KU LEUVEN

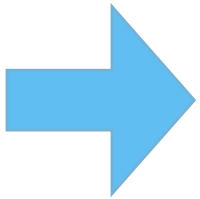
Overerving & Polymorfisme

joost.vennekens@kuleuven.be



Overerving

- Sommige klassen zijn **speciaal geval** van andere klasse
- Docent is een **speciaal geval** van werknemer, dwz. elke docent is ook werknemer
- Functionaliteit van docent = functionaliteit van werknemer + extra's



Hergebruik van code

```java
public class Werknemer {
    private Datum geboorteDatum;
    private String naam;
    public Werknemer() {
    }
    public String getNaam() {
        return naam;
    }
    public void setNaam(String n) {
        naam = n;
    }
}
```

```java
public class Docent {
    private Datum geboorteDatum;
    private String naam;
    public Docent() {
    }
    public String getNaam() {
        return naam;
    }
    public void setNaam(String n) {
        naam = n;
    }
    private Vak[] vakken;
    public boolean doceert(Vak v) {
        for (int i = 0; i < vakken.length; i++)
            if (vakken[i].equals(v))
                return true;
        return false;
    }
}
```

```java
public class Werknemer {
    private Datum geboorteDatum;
    private String naam;
    public Werknemer() {
    }
    public String getNaam() {
        return naam;
    }
    public void setNaam(String n) {
        naam = n;
    }
}
```

```java
public class Docent extends Werknemer {

    public Docent() {
    }

    private Vak[] vakken;
    public boolean doceert(Vak v) {
        for (int i = 0; i < vakken.length; i++)
            if (vakken[i].equals(v))
                return true;
        return false;
    }
}
```

```java
Docent d = new Docent();
d.setNaam("Joost Vennekens");
System.out.println(d.getNaam());
```


KU LEUVEN

```
public class ComplexGetal {
    private double reeel;
    private double imaginair;
    public ComplexGetal(double r, double i) {
        reeel = r;
        imaginair = i;
    }
    public ComplexGetal plus(ComplexGetal ander) {
        return new ComplexGetal(ander.getReeel() + getReeel(),
                                ander.getImaginair() + getImaginair());
    }
}
public class ReeelGetal extends ComplexGetal {
    public ReeelGetal(double getal) {
        super (getal, 0);
    }
}
```

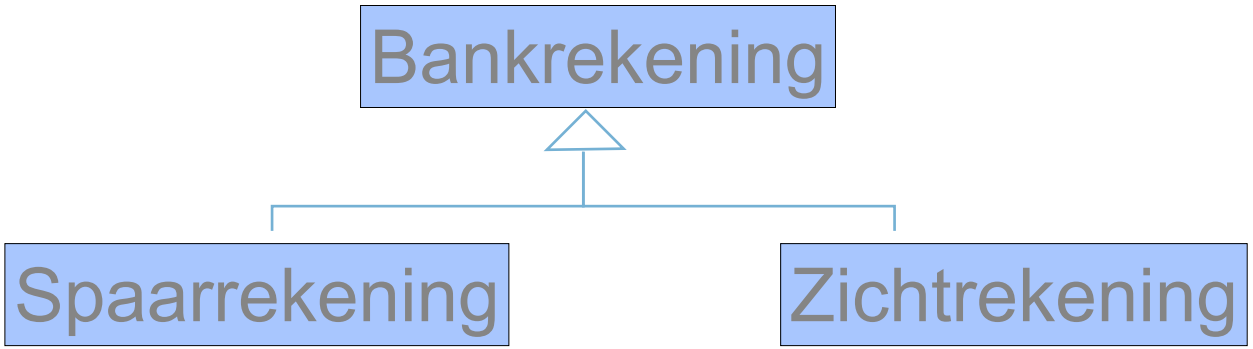


```
ReeelGetal r = new ReeelGetal(5.0);
ReeelGetal s = new ReeelGetal(5.0);
ComplexGetal som = r.plus(s);
```


KU LEUVEN

Superklasse

Subklasse




```
public class Bankrekening {
    ...
}

public class Zichtrekening extends Bankrekening {
    ...
}

public class Spaarrekening extends Bankrekening {
    ...
}
```


