przykład 5 - interfejsy

```
public class Main {

    public static void main(String args[]) {

        Maluch m = new Maluch();
        m.zwiekszPredkosc(3);
        m.zmniejszPredkosc(1);

    }

}

interface Pojazd {
    public void zwiekszPredkosc(int oIle);
    public void zmniejszPredkosc(int oIle);
}

class Maluch implements Pojazd {

    public void zwiekszPredkosc(int oIle) {
        System.out.println("zwiększyłem o : " + oIle);
    }

    public void zmniejszPredkosc(int oIle) {
        System.out.println("zmniejszyłem o : " + oIle);
    }

}
```

Przykład 6 - interfejsy

Klasy implementujące mogą dodawać własne metody

```
class Maluch implements Pojazd {  
  
    public void zwiekszPredkosc(int oIle) {  
        System.out.println("zwiększyłem o : " + oIle);  
    }  
  
    public void zmniejszPredkosc(int oIle) {  
        System.out.println("zmniejszyłem o : " + oIle);  
    }  
  
    public void uzyjKlaksону() {  
        System.out.println("honk honk");  
    }  
  
}
```

Przykład 7 – obsługa błędów

```
public static void main(String args[]) {  
    System.out.println(2 / 2); // 1  
}
```

```
public static void main(String args[]) {  
    System.out.println(2 / 0);  
}
```

Exception in thread "main" java.lang.ArithmeticException: / by zero
at test01.Main.main(Main.java:6)

Obsługa błędów c.d.

```
public static void main(String args[]) {  
    try {  
        System.out.println(2 / 0);  
    } catch (Exception e) {  
        System.out.println(e.getMessage());  
    }  
}
```

/ by zero

Obsługa błędów c.d.

```
public static void main(String args[]) {  
    try {  
        System.out.println(2 / 0);  
    } catch (ArithmeticException ae) {  
        System.out.println("ArithmeticException: " + ae.getMessage());  
    } catch (Exception e) {  
        System.out.println("Exception: " + e.getMessage());  
    }  
}
```

ArithmeticException: / by zero

Obsługa błędów c.d.

```
public static void main(String args[]) {  
    try {  
        System.out.println(2 / 1);  
    } catch (Exception e) {  
        System.out.println("Wykonam sie tylko jesli wystapi blad");  
        System.out.println("Exception: " + e.getMessage());  
    } finally {  
        System.out.println("Czy jest blad czy nie i tak sie wykonam");  
    }  
}
```

Przykład 8 - specyfikacja wyjątków

Jakie wyjątki mogą wystąpić w metodzie?

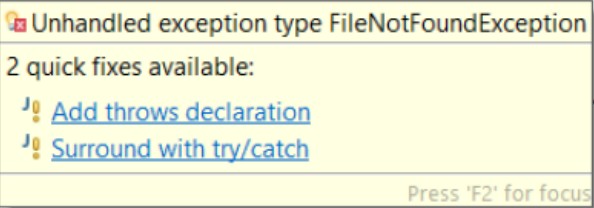
```
public static void main(String args[]) {  
    wydrukujZawartoscPliku("C:\\Users\\Wlodek\\Desktop\\abc.txt");  
}  
  
public static void wydrukujZawartoscPliku(String sciezka) {  
  
}
```

Specyfikacja wyjątków c.d.

```
public static void main(String args[]) {  
    wydrukujZawartoscPliku("C:\\Users\\Wlodek\\Desktop\\abc.txt");  
}
```

```
public static void wydrukujZawartoscPliku(String sciezka) throws  
FileNotFoundException{  
  
}
```

```
public static void main(String args[]) {  
    wydrukujZawartoscPliku("C:\\Users\\Wlodek\\Desktop\\abc.txt");  
}
```



```
publ    rtoscPliku(String sciezka) throws FileNotFoundException{  
  
}
```