

---

# Mantošana un Superklase OOP

*Ainārs Skrubis*

---

---

# Kas ir OOP?

**OOP** jeb **Objektorientēta programmēšana** ir datorprogrammēšanas paradigma, kas programmatūras dizainu organizē pamatojoties uz objektiem ar, kuriem izstrādātājs vēlas manipulēt, nevis uz funkcijām vai loģiku, kas nepieciešama, lai veiktu manipulācijas. OOP būtība ir censties programmēšanā ieviest tādas reālās dzīves entītijas kā mantošanu (*inheritance*), slēpšanu (*hiding*), plimorfismu (*polymorphism*), objektus, u.c

# Kas ir OOP?

Šī paradigma ir labi piemērota lielām, **sarežģītām programmām**, kuras aktīvi tiek papildinātas vai uzturētas. Piemēram priekš programmām, kas tiek radītas, lai nodrošinātu ražotnes sistēmas simulēšanu, arhitektūras/dizaina rīkiem, mobilajām lietotnēm.

OOP ir četri stūrakmeņi

- Abstrakcija
- Polimorfisms
- **Mantošana**
- Iekapsulēšana



---

# Kas ir mantošana?

**Mantošana** (Inheritance) ir fundamentāls OOP jēdziens. Java valodā tas ir mehānisms, ar kura palīdzību viena klase var mantot citas klases īpašības (**atribūtus un metodes**). Java valodā mantošana nozīmē jaunu klašu veidošanu, balstoties uz esošajām. Klase, kas manto no citas klases, var atkārtoti izmantot šīs klases metodes un atribūtus. Papildus tam, esošajai klasei var pievienot jaunus atribūtus un metodes.

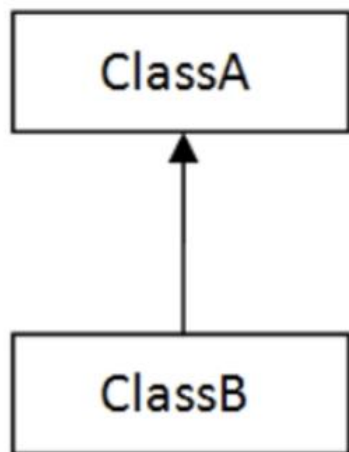
---

# Kas ir superklase un subklase

- **Superklase/Vecāku klase:** Klase, kuras īpašības tiek mantotas, tiek saukta par superklasi (vai bāzes klasi, vai vecāku klasi).
- **Subklase/Bērnu klase:** Klase, kas manto no citas klases, tiek saukta par subklasi (vai atvasinātu klasi, paplašinātu klasi, vai bērnu klasi). Subklase var pievienot savus atribūtus un metodes papildus tiem, ko tā manto no superklases.

