



VILNIAUS TECHNOLOGIJŲ IR VERSLO
PROFESINIO MOKYMO CENTRAS

9 - OBJEKTINIS PROGRAMAVIMAS 2

Jaroslav Grablevski / Justina Balsė

Paveldėjimas (inheritance)

- Tai klasės gebėjimas paveldėti protėvių klasės duomenis (kintamuosius, laukus) ir metodus.
- Klasė, kuri yra išvesta iš kitos klasės vadinama poklase arba **vaikine** (angl. *subclass*, *derived class*, *extended class* arba *child class*).
- Klasė, iš kurios poklasė yra išvesta, vadinama superklase arba **tėvine** (angl. *superclass*, *base class*, *parent class*) .

Paveldėjimas

- Tai klasės gebėjimas paveldėti protėvių klasės duomenis (kintamuosius, laukus) ir metodus.
- Tėvinė klasė apibrėžia tam tikrus kintamuosius bei metodus, kuriuos vaikinė klasė paveldi.
- Vaikinė klasė gali pasipildyti savais kintamaisiais ir metodais
- Vaikinė klasė turi galimybę pakeisti tėvinių metodų veikimą juos perrašant (angl. *override*)

Paveldėjimas

- Java kalboje paveldėjimo ryšys deklaruojamas naudojant žodelį **extends** po kurio nurodoma klasė iš kurios yra paveldima

```
public class Car extends Machine {  
    // ...  
}
```

Paveldējimas



Programmer

name
address
phoneNumber
experience
programmingLanguages
writeCode()

```
class Programmer {  
    String name;  
    String address;  
    String phoneNumber;  
    float experience;  
    String[] programmingLanguages;  
    void writeCode() {}  
}
```

