Juniorprogrammierer.de

Java 11: Inheritance

2024/25 – Sascha Stojanovic

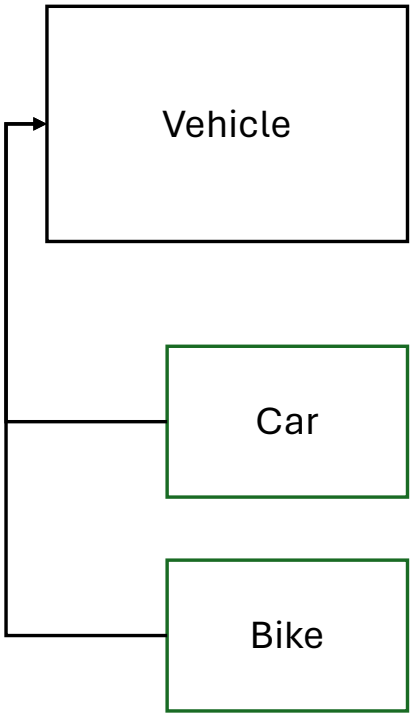
Agenda

- Inheritances
- Super()
- Exercises
- Abstract classes and Methods
- Exercise

Inheritance I

Superclass/Parent

Subclasses/Child




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

class Car extends Vehicle {
  private String modelName = "Mustang";    // Car attribute
  public static void main(String[] args) {

    // Create a myCar object
    Car myCar = new Car();

    // Call the honk() method (from the Vehicle class) on the myCar object
    myCar.honk();
```

Inheritance II

- You can also overwrite inherited methods

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

class Pig extends Animal {
  public void animalSound() {
    System.out.println("The pig says: wee wee");
  }
}

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

class Main {
  public static void main(String[] args) {
    Animal myAnimal = new Animal();  // Create a Animal object
    Animal myPig = new Pig();  // Create a Pig object
    Animal myDog = new Dog();  // Create a Dog object
    myAnimal.animalSound();
    myPig.animalSound();
    myDog.animalSound();
  }
}
```

Super-keyword

- Keyword “super” is used to access parental-class methods
- Keyword “super()” accesses the parental-class constructor
- If parent has a constructor without importingparameter it is set automatically in childrens constructor

```java
public class Vehicle {
    private final int numberEngine = 1;
    private int numberWheels;

    public void honk(){
        System.out.println(x:"tut tuuut!");
    }

    public Vehicle(){
        setNumberWheels(numberWheels:0);
    }

    public Vehicle(int numberWheels){
        setNumberWheels(numberWheels);
    }
```

```java
package Inheritence;

public class Motorbike extends Vehicle{

    public Motorbike(){
        super(numberWheels:2);
    }

    public void honk(){
        super.honk();
        System.out.println(x:"Miiib miiib");
    }
```

```java
package Inheritence;

public class InheritanceMain {

    Run | Debug
    public static void main(String[] args) {
        Motorbike dukati = new Motorbike();
        System.out.println(dukati.getNumberEngine());
        System.out.println(dukati.getNumberWheels());
        dukati.honk();
    }
}
```

Exercise “add the car to the vehicle class”

- Copy the Vehicle.java and create a InheritanceMain.java with a corresponding static main method
- Create Car.java
  - Make Car.java a child of Vehicle.java
- In the InheritanceMain.java create a car-object and call all relevant methods available
- Next create in Car.java an empty constructor
  - Discuss with or without “super()”;
  - Add super() and importingparameters if needed
- Last but not least play around with the honk-method in the Car.java
- Now discuss with Sascha the advantages of parental and child-classes
- Do you see a problem here? Think about creating an object of the …java

Abstract

- You can only declare abstract methods in an abstract class
- You cant create an object of an Abstract class, so these classes are used to slimmen their child-classes
- Next to classes you can also define abstract-methods
  - The have no content
  - All children need them but they have different content
  - Child without parental abstract class definition shows and error

Inheritence > J Vehicle.java > ...

```
3   public abstract class Vehicle {
29      }
30
31      // Define an abstract method
32      public abstract void startEngine();
33  }
34
```



```
1   package Inheritence;
2
3   public class Motorbike extends Vehicle{
4
5       public Motorbike(){
6           super(numberWheels:2);
7       }
8
9       public void honk(){
10          super.honk();
11          System.out.println(x:"Miiib miiib");
12      }
13
14      public void startEngine(){
15          System.out.println(x:"Start with kickstarter");
16      }
```

Exercise

- Change your Vehicle.java into an abstract class
- Add the abstract void method “startEngine()”
- Adapt the cars.java accordingly