KU LEUVEN

```
public class Rechthoek {

    private double breedte;
    private double hoogte;

    public Rechthoek(double h, double b) {
        breedte = b;
        hoogte = h;
    }
    public double oppervlakte() {
        return breedte * hoogte;
    }
    public void draai() {
        double tmp = breedte;
        breedte = hoogte;
        hoogte = tmp;
    }
}
```

```
public class Vierkant extends Rechthoek {
    public Vierkant(double grootte) {
        super (grootte, grootte);
    }
}
```

```
Vierkant v = new Vierkant(7.5);
v.draai();
```

Overriding

```
public class Rechthoek {

    private double breedte;
    private double hoogte;

    public Rechthoek(double h, double b) {
        breedte = b;
        hoogte = h;
    }
    public double oppervlakte() {
        return breedte * hoogte;
    }
    public void draai() {
        double tmp = breedte;
        breedte = hoogte;
        hoogte = tmp;
    }
}
```

```
public class Vierkant extends Rechthoek {
    public Vierkant(double grootte) {
        super (grootte, grootte);
    }
    public void draai() {
    }
}
```

```
Vierkant v = new Vierkant(7.5);
v.draai();
```

Polymorfisme

- Verschillende methodes mogen dezelfde naam hebben als hun argumenten verschillen
- Geldt ook voor impliciete argument “this”
- De **meest specifieke** methode die van toepassing is wordt uitgevoerd

Dynamische typering

- Een **variabele** heeft een type
  - Statisch gedeclareerd

    ```
    Bankrekening mijnRekening;
    ```

- Een **object** heeft een type
  - Dynamisch bepaald

    ```
    mijnRekening = new Zichtrekening();
    ```

← foutendetectie

← selectie methode

Overeenkomst

- Type van een object moet een **subklasse** zijn van type van de variable

```
Bankrekening mijnRekening = new Bankrekening();
Bankrekening mijnRekening = new Zichtrekening();
Zichtrekening mijnRekening = new Bankrekening();

Vierkant x = new Rechthoek();
Rechthoek x = new Vierkant();
Vierkant x = new Vierkant()
```

- Gelijkaardig voor primitieve types

```
int a = 9.2;
double a = 9;
```


KU LEUVEN

```
public class Rechthoek {
   protected double breedte; protected double hoogte;  ...
   public boolean pastIn(Rechthoek ander) {
      return (this.breedte <= ander.breedte) &&  (this.hoogte <= ander.hoogte) ||
                          (this.breedte <= ander.hoogte) &&  (this.hoogte <= ander.breedte) ;
    }
    public boolean pastIn(Vierkant ander) {
       return (this.breedte <= ander.breedte) && (this.hoogte <= ander.breedte) ;
    }
}
public class Vierkant extends Rechthoek {
    ...
    public boolean pastIn(Vierkant ander) {
       return (this.breedte <= ander.breedte);
    }
    public boolean pastIn(Rechthoek ander) {
       return (this.breedte <= ander.breedte)
              && (this.breedte <= ander.hoogte) ;
    }
}
```

```
Vierkant x = new Vierkant(7);
Rechthoek y = new Vierkant(2);
Rechthoek z = new Rechthoek(3,4);
x.pastIn(x);
y.pastIn(x);
z.pastIn(x);
x.pastIn(y);
```

Dynamische type Statische type

KU LEUVEN

Kwis

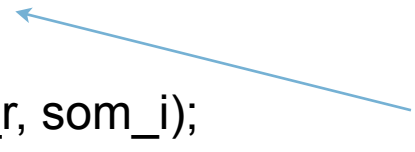
```
class X {
    public void a() {
        System.out.println("x");
    }
}

class Y extends X {
    public void a() {
        System.out.println("y");
    }
}

class Z extends Y {
    public void a() {
        System.out.println("z");
    }
```

```
public static void main(String[] args) {
        X x = new X();
        x.a();
        x = new Z();
        x.a();
        Y y = new Z();
        y.a();
        y = new Y();
        y.a();
    }
}
```

```
public class ComplexGetal {
    ...
    public ComplexGetal plus(ComplexGetal ander) {
        return new ComplexGetal(ander.getReeel() + getReeel(),
                                ander.getImaginair() + getImaginair());
    }
}
public class ReeelGetal extends ComplexGetal {
    ...
    public ReeelGetal plus(ReeelGetal ander) {
        return new ReeelGetal(ander.getReeel() + getReeel());
    }
}
```

```
ReeelGetal r = new ReeelGetal(5.0);
ReeelGetal s = new ReeelGetal(5.0);
ReeelGetal som = r.plus(s);
```

```
public class ComplexGetal {
    ...
    public ComplexGetal plus(ComplexGetal ander) {
        double som_r = ander.getReeel() + getReeel();
        double som_i = ander.getImaginair() + getImaginair();
        if (som_i == 0)
            return new ReeelGetal(som_r);
        else
            return new ComplexGetal(som_r, som_i);
    }
}
public class ReeelGetal extends ComplexGetal {
    ...
    public ReeelGetal plus(ReeelGetal ander) {
        return new ReeelGetal(ander.getReeel() + getReeel());
    }
}
```