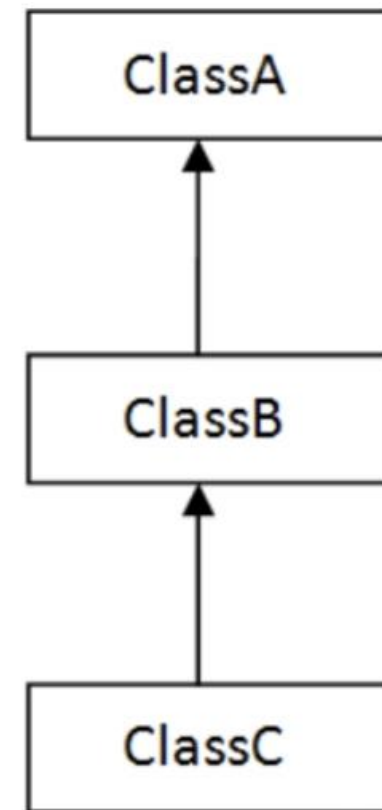
# Mantošanas veidi

**Viena līmeņa** - viena klase manto otras īpašības



1) Single

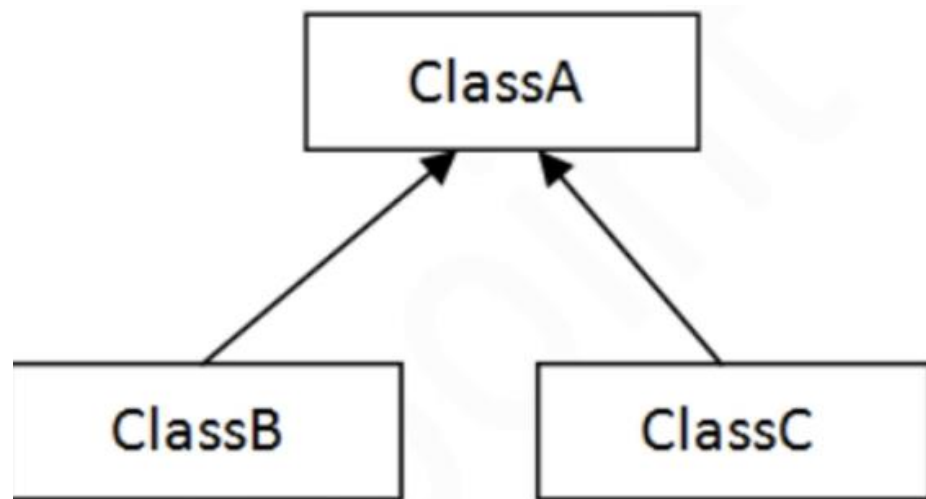
**Vairāku līmeņu** - veidojas mantošanas ķēde starp klasēm, katra nākošā klase manto iepriekšējās klases īpašības



2) Multilevel

# Mantošanas veidi

**Hierarhiska** - situācija kad divas vai vairākas klases manto īpašības no vienas klases.



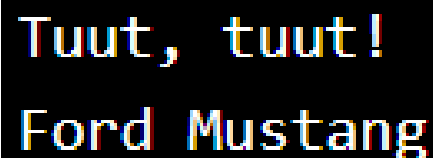
3) Hierarchical

```
class Animal{  
    void eat(){System.out.println("eating...");}  
}  
  
class Dog extends Animal{  
    void bark(){System.out.println("barking...");}  
}  
  
class Cat extends Animal{  
    void meow(){System.out.println("meowing...")  
}  
  
class TestInheritance3{  
    public static void main(String args[]){  
        Cat c=new Cat();  
        c.meow();  
        c.eat();  
    }  
}
```

# Kā pielieto mantošanu

```
class Vehicle {  
    protected String brand = "Ford";  
    public void honk() {  
        System.out.println("Tuut, tuut!");  
    }  
}
```

```
class Car extends Vehicle {  
    private String modelName = "Mustang";  
    public static void main(String[] args) {  
        Car myFastCar = new Car();  
        myFastCar.honk();  
        System.out.println(myFastCar.brand + " " + myFastCar.modelName);  
    }  
}
```



Tuut, tuut!  
Ford Mustang



# Kā un kad izmanto **super**

Atslēgvārds “**super**” attiecas uz superklases (vecāku) objektiem.

To izmanto, lai izsauktu superklases metodes un piekļūtu superklases konstruktoram.

Visbiežāk atslēgvārdu “**super**” izmanto, lai novērstu neskaidrības starp superklasēm un apakšklasēm, kurām ir metodes ar vienādu nosaukumu.

```
class Dzīvnieks {
    Dzīvnieks() {
        System.out.println("Dzīvnieks konstruktors");
    }

    void skaņa() {
        System.out.println("Dzīvnieks skaņa");
    }
}

class Kaķis extends Dzīvnieks {
    Kaķis() {
        super(); // Izsauc Dzīvnieks konstruktora
        System.out.println("Kaķis konstruktors");
    }

    void skaņa() {
        super.skaņa(); // Izsauc Dzīvnieks skaņa
        System.out.println("Kaķis ņaud");
    }
}
```

# Kā pielieto super

```
class Animal { // Superklase
    public void animalSound() {
        System.out.println("The animal makes a sound");
    }
}
```

```
class Dog extends Animal { // Subklase
    public void animalSound() {
        super.animalSound(); // Izsauc super
        System.out.println("The dog says: bow wow");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        myDog.animalSound();
    }
}
```