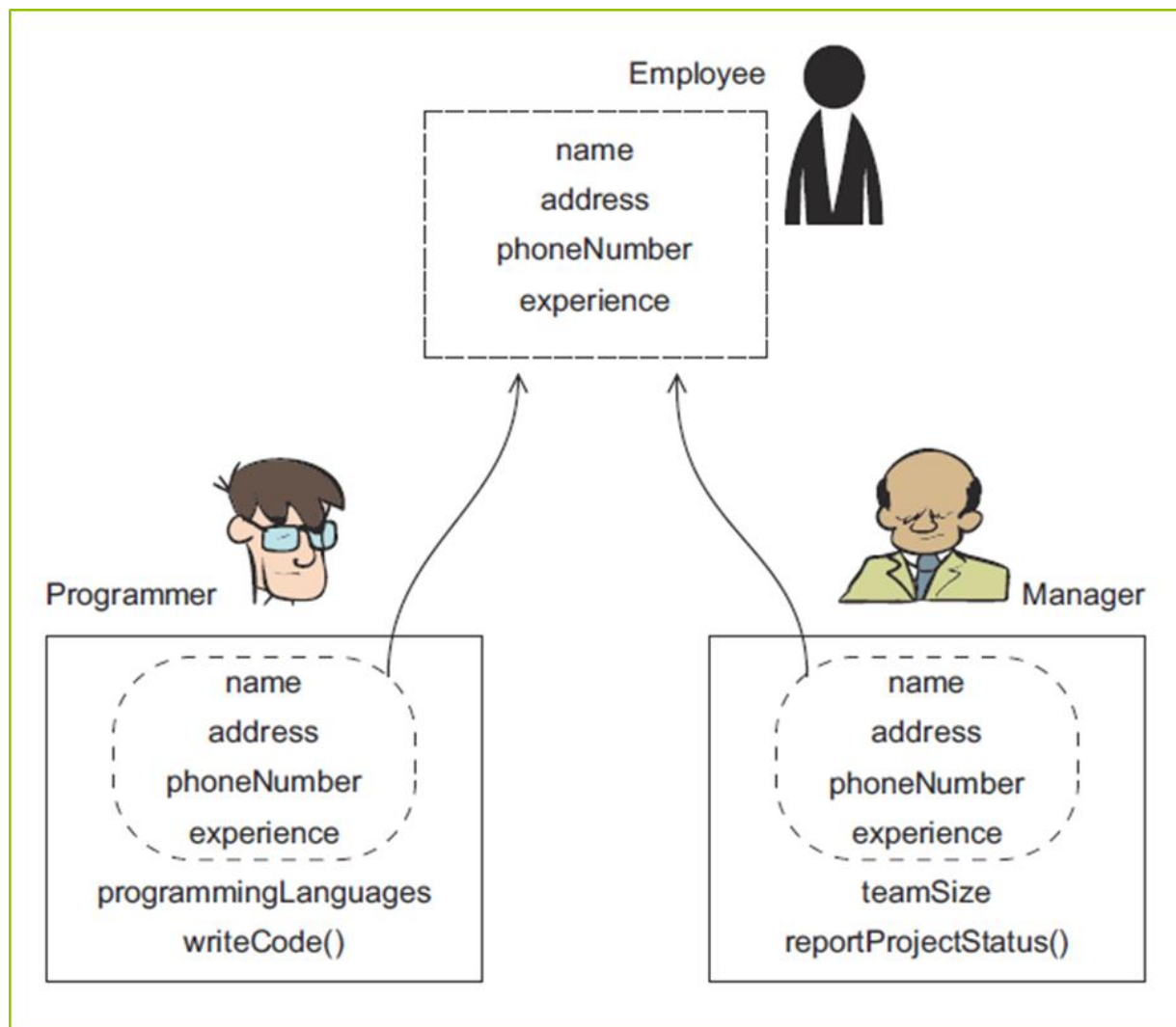


Manager

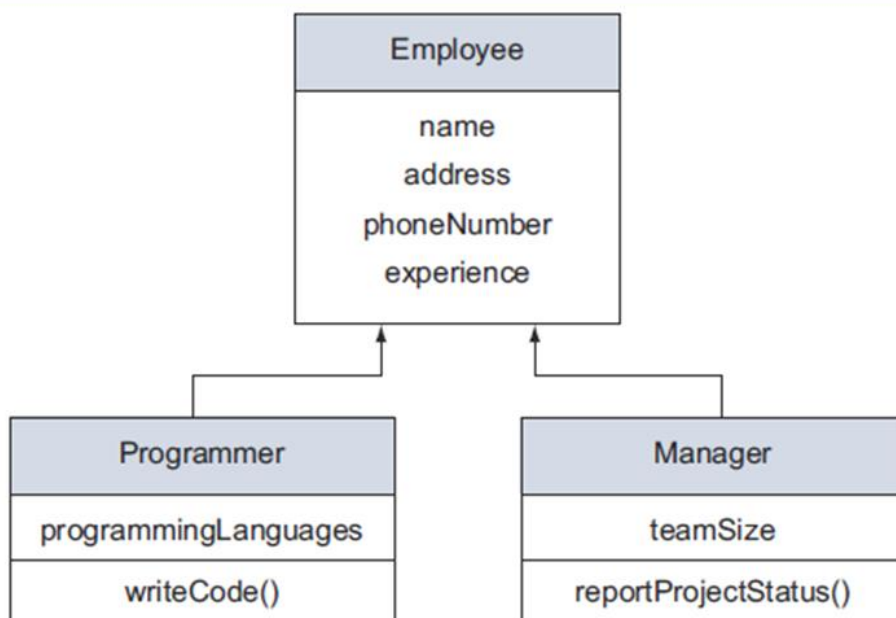
name
address
phoneNumber
experience
teamSize
reportProjectStatus()

```
class Manager {  
    String name;  
    String address;  
    String phoneNumber;  
    float experience;  
    int teamSize;  
    void reportProjectStatus() {}  
}
```

