



Java Essentials

Hoofdstuk 9

Interfaces

**DE HOGESCHOOL
MET HET NETWERK**

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



Inhoud

1. Inleiding
2. Een interface definiëren
3. Een interface implementeren in een klasse
4. Interface als datatype
5. Samenvatting



1. Inleiding

Definitie

Interface is een soort abstracte klasse die alleen abstracte methoden en constanten (final eigenschappen) bevat.

Doel

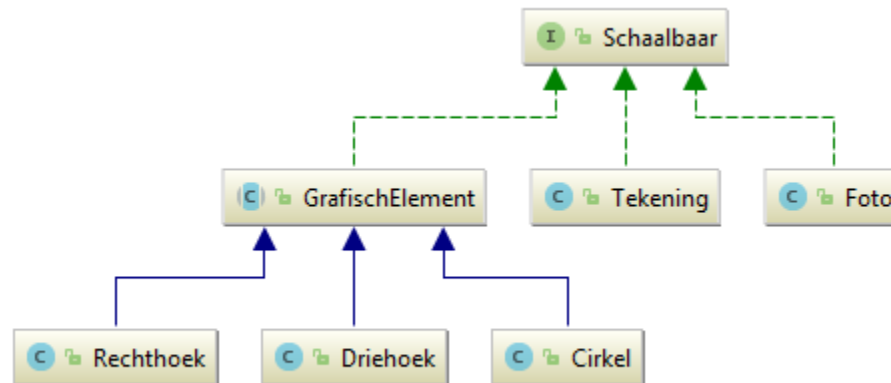
Aan bepaalde klassen methoden opleggen die moeten geïmplementeerd worden



Voorbeeld: grafisch programma

=> hoe er zeker van zijn dat alle grafische objecten herschaald kunnen worden

=> gebruik maken van interface Schaalbaar



=> door een interface is het mogelijk om verschillende objecten toch een gemeenschappelijk gedrag te geven

2. Een interface definiëren

2.1 Declaratie van de interface

```
public abstract interface InterfaceNaam extends SuperInt1, SuperInt2  
{  
    ...  
}
```

Vet gedrukt = verplicht

Interface kan afgeleid zijn van meerdere superinterfaces

Voorbeeld:

```
public interface Schaalbaar {  
    ...  
}
```

Toegangsniveau enkel public of package



2.1 De beschrijving van de interface

= declaratie van alle methoden en constanten

```
public interface Schaalbaar {
```

```
// constanten
```

```
public static final int DUBBEL = 200;
```

```
public static final int HELFT = 50;
```

```
public static final int VIERDE = 25;
```

```
// methoden
```

```
public abstract void herschaal(int factor);
```

```
}
```

alleen finale eigenschappen

Abstracte methode:

- wel definitie
- geen implementatie

Alle methoden zijn impliciet
`public` en `abstract`



Impliciete gegevens mogen
weggelaten worden

Alle variabelen zijn impliciet
`public`, `static` en `final`

```
public interface Schaalbaar {  
    // constanten  
    int DUBBEL = 200;  
    int HELFT = 50;  
    int VIERDE = 25;  
  
    // methoden  
    void herschaal(int factor);  
}
```

3. Een interface implementeren in een klasse

Bij declaratie van de klasse wordt aangegeven welke interfaces geïmplementeerd worden.

```
public class KlasseNaam implements InterfaceNaam {  
...  
}
```

Voorbeeld

```
public class Rechthoek extends GrafischElement implements  
Schaalbaar {  
...  
}
```

- Alle methoden gedefinieerd in de interface Schaalbaar **moeten** geïmplementeerd worden in de klasse Rechthoek.
- De klasse Rechthoek kan gebruik maken van alle constanten gedefinieerd in de interface Schaalbaar.


```
public class Rechthoek extends
GrafischElement implements Schaalbaar {

private int hoogte;
private int breedte;

public Rechthoek (int x, int y,
                  int h, int b) {
    super(x, y);
    this.hoogte = h;
    this.breedte = b;
}

//andere methoden

public void herschaal(int factor) {
    hoogte = hoogte * factor / 100;
    breedte = breedte * factor / 100;
}

}
```