The animal makes a sound  
The dog says: bow wow

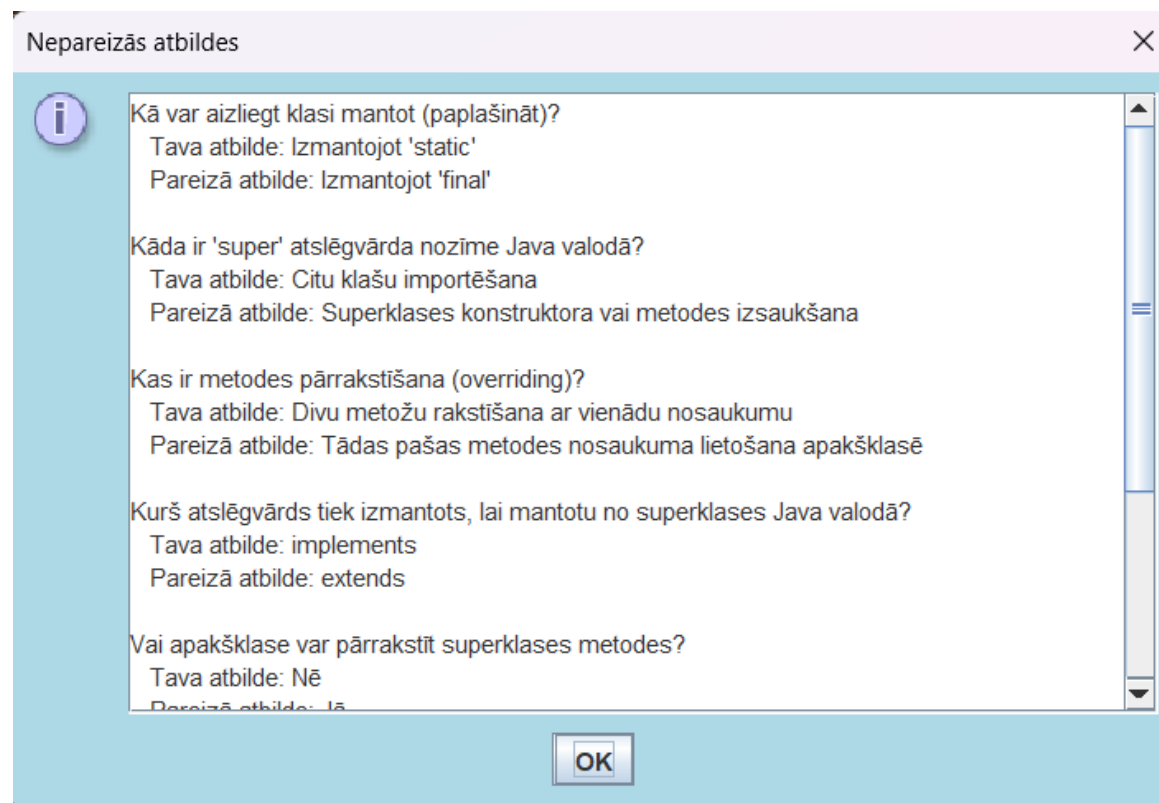
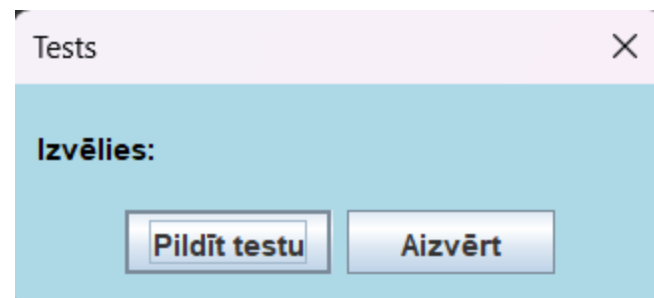
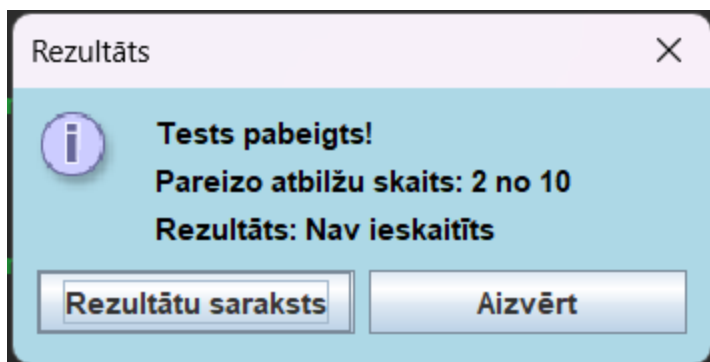
---

# Kapēc lietot mantošanu

- **Koda atkārtota izmantošana:** Superklasē rakstītais kods ir kopīgs visām apakšklasēm. Bērnu klases var tieši izmantot vecāku klases kodu.
- **Metodes pārrakstīšanai (Overriding):** Metodes pārrakstīšana ir iespējama tikai ar mantošanas palīdzību. Tas ir viens no veidiem, kā Java panāk izpildes laika polimorfismu.
- **Abstrakcija:** Abstrakcijas jēdziens, kurā mums nav jāsniedz visa informācija, tiek panākts ar mantošanas palīdzību. Abstrakcija lietotājam parāda tikai funkcionalitāti.
- Tas organizē klases strukturētā veidā, uzlabojot lasāmību un uzturēšanas iespējas.

# Par projektu

- Tika programmēts "Eclipse"
- Patērētais laiks: ~ 2 dienas
- Koda garums: 172 rindas
- Iztrādes līdzekļi: Lists, Mantošana, JOptionPane



---

# Izmantotie avoti

- [GeeksForGeeks](#)
- [W3School](#)
- [ChatGPT](#)
- [Skolo.lv](#) (skolotāj Rāvalda java programmēšanas kursi)

---

**Paldies par uzmanību**