Paveldējimas

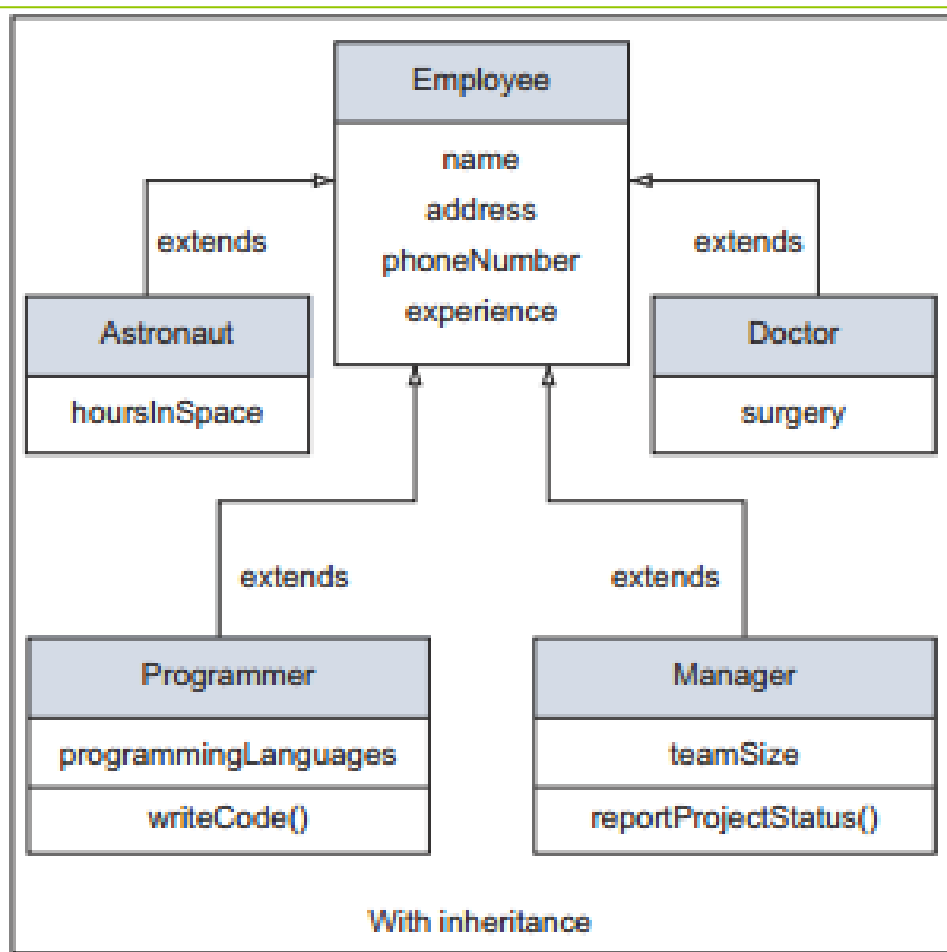
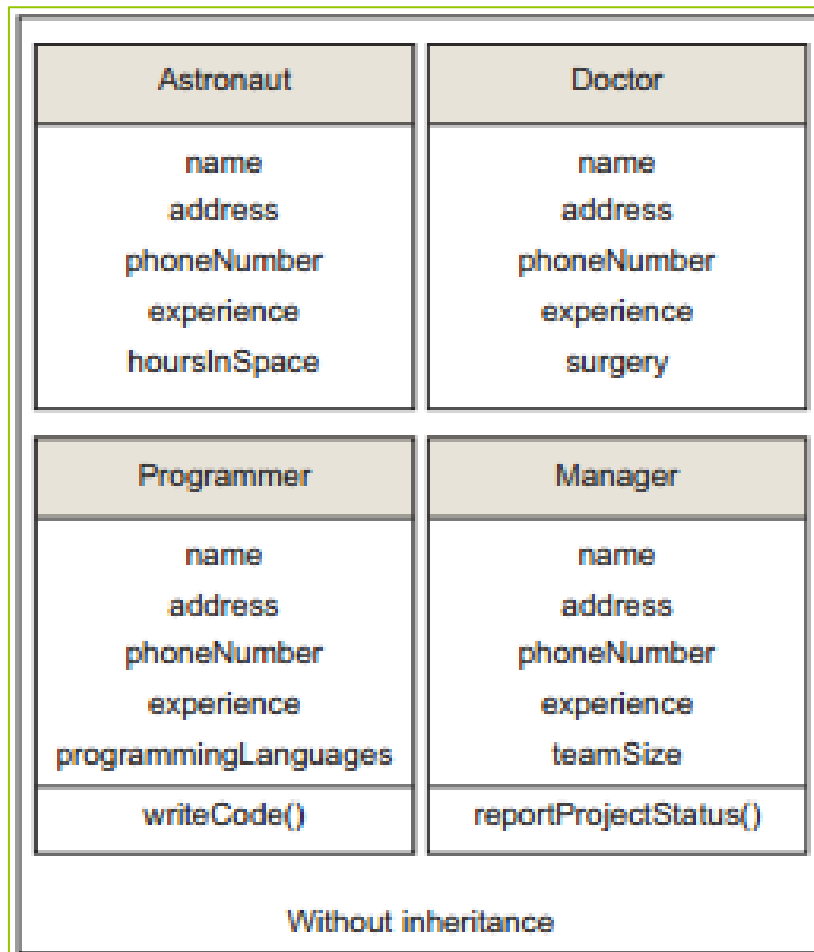