Mag dit wel?

```
public class ComplexGetal {
    ...
    public ComplexGetal plus(ComplexGetal ander) {
        double som_r = ander.getReeel() + getReeel();
        double som_i = ander.getImaginair() + getImaginair();
        if (som_i == 0)
            return new ReeelGetal(som_r);
        else
            return new ComplexGetal(som_r, som_i);
    }
}
public class ReeelGetal extends ComplexGetal {
    ...
    public ReeelGetal plus(ReeelGetal ander) {
        return new ReeelGetal(ander.getReeel() + getReeel());
    }
}
```

Mag dit wel?

```
ComplexGetal r = new ComplexGetal(1,-1);
ComplexGetal s = new ComplexGetal(1,1);
ReeelGetal som = r.plus(s);
```

Typecasting

- Soms weten wij dat een variabele een **strenger** type heeft dan de compiler kan afleiden

```
ComplexGetal r = new ComplexGetal(1,-1);
ComplexGetal s = new ComplexGetal(1,1);
ReeelGetal som = (ReeelGetal) r.plus(s);
```


KU LEUVEN

Ingebouwde klassen & methodes

KU LEUVEN

Elke klasse erft over van Object

```
public class Bla { ... }
```

```
public class Bla extends Object { ... }
```

Elke klasse heeft **exact één** superklasse

```
public class Lijst {
    private Element eerste;
    public void voegtoe(Object o) {
        if (eerste != null)
            eerste.voegtoe(o);
        else eerste = new Element(o);
    }
    public String toString() {
        Element huidige = eerste;
        String resultaat = "";
        while (huidige != null) {
            resultaat += huidige.toString();
            resultaat += " ";
            huidige = huidige.getVolgende();
        }
        return resultaat;
    }
}
```

```
class Element {
    private Object waarde;
    private Element volgende;
    private void setVolgende(Element v) {
        volgende = v;
    }
    Element getVolgende() {
        return volgende;
    }
    Element(Object i) {
        waarde = i;
    }
    void voegtoe(Object i) {
        if (volgende == null)
            setVolgende(new Element(i));
        else getVolgende().voegtoe(i);
    }
    public String toString() {
        return "" + waarde;
}}
```

```
public class Lijst {
    ...
    public void voegtoe(Object o) {
        ...
    }
    public String toString() {
        ...
    }
}
```

```
public class Boek {
    private String titel;
    Boek(String t) {
        titel = t;
    }
    public String toString() {
        return "\"" + titel + "\"";
    }
}
```

```
public class Bibliotheek {
    private Lijst boeken;
    public Bibliotheek() {
        boeken = new Lijst();
        boeken.voegtoe(new Boek("Catcher in the Rye"));
    }
    public static void main(String[] a) {
        System.out.println((new Bibliotheek()).boeken);
    }
}
```


KU LEUVEN

```
public class Lijst {
    ...
    public void voegtoe(Object o) {
        ...
    }
    public String toString() {
        ...
    }
}
```

```
public class Vermenigvuldigingstafel {
    private Lijst tafel;
    public Vermenigvuldigingstafel(int g) {
        tafel = new Lijst();
        for (int i = 1; i <= 10; i++) {
            tafel.voegtoe(new Integer(g*i));
        }
    }
    public Lijst getTafel() {
        return tafel;
    }
    public static void main(String[] a) {
        int getal = Integer.parseInt(a[0]);
        Vermenigvuldigingstafel t;
        t = new Vermenigvuldigingstafel(getal);
        System.out.println(t.getTafel());
    }
}
```

