Dziedziczenie klas

Karol Kulinowski, Agnieszka Kazimierska

9 maja 2023

Dziedziczenie klas

- **Dziedziczenie** - przekazanie pewnych cech (pól czy metod) innym klasom. Pozwala to na tworzenie hierarchii klas, które posiadają wspólne cechy.
- Jak w genetyce dzieci dziedziczą pewne cechy po swoich rodzicach, tak samo klasy mogą „dziedziczyć" pola oraz metody.
- Dziedziczenie tworzy relację typ „to jest" albo „należy do" między klasami np.:
  - `Pies` *należy do* `Ssak`, `FiatPanda` *to jest* `Auto`,
  - Za to nie działa już:
    `FiatPanda` *to jest* `Zwierze`, bo Fiat Panda nie jest zwierzęciem,
    `Książka` *należy do* `WładcaPierścieni`, Władca Pierścieni jest książką ale nie każda książka to Władca Pierścieni.
  - Klasa podrzędna **rozszerza** klasę nadrzędną.
- Jak klasy dziedziczą? Poprzez użycie słowa kluczowego `extends`.

```
1 // Klasa nadrzędna
2 public class Shape {
3     //definicje pól i metod w klasie Shape
4 }
5 // Klasa Square dziedzicząca klasie Shape
6 public class Square extends Shape {
7     //metody i pola w klasie Square, które dziedziczą od klasy Shape
8 }
9
```

- Klasa nadrzędna może mieć wiele klas podrzędnych, ale każda klasa może dziedziczyć po tylko **jednej** klasie.
- Modyfikator `final` oznacza klasę, po której nie można dziedziczyć, w przypadku zmiennych oznacza on, że dana zmienna jest stała.
- Słowo kluczowe `this` oznacza dany obiekt, w ogólności można je pominąć. Pomaga on odróżnić zmienne lokalne od pól klasy.
- Instrukcja `super` pozwala wywołać konstruktor klasy nadrzędnej i powinna się znajdować na początku konstruktora. W przypadku konstruktorów domyślnych nie jest ona konieczna.

- **Polimorfizm** oznacza przyjmowanie przez jeden element wielu form
  - **przeciążanie** (*overloading*) - metody mają tę samą nazwę, ale przyjmują różne parametry (różne typy albo różną liczbę parametrów), tzn. ich sygnatury się różnią; nie jest bezpośrednio związane z dziedziczeniem,
  - **przesłanianie** (*overriding*) - klasa podrzędna i nadrzędna mają metody o tej samej sygnaturze, metoda z klasy podrzędnej "przesłania", zastępuje metodę z klasy bazowej; metody przesłaniane są często podawane z adnotacją `@Override` (ale nie jest to obowiązkowe).

Przykład dziedziczenia klas: Zwierz → Ssak → Pies, Kot

Nazwa pliku: Zwierz.java

```
1  package zwierzeta;
2
3  /**
4   * Klasa nadrzędna Zwierz
5   */
6  public class Zwierz {
7      protected int wiek;
8      protected String gatunek;
9      /**
10      * Konstruktor
11      */
12     public Zwierz(int wiek, String gatunek) {
13         this.wiek = wiek;
14         this.gatunek = gatunek;
15     }
16     /**
17      * Ustal zmienną gatunek
18      */
19     public void set_gatunek(String gatunek) {
20         this.gatunek = gatunek;
21     }
22     /**
23      * Pobierz zmienną gatunek
24      */
25     public String get_gatunek() {
26         return this.gatunek;
27     }
28     /**
29      * Metoda wypisująca informacje
30      */
31     public void getInfo() {
32         System.out.println("Cześć, należę do gatunku " + this.gatunek+ "
       i mam " + this.wiek + " lat");
33     }
34 }
```

Nazwa pliku: Ssak.java

```java
package zwierzeta;

/**
 * Klasa Ssak dziedzicząca po klasie Zwierz (Ssak to jest Zwierz)
 */
public class Ssak extends Zwierz {
    protected boolean lata;
    protected boolean plywa;
    /**
     * Konstruktor klasy Ssak
     */
    public Ssak() {
        super(10, "Homo Sapiens");
        this.lata = false;
        this.plywa = false;
    }
}
```

Nazwa pliku: Pies.java

```java
package zwierzeta;

/**
 * Klasa Pies dziedzicząca po klasie Ssak (Pies należy do Ssak)
 */
public final class Pies extends Ssak {
    private String rasa;
    private String imie;  // Zwierzęta nieudomowione nie mają imion
    /**
     * Konstruktor klasy Pies pobierający trzy parametry
     */
    public Pies (int wiek, String rasa, String imie) {
        // Nie ma potrzeby zawierania instrukcji super(), ponieważ klasa nadrzędna Ssak posiada konstruktor domyślny
        super();
        this.wiek = wiek;
        this.rasa = rasa;
        this.imie = imie;
    }
    /**
     * Metoda wypisująca informacje o obiekcie
     */
    @Override
    public void getInfo() {
        System.out.println("Hau hau, mam na imię " + this.imie + " jestem psem rasy " + this.rasa + " i mam " + this.wiek + " lat, szczek szczek");
    }
}
```

Nazwa pliku: Kot.java

```
1  package zwierzeta;
2
3  /**
4   * Klasa Kot dziedzicząca po klasie Ssak (Pies należy do Ssak)
5   */
6  public final class Kot extends Ssak {
7      private String kolor;
8      private String imie;  // Zwierzęta nieudomowione nie mają imion
9      /**
10      * Konstruktor klasy Kot pobierający trzy parametry
11      */
12     public Kot (int init_wiek, String init_kolor, String init_imie) {
13         // Nie ma potrzeby zawierania instrukcji super(), ponieważ klasa
       nadrzędna Ssak posiada konstruktor domyślny
14         this.wiek = wiek;
15         this.kolor = init_kolor;
16         this.imie = init_imie;
17     }
18     /**
19      * Metoda wypisująca informacje o obiekcie
20      */
21     @Override
22     public void getInfo() {
23         System.out.println("Miau miau, mam na imię " + this.imie + "
       jestem kotem koloru " + this.kolor + " i mam " + this.wiek
24         + " lat, idę spać");
25     }
26 }
```

Nazwa pliku: Main.java

```
1  import zwierzeta.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          Zwierz jakis_zwierz_1 = new Zwierz(10, "Homo Sapiens");
6          Pies pies_1 = new Pies(5, "Welsh corgi", "Kokos");
7          Kot kot_1 = new Kot(6, "Czarny", "Salem");
8          //
9          System.out.print("Gatunek psa przed wywołaniem metody set_gatunek
       (): " + pies_1.get_gatunek());
10         pies_1.set_gatunek("Canis familiaris");
11         System.out.println(" oraz po: " + pies_1.get_gatunek());
12         //
13         System.out.print("Gatunek kota przed wywołaniem metody
       set_gatunek(): " + kot_1.get_gatunek());
14         kot_1.set_gatunek("Felis catus");
15         System.out.println(" oraz po: " + kot_1.get_gatunek());
16         //
17         jakis_zwierz_1.getInfo();
18         pies_1.getInfo();
19         kot_1.getInfo();
20     }
21 }
```