Paveldējimas

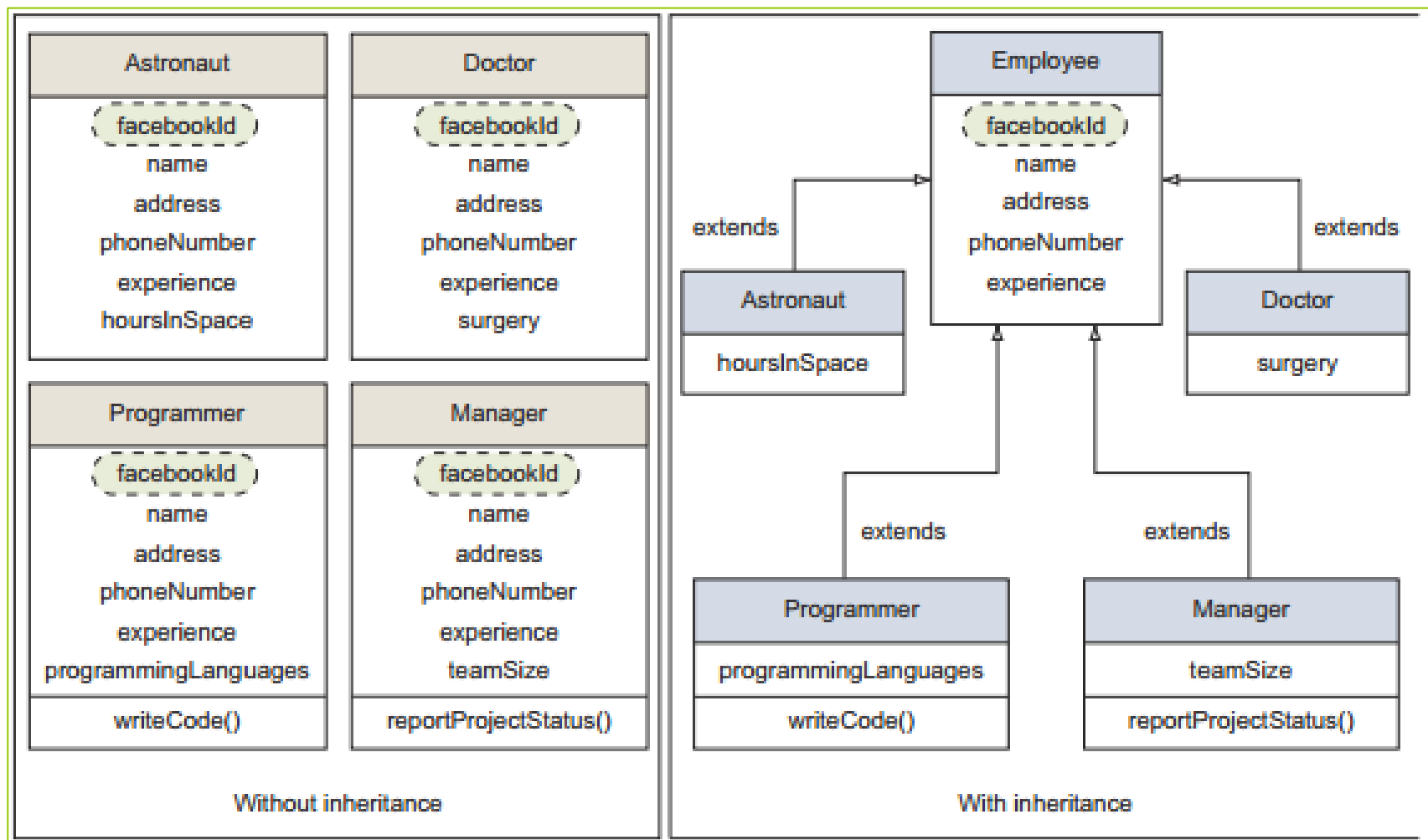


```
class Employee {
    String name;
    String address;
    String phoneNumber;
    float experience;
}
class Programmer extends Employee {
    String[] programmingLanguages;
    void writeCode() {}
}
class Manager extends Employee {
    int teamSize;
    void reportProjectStatus() {}
}
```