Rechthoek is een subklasse van
de abstracte klasse
GrafischElement

implements

Rechthoek maakt gebruik van
de interface Schaalbaar



```
public class Rechthoek extends  
GrafischElement implements Schaalbaar {
```

```
private int hoogte;  
private int breedte;
```

```
public Rechthoek (int x, int y,  
                  int h, int b) {  
    super(x, y);  
    this.hoogte = h;  
    this.breedte = b;  
}
```

```
//andere methoden
```

```
public void herschaal(int factor) {  
    hoogte = hoogte * factor / 100;  
    breedte = breedte * factor / 100;  
}  
  
}
```

Eigenschappen

- x, y overgeërfd van GrafischElement
- hoogte
- breedte



```
public class Rechthoek extends
GrafischElement implements Schaalbaar {

private int hoogte;
private int breedte;

public Rechthoek (int x, int y,
                  int h, int b) {
    super(x, y);
    this.hoogte = h;
    this.breedte = b;
}

//andere methoden

public void herschaal(int factor) {
    hoogte = hoogte * factor / 100;
    breedte = breedte * factor / 100;
}

}
```

constructor



```
public class Rechthoek extends  
GrafischElement implements Schaalbaar {
```

```
    private int hoogte;  
    private int breedte;
```

```
    public Rechthoek (int x, int y,  
                      int h, int b) {  
        super(x, y);  
        this.hoogte = h;  
        this.breedte = b;  
    }
```

```
//andere methoden
```

```
    public void herschaal(int factor) {  
        hoogte = hoogte * factor / 100;  
        breedte = breedte * factor / 100;  
    }  
  
}
```

Andere methoden

waaronder de implementatie
van de abstracte methoden
gedefinieerd in de abstracte
klasse GrafischElement



```
public class Rechthoek extends  
GrafischElement implements Schaalbaar {  
  
    private int hoogte;  
    private int breedte;  
  
    public Rechthoek (int x, int y,  
                      int h, int b) {  
        super(x, y);  
        this.hoogte = h;  
        this.breedte = b;  
    }  
  
    //andere methoden  
  
    public void herschaal(int factor) {  
        hoogte = hoogte * factor / 100;  
        breedte = breedte * factor / 100;  
    }  
  
}
```

Abstracte methodes uit
interface Schaalbaar

moeten geïmplementeerd
worden



Implementatie van meerdere interfaces

```
public class KlasseNaam implements InterfaceNaam1,  
InterfaceNaam2 {  
...  
}
```

Voorbeeld

```
public class Rechthoek extends GrafischElement implements  
Schaalbaar, Tekenbaar {  
...  
}
```

- Alle methoden gedefinieerd in de interface Schaalbaar en Tekenbaar moeten geïmplementeerd worden in de klasse Rechthoek.
- De klasse Rechthoek kan gebruik maken van alle constanten gedefinieerd in de interface Schaalbaar en Tekenbaar.



Implementatie van meerdere interfaces

= meervoudige overerving (=multiple inheritance)

Bij klassen enkelvoudige overerving!!!!

= elke klasse heeft exact één directe superklasse



Voorbeeld

```
public class Dier{                                // superklasse is Object
}
```

```
public class Hond extends Dier{                  // superklasse is Dier
}
```

```
class VreemdDing extends Dier, Auto{            // FOUT
}
```

```
public interface Schaalbaar{
    // constanten
    int DUBBEL = 200;
    int HELFT = 50;
    int VIERDE = 25;

    // methoden
    void herschaal(int factor);
}
```

```
public interface Tekenbaar{
    // methoden
    void teken();
}
```

```
public class Rechthoek extends GrafischElement
implements Schaalbaar, Tekenbaar {

    private int hoogte;
    private int breedte;

    public Rechthoek (int x, int y,
                      int h, int b) {
        super(x, y);
        this.hoogte = hoogte;
        this.breedte = breedte;
    }

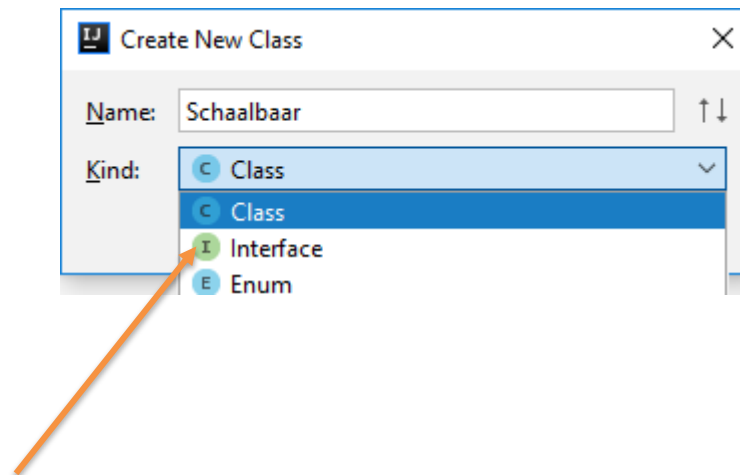
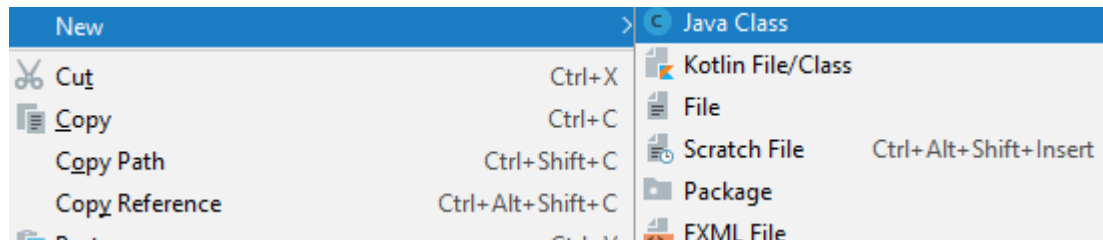
    //andere methoden

    public void herschaal(int factor){
        height = height * factor / 100;
        width = width * factor / 100;
    }

    public void teken(){
        System.out.printf("x:%d y:%d h:%d w:%d",
                          super.getX(), super.getY(), hoogte,
                          breedte);
    }
}
```


Interfaces in IntelliJ

RMK op package, New, Java Class



public class Cirkel extends GrafischElement **implements** Schaalbaar

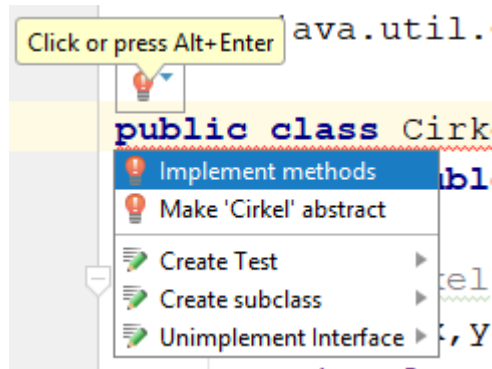
```
public class Cirkel extends GrafischElement implements Schaalbaar {  
    private double straal;  
}
```

Fout: Cirkel heeft nog niet alle methodes uit Schaalbaar geïmplementeerd

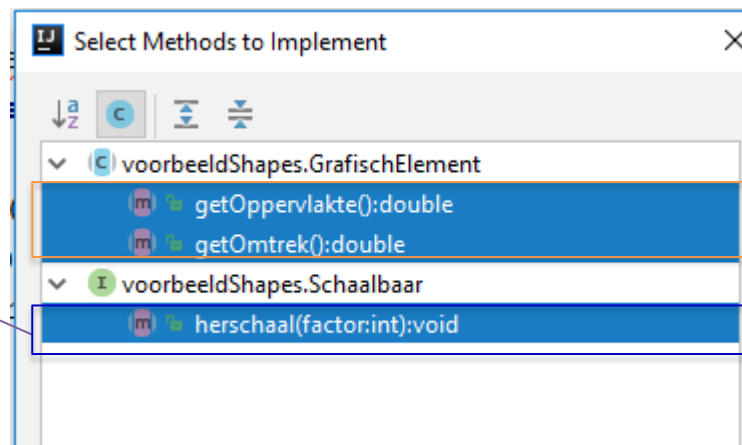
Klik op de onderlijnde lijn

```
public class Cirkel extends GrafischElement implements Schaalbaar{  
    private double straal;
```

Klik op



Abstracte methoden uit de interface Schaalbaar



Abstracte methoden uit de klasse GrafischElement

6. De interface als data-type

Tot nu toe: polymorfisme

= **variabele van een superklasse** kan verwijzen naar een **object van een klasse afgeleid van deze superklasse.**

```
Rechthoek rechthoek = new Rechthoek();  
GrafischElement vorm = new Rechthoek();
```

Nieuwe vorm van polymorfisme:

= **variabele van een interface** kan verwijzen naar een **object van een klasse die deze interface implementeert.**

```
Schaalbaar schaalbaar = new Rechthoek();
```



```

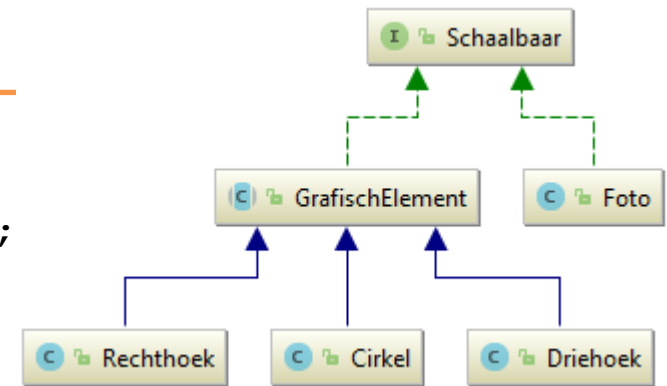
public class Tekenprogramma {
    public static void main(String[] args) {
        Schaalbaar[] tek = new Schaalbaar[4];
        tek[0] = new Foto();
        tek[1] = new Cirkel(3, 2, 1);
        tek[2] = new Driehoek();
        tek[3] = new Rechthoek(5, 10, 2, 3);

        for (int i = 0; i < tek.length; i++) {
            tek[i].herschaaal(50);
        }

        tek[3].setHoogte(20);    // foutmelding, oplossing?

        if (tek[1] instanceof Schaalbaar) {
            System.out.println("dit object implementeert de
                               interface Schaalbaar");
        }
    }
}

```



```

public class Tekenprogramma {
    public static void main(String[] args) {
        Schaalbaar[] tek = new Schaalbaar[4];
        tek[0] = new Foto();
        tek[1] = new Cirkel(3, 2, 1);
        tek[2] = new Driehoek();
        tek[3] = new Rechthoek(5, 10, 2, 3);

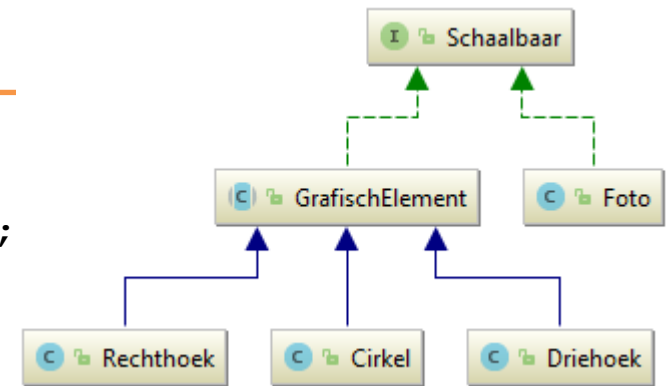
        for (int i = 0; i < tek.length; i++) {
            tek[i].herschaaal(50);
        }

        (Rechthoek) tek[3].setHoogte(20);

        if (tek[1] instanceof Schaalbaar) {
            System.out.println("dit object implementeert de
                                interface Schaalbaar");
        }
    }
}

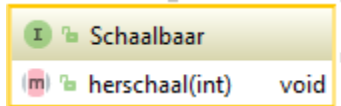
```

casting



Opdracht

Interfaces

1. Maak een module H9 aan.
Maak hierin een package `be.pxl.h9.opdracht` aan.
Importeer hierin de klassen `Rechthoek`, `Vierkant`, `Driehoek`, `Cirkel` en `GrafischElement` vanuit de module H6, package `be.pxl.h6.opdracht8`
2. Maak de Interface `Schaalbaar` aan.
3. Implementeer de interface `Schaalbaar` voor alle figuren.
Opmerking : De oppervlakte van elke figuur moet met de opgegeven factor vergroten/verkleinen.
4. Maak een programma `VormApp` dat een `arrayList` van `GrafischeElementen` aanmaakt.
Steek in deze `arrayList` een `Cirkel`, een `Rechthoek`, een `Driehoek` en een `Vierkant`.
Verdubbel de schaal van de figuren en druk de oorspronkelijke oppervlakte en de gewijzigde oppervlakte af.

4. Samenvatting

- Interface is een verzameling van abstracte methoden die in een klasse die deze interface gebruikt geïmplementeerd moeten worden.
- Interface kan daarnaast ook constanten bevatten.
- Mbv interface nieuw soort datatype
=> objecten die heel verschillend zijn, kunnen toch op dezelfde wijze behandeld worden

