

Klasių hierarchija

Mindaugas Karpinskas
2018



Klasių hierarchija

- Visos klasės paveldi klasės **java.lang.Object**
- Jei nėra nurodyta tėvinė klasė, tai pagal nutylėjimą paveldės Object savybes
- Jei kokia nors tėvinė klasė nurodyta, tada paveldės Object savybes per tėvinę klasę

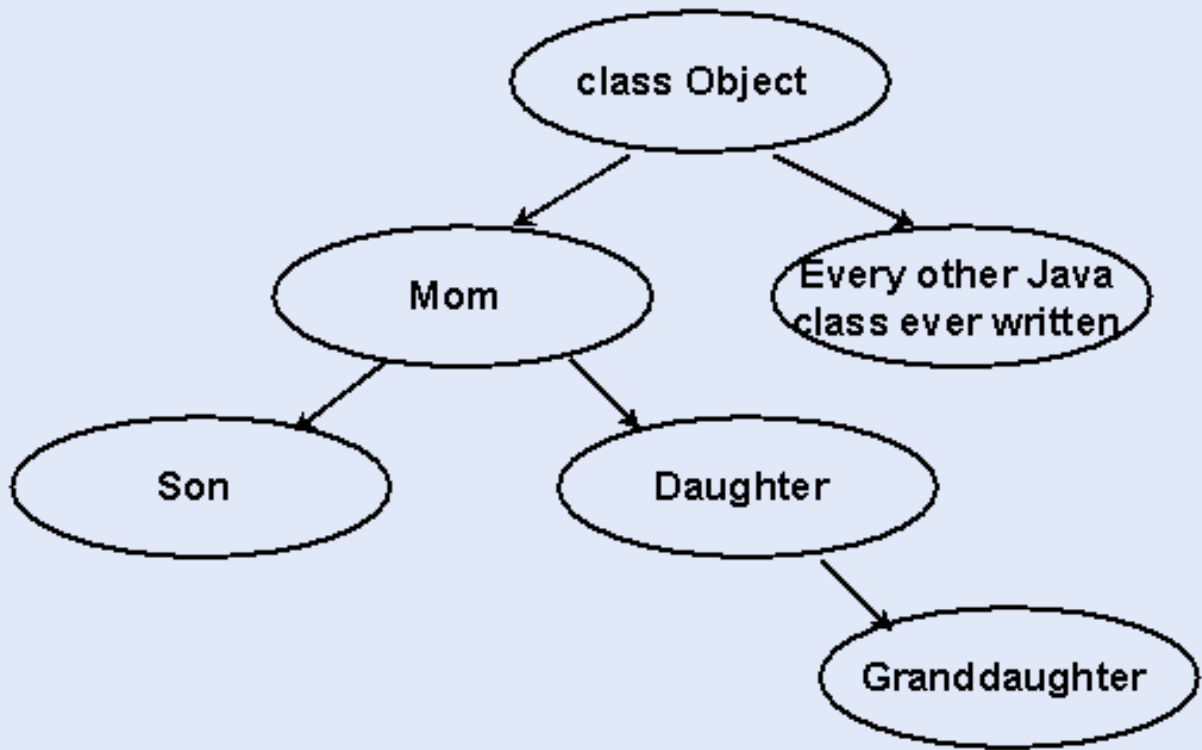
Pvz.

```
class Mom {  
    // declarations, definitions  
}  
class Son extends Mom {  
    // declarations, definitions  
}  
class Daughter extends Mom {  
    // declarations, definitions  
}
```

- Son ir Daughter klasės paveldės Mom klasės kintamuosius ir metodus
- Mom yra pagrindinė klasė, į kurią remiasi kitos dvi klasės.
- Son ir Daughter yra Mom klasės poklasės (subclasses) ir Mom yra Son ir Daughter superklasė (superclass)
- Java kalboje kiekviena klasė turi tik **vieną** tiesioginę superklasę

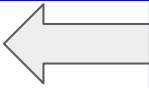
Mom klasės superklasė?

Paveikslėlis iliustruoja šio klausimo atsakymą. Trumpai tariant visų klasių hierarchijos viršūnėje yra Object klasė ir, jei nenurodytas paveldimumas, tai nutylimasis paveldimumas yra **extends** Object



Taigi mūsų hierarchijos pavyzdys ekvivalentus tokiam kodui:

```
class Mom extends Object {  
    // declarations, definitions  
}  
class Son extends Mom {  
    // declarations, definitions  
}  
class Daughter extends Mom {  
    // declarations, definitions  
}
```



- Kodėl tokia globali hierarchija naudinga?
 - kadangi žinome, kad visos klasės turi bendrą superklasę Object, jos metodus ir kintamuosius gali naudoti bet kuri kita klasė.
- Object klasė apibrėžia **equality** metodą skirtą patikrinti, ar dviejų klasių turinys vienodas.
- Taip pat Object klasėje yra realizuotas daugiagijškumo (multithreading) savybės.
- Taip pat atpuola daugelio hierarchijų tarpusavio sąsajų problema, nes visos jos yra vienos globalios hierarchijos dalys.
- Naudinga ir tai, kad esame garantuoti, kad kiekviena klasė turi savo superklasę.

Keletas klasės **Object** metodų

- equals – lygina objektų turinį
- clone – sukuria naują objektą - to paties objekto kopiją
- toString – gražina tekstinę objekto informaciją

Metodo equals pavyzdys

```
class A {  
}
```

instanceof - patikrina kokios klasės objektas

B b = (B) obj;

(B) - **cast**, traktuojama, kad objektas gali būti priskirtas kitos klasės kintamajam

```
class B {  
    int i;  
    A a = new A();  
    @Override  
    public boolean equals(Object obj) {  
        if (obj instanceof B) {  
            B b = (B) obj;  
            if ((i == b.i) &&  
                a.equals(b.a))  
                return true;  
            else  
                return false;  
        }  
        return false;  
    }  
}
```


Užduotis 1

Sukurti klasę Asmuo, su dviem laukais: vardas, pavardė. Klasėje perrašyti equals metodą.

Main metode sukurti keletą egzempliorių / objektų ir palyginti juos.

Metodo toString pavyzdys

```
class A {  
    // toString....  
}
```

```
class B {  
    int i;  
    A a = new A();  
  
    public String toString() {  
        return "i=" + i + " a=" + a.toString();  
    }  
}
```

Užduotis2

Sukurti klasę `Automobilis`, su dviem laukais: `valstNr`, `marke`. Klasėje perrašyti `toString` metodą.

Sukurti klasę `AutoParkas` kuri turi deklaruotą kintamąjį/masyvą `Automobilis[] autos`; `AutoParkas` taip pat perrašo metoda `toString` ir gražina visą info apie automobilius, kiek automobilių, koks pirmas, koks paskutinis, visų automobilių info...

Main metode sukurti keletą egzempliorių / objektų ir išspausdinti `toString` rezultatą.