Paveldējimas

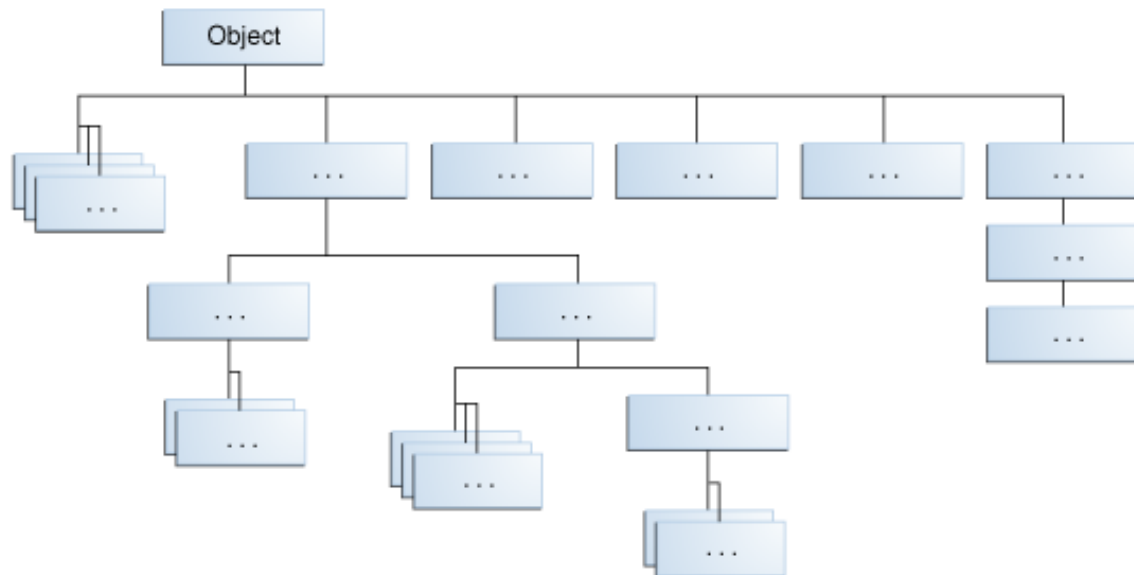


Paveldējimas



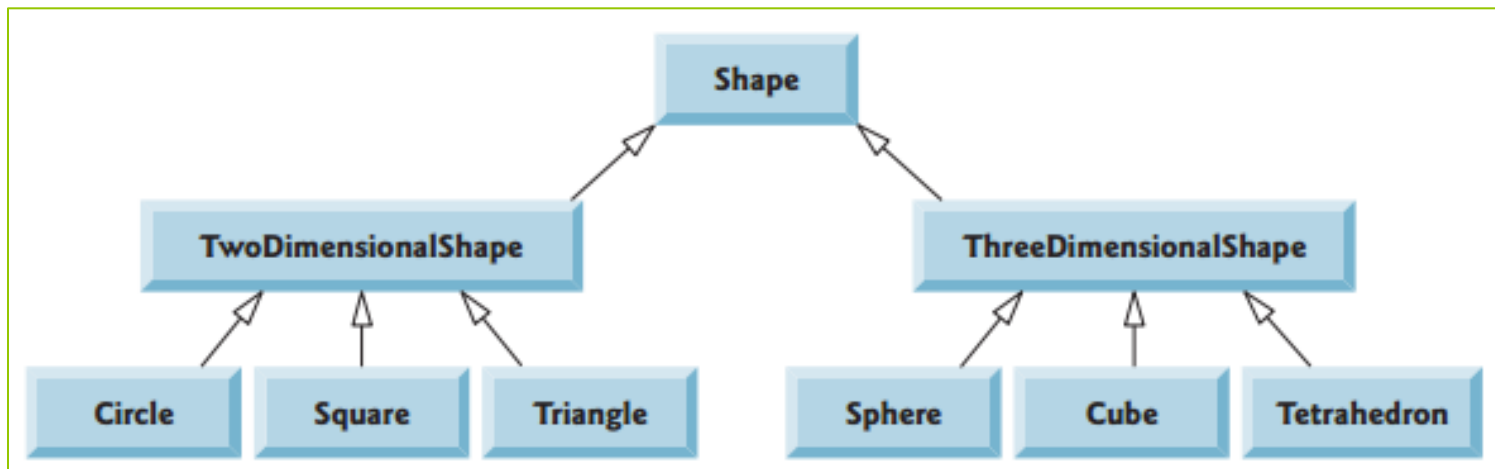
Paveldėjimas

- Klasė gali turėti tik vieną tiesioginę superklasę
- Visos Javos klasės yra kilusios iš *java.lang.Object* klasės ir automatiškai paveldi visus jos metodus



is-a relationship

Superclass	Subclasses
Student	GraduateStudent, UndergraduateStudent
Shape	Circle, Triangle, Rectangle, Sphere, Cube
Loan	CarLoan, HomeImprovementLoan, MortgageLoan
Employee	Faculty, Staff
BankAccount	CheckingAccount, SavingsAccount



Machine ir Car klasės | Pavyzdys (1)

```
public class Machine {  
  
    public void start() {  
        System.out.println("Machine started.");  
    }  
  
    public void stop() {  
        System.out.println("Machine stopped.");  
    }  
}
```

Machine.java

```
public class Car extends Machine {  
  
}
```

Car.java

Main klasė | Pavyzdys (1.2)

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Machine m1 = new Machine();  
  
        System.out.println("Machine class: ");  
        m1.start();  
        m1.stop();  
  
        Car m2 = new Car();  
        // Machine m2 = new Car();  
  
        System.out.println("Car class: ");  
        m2.start();  
        m2.stop();  
  
    }  
}
```

```
Machine class:  
Machine started.  
Machine stopped.  
Car class:  
Machine started.  
Machine stopped.
```

Main.java

Papildome **Car** klasę | Pavyzdys (2)

```
public class Car extends Machine {  
  
    public void turnRight() {  
        System.out.println("Car turned right.");  
    }  
  
    public void turnLeft() {  
        System.out.println("Car turned left.");  
    }  
  
}
```

Car.java

Vaikinė klasė papildoma jai
trūkstamais metodais.

Main klasė | Pavyzdys (2.1)

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Car m2 = new Car();  
  
        System.out.println("Car class: ");  
  
        m2.start();  
        m2.stop();  
  
        m2.turnLeft();  
        m2.turnRight();  
  
    }  
}
```

Car class:
Machine started.
Machine stopped.
Car turned left.
Car turned right.