Of met array

```
public static void main(String[] a) {
    Boek[] lijst = new Boek[](10);
    for (int i = 0; i < lijst.length; i++)
          lijst[i] = new Boek();
}
```


KU LEUVEN

Elke klasse erft over van Object

```
public class Object {

    public boolean equals(Object other) {
        ...
    }
    public String toString() {
        ...
    }
   ...
}
```

Vergelijkingen

- Java heeft vergelijkingsoperatoren

  ==, !=, >, <, <=, >=
- Zoals verwacht bij primitieve types
- **Let op:** bij objecten testen == en != of het om **hetzelfde** object gaat

```
ReeelGetal r = new ReeelGetal(12);
ReeelGetal s = new ReeelGetal(12);
if (r == s) System.out.println("gelijk");
if ("foo" == "foo") System.out.println("gelijk");
```


KU LEUVEN

Vergelijkingen

- Om de **inhoud** van objecten te vergelijken

```
public boolean equals(Object);
```

- Bijvoorbeeld

```
if ("foo".equals("foo")) System.out.println("gelijk");
Integer i = new Integer(12);
Integer j = new Integer(12);
if (i.equals(j)) System.out.println("gelijk");
```

- Voor eigen klassen, zelf implementeren

KU LEUVEN

Verschilpunten tussen objecten en primitieve types

- **new** operator
- methode oproepen
- == en !=
- Call-by-value vs call-by-reference

Strings zijn objecten

Arrays zijn ongeveer objecten

```
public class Test {
  private int waarde;
  public static void veranderPrimitief(int x) {
     x = 99;
  }
  public static void veranderObject(Test x) {
    x.waarde = 99;
  }
  public Test(int i) {
    waarde = i;
  }
  public static void main(String[] args) {
    int i = 1;
    veranderPrimitief(i);
    System.out.println(i);
    Test object = new Test(1);
    veranderObject(object);
    System.out.println(object.waarde);
  }
}
```

Afprinten

```
public class ComplexGetal {
    private double reeel;
    private double imaginair;
    public ComplexGetal(double r, double i) {
        reeel = r;
        imaginair = i;
    }
    public static void main(String [] args) {
        ComplexGetal x = new ComplexGetal(7,8);
        System.out.println(x);
    }
}
```

```
bash-3.2$ java ComplexGetal
ComplexGetal@66848c
bash-3.2$
```


KU LEUVEN

Afprinten

```
public class ComplexGetal {
    private double reeel;
    private double imaginair;
    public ComplexGetal(double r, double i) {
        reeel = r;
        imaginair = i;
    }
    public static void main(String [] args) {
        ComplexGetal x = new ComplexGetal(7,8);
        System.out.println(x);
    }
    public String toString() {
        return  reeel + " + " + imaginair + "i";
    }
    public boolean equals(Object o) {
       if (o instanceof ComplexGetal) {
          ComplexGetal g = (ComplexGetal) o;
          return g.reeel == reeel && g.imaginair == imaginair;
       }
       else return  false;
    }
}
```

```
bash-3.2$ java ComplexGetal
7.0 + 8.0i
bash-3.2$
```


KU LEUVEN

Inlezen

```
import java.io.*;
class PersonalHello {
    public static void main (String args[])   {
        byte[] name = new byte[100];
        int nr_read = 0;
        System.out.println("What is your name?");
        try {
            nr_read = System.in.read(name);
            System.out.print("Hello ");
            System.out.write(name,0,nr_read);
        }
        catch (IOException e) {
            System.out.print("I'm Sorry.  I didn't catch your name.");
        }
    }
}
```


KU LEUVEN

Inlezen (2)

```
import java.io.*;
class Addition {
    public static void main (String[] args) throws IOException {
        BufferedReader stdin = new BufferedReader(
                        new InputStreamReader(System.in) );
        int num1, num2;
        System.out.print ("Enter an integer number: ");
        num1 = Integer.parseInt(stdin.readLine());
        System.out.print ("Enter another integer number: ");
        num2 = Integer.parseInt(stdin.readLine());
        System.out.println ("The sum is " + (num1+num2));
    }
}
```


KU LEUVEN