Java Essentials

Hoofdstuk 2

Klassen definiëren

DE HOGESCHOOL
MET HET NETWERK

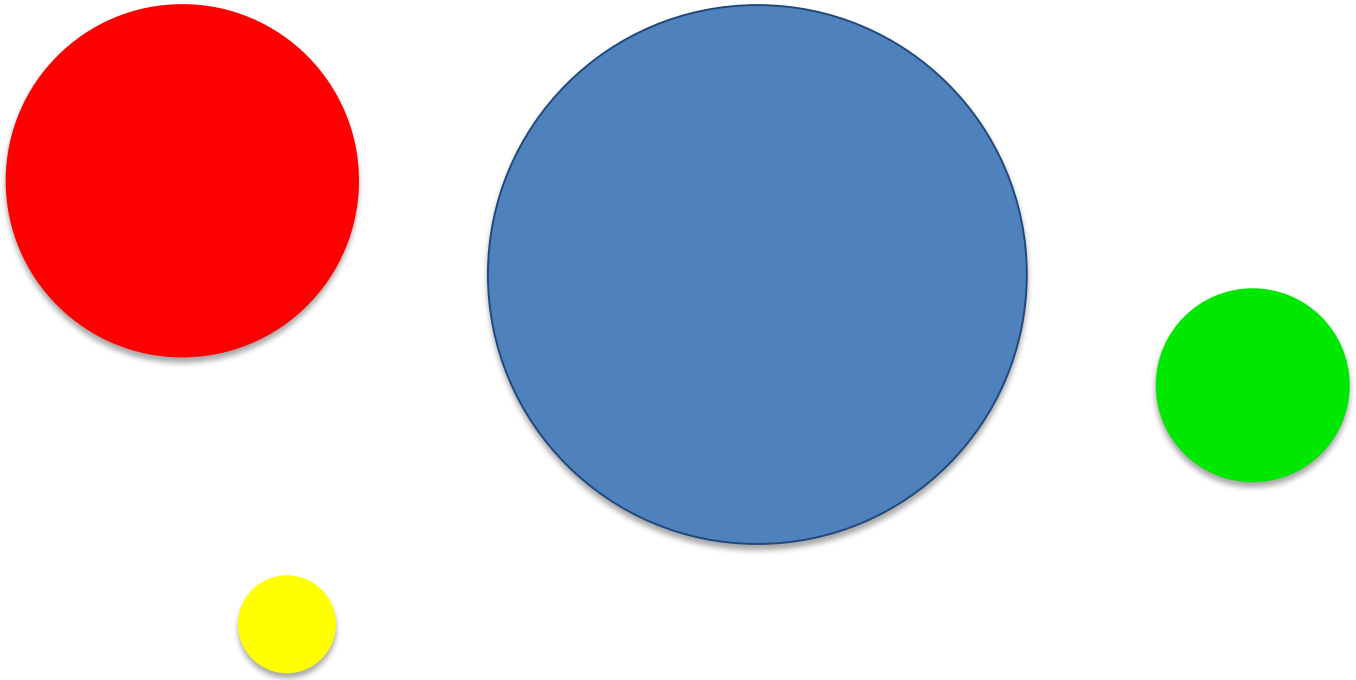
Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Inhoud

1. Inleiding
2. De declaratie van de klasse
3. De klasse-omschrijving (body)
    1. Eigenschappen
    2. Methoden
    3. Constructors
    4. Instantievariabelen en klassevariabelen
    5. Instantiemethoden en klassemethoden

1. Inleiding