Main.java

Metodų užklotis / Overriding (1)

- Vaiko klasėje paveldėtus Tėvo klasės metodus galima keisti.
- Turi sutapti:
 - metodų vardai
 - jų antraštės
 - metodų grąžinamos reikšmės tipai (gali būti labiau specifinis tipas)
- Priėjimo modifikatorius išvestinėje klasėje neturi padidinti metodo uždarumo laipsnio
- Užklojantysis metodas neturi mesti naujų *checked exceptions*
- Neužklojami
 - konstruktoriai
 - ***final*** tipo metodai
 - statiniai metodai (bet galima „paslėpti“ statiniu metodu)

Machine ir Car klasės | Pavyzdys (3)

```
public class Machine {  
  
    public void start() {  
        System.out.println  
            ("Machine started.");  
    }  
  
    public void stop() {  
        System.out.println  
            ("Machine stopped.");  
    }  
}
```

Machine.java

```
public class Car extends Machine {  
  
    @Override  
    public void start() {  
        System.out.println  
            ("Car started.");  
    }  
  
    @Override  
    public void stop() {  
        System.out.println  
            ("Car stopped.");  
    }  
  
    public void turnRight() {..  
  
    public void turnLeft() {..  
  
}
```

Car.java

Main klasė | Pavyzdys (3.1)

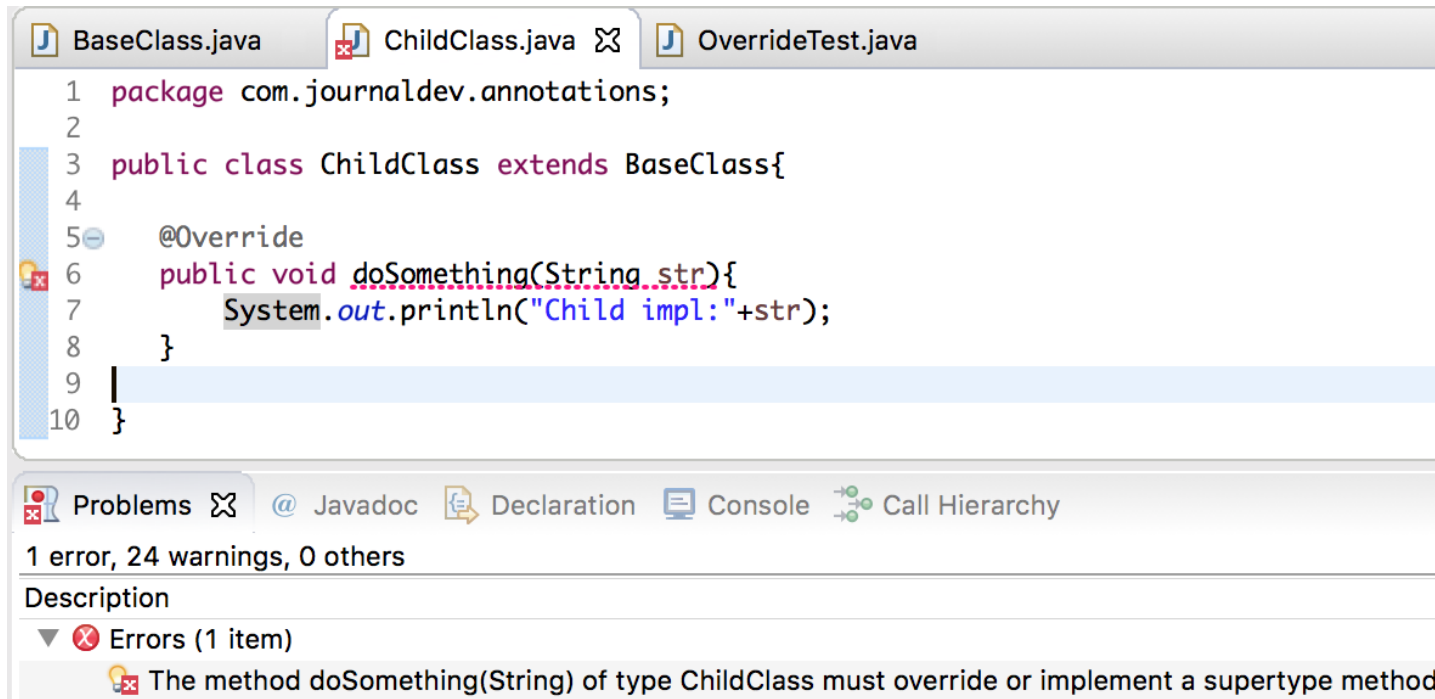
```
public class Main {  
  
    public static void main(String[] args) {  
  
        Machine m1 = new Machine();  
  
        System.out.println("Machine class: ");  
        m1.start();  
        m1.stop();  
  
        Car m2 = new Car();  
  
        System.out.println("Car class: ");  
        m2.start();  
        m2.stop();  
  
    }  
}
```

Main.java

```
Machine class:  
Machine started.  
Machine stopped.  
Car class:  
Car started.  
Car stopped.
```

@Override – anotacija


- Norint pabrėžti, kad metodas yra perrašytas galima nurodyti anotaciją **@Override**, nors tai nėra būtina
- Taip užtikrinama kontrolė, kad metodas perrašytas, o ne naujai sukurtas




```
BaseClass.java  ChildClass.java  OverrideTest.java
1 package com.journaldev.annotations;
2
3 public class ChildClass extends BaseClass{
4
5     @Override
6     public void doSomething(String str){
7         System.out.println("Child impl:"+str);
8     }
9
10 }
```

Problems 1 error, 24 warnings, 0 others

Description

▼  Errors (1 item)

 The method doSomething(String) of type ChildClass must override or implement a supertype method

super

- Perrašančio metodo viduje yra galimybė kviesti perrašomą metodą naudojant žodelį **super**.

```
public class Superclass {  
  
    public void printMethod() {  
        System.out.println("Printed in Superclass.");  
    }  
}
```

```
public class Subclass extends Superclass {  
  
    // overrides printMethod in Superclass  
    public void printMethod() {  
        super.printMethod();  
        System.out.println("Printed in Subclass");  
    }  
    public static void main(String[] args) {  
        Subclass s = new Subclass();  
        s.printMethod();  
    }  
}
```

```
Printed in Superclass.  
Printed in Subclass
```

super

- Norint iškviesti tam tikrą tėvinės klasės konstruktorių, naudojamas žodelis *super(...)*.
- Šis sakinyss turi būti pats pirmasis sakinyss konstruktoriuje.
- Jei toks sakinyss nėra nurodytas, tuomet java kompiliatorius automatiškai jį priskiria - kviečiamas konstruktorius be argumentų.

super

```
class Employee {
    String name;
    String address;

    Employee(String name, String address) {
        this.name = name;
        this.address = address;
    }
}

class Programmer extends Employee {
    String progLanguage;

    Programmer(String name, String address, String progLang) {
        super(name, address);
        this.progLanguage = progLang;
    }
}
```

super

```
public class A {  
    public A() {  
        System.out.println("A's constructor called");  
    }  
}  
public class B extends A {  
    public B() {  
        System.out.println("B's constructor called");  
    }  
}  
public class C extends B {  
    public C() {  
        System.out.println("C's constructor called");  
    }  
}  
...  
public static void main(String[] args) {  
    new C();  
}
```

super

```
public class A {  
    public A() {  
        super();  
        System.out.println("A's constructor called");  
    }  
}  
public class B extends A {  
    public B() {  
        super();  
        System.out.println("B's constructor called");  
    }  
}  
public class C extends B {  
    public C() {  
        super();  
        System.out.println("C's constructor called");  
    }  
}  
...  
public static void main(String[] args) {  
    new C();  
}
```


casting

```
public class A {  
    //statements  
}  
  
public class B extends A {  
    public void foo() {}  
}  
  
A a = new B();  
  
//a.foo();  
//To execute **foo()** method.  
  
((B) a).foo();
```