Zaawansowane programowanie obiektowe - wykład 2

dr Piotr Jastrzębski

Dziedziczenie

Dziedziczenie

- „Mechanizm”, dzięki któremu jedna z klas może posiąść cechy innej klasy.
- „cechy” – modyfikator dostępu
- Klasa bazowa – klasa, z której jest dziedziczone
- Klasa potomna/pochodna – klasa, która dziedziczy

Rozważmy przykład:

- Rozważmy mamy trzy klasy `Pojazd`, `Rower`, `Samochod`.
- Samochód jest Pojazdem.
- Rower jest Pojazdem.

Po co jest dziedziczenie?

- Klasy potomne mogą współdzielić zachowania klas potomnych.
- Możemy rozszerzyć klasy bez powielania kodu.
- Uwypukla wspólne cechy (wspiera abstrakcję).

```
class Bazowa
{
    public int pole;
    public void Metoda1(){ }
}

class Pochodna extends Bazowa
{
    public int Metoda2()
    {
        return pole * 2;
    }
}
```

Wielodziedziczenie klas?

Wielodziedziczenie klas – dziedziczenie z kilku klas bazowych jednocześnie. W języku Java jest to nie możliwe.

Inne przykładowe języki programowanie umożliwiające wielodziedziczenie: C++, Perl, Python.

```
class Pojazd
{
    // elementy klasy
}
class Samochod extends Pojazd
{
    // elementy klasy
}
class Tir extends Samochod, Pojazd
{
    // elementy klasy
}
```

Klasy zaplombowane, finalne

- Klasy z modyfikatorem dostępu final są traktowane jako „zaplombowane”.
- Nie mogą być klasami bazowymi dla innych - nie można z nich dziedziczyć.
- do Java 14 `final`, obecnie `sealed`

https://javastart.pl/baza-wiedzy/slownik/sealed-classes

```
final class KlasaBazowa
{
    // elementy klasy
}

class KlasaDruga extends KlasaBazowa
{
    // elementy klasy
}
```

- Każda klasa w Java dziedziczy niejawnie z klasy `Object`.

Konstruktory a dziedziczenie

- Konstruktory nie podlegają dziedziczeniu.
- W każdej klasie konstruktor należy napisać na nowo.
- Za pomocą inicjatora base możemy wywołać konstruktor klasy bazowej.

```
class Pojazd
{
    protected String marka;
    public Pojazd(String marka)
    {
        this.marka = marka;
    }
}
class Samochod extends Pojazd
{
    int iloscKol;
    public Samochod(String marka, int iloscKol)
    {
        super(marka);
        this.iloscKol = iloscKol;
    }
}
```

Rzutowanie a dziedziczenie

- Rzutowanie w górę (zawsze możliwe i skuteczne).
  - Samochód możemy traktować jak Pojazd.
- Rzutowania w dół (możliwe gdy rzeczywiście obiekt jest typu pochodnego).
  - Nie każdy Pojazd jest Samochodem.
  - Nie każdy prostokąt jest kwadratem.

Rzutowanie w górę

```
//obiekt klasy potomnej
Samochod a1 = new Samochod();
//rzutowanie w górę
Pojazd a2 = a1;
```

Rzutowanie w dół

```
Samochod a1 = new Samochod();
Pojazd a2 = a1;
Samochod a3 = a2;
```

Jak to naprawić? I sposób - jawny rzut

```
Samochod a1 = new Samochod();
Pojazd a2 = a1;
Samochod a3 = (Samochod)a2;
```

Ale mamy ryzyko błędu:

```
Pojazd p1 = new Pojazd();
Samochod p2 = (Samochod)p1;
```

II sposób - bezpieczne rzutowanie - `is` - zwraca `true` jeśli lewa strona obiektu może zostać rzutowana na typ określony po prawej stronie.

```
Pojazd p1 = new Samochod();
if (p1 instanceof Samochod)
{
    Samochod p2 = (Samochod)p1;
}
```

III sposób: pattern matching (preview)

```
Pojazd p1 = new Samochod();
if (p1 instanceof Samochod p2)
{
    System.out.println(p1.equals(p2));
}
```

Modyfikator dostępu w dziedziczeniu

- `protected` - elementy dostępne dla klas pochodnych

Interfejsy

Definicja interfejsu

interfejs to kontrakt, który mówi co klasa implementująca może robić, ale nie jak ma to robić.

https://www.samouczekprogramisty.pl/interfejsy-w-jezyku-java/

Przykłady implementacji systemowych interfejsów

Sortowanie https://www.geeksforgeeks.org/comparable-vs-comparator-in-java/

Polimofizm

Definicja polimorfizmu wg wikipedii

Polimorfizm (z gr. wielopostaciowość) – mechanizmy pozwalające programiście używać wartości, zmiennych i podprogramów na kilka różnych sposobów. Inaczej mówiąc jest to możliwość wyabstrahowania wyrażeń od konkretnych typów.

Polimorfizm w Java

- statyczny
  - przeciążenie funkcji
  - przeciążenie operatorów
- dynamiczny
  - metody wirtualne?
  - metody abstrakcyjne

Metody abstrakcyjne

- poprzedzone słowem kluczowym `abstract`
- zdefiniowane w klasie bazowej
- nie zawierają ciała metody (podobnie jak przy interfejsach)
- mogą być zadeklarowane tylko w klasie abstrakcyjnej (poprzedzonej słowem `abstract`)
- nie możemy stworzyć egzemplarza (obiektu) klasy abstrakcyjnej (podobnie jak przy interfejsach), ale możemy dziedziczyć po klasie abstrakcyjnej
- klasa abstrakcyjna może posiadać zwykłe metody (z implementacją)
- klasa pochodna do klasy abstrakcyjnej musi przesłonić wszystkie metody abstrakcyjne

https://www.w3schools.com/java/java_abstract.asp

Metody wirtualne w Java?

https://stackoverflow.com/questions/4547453/can-you-write-virtual-functions-methods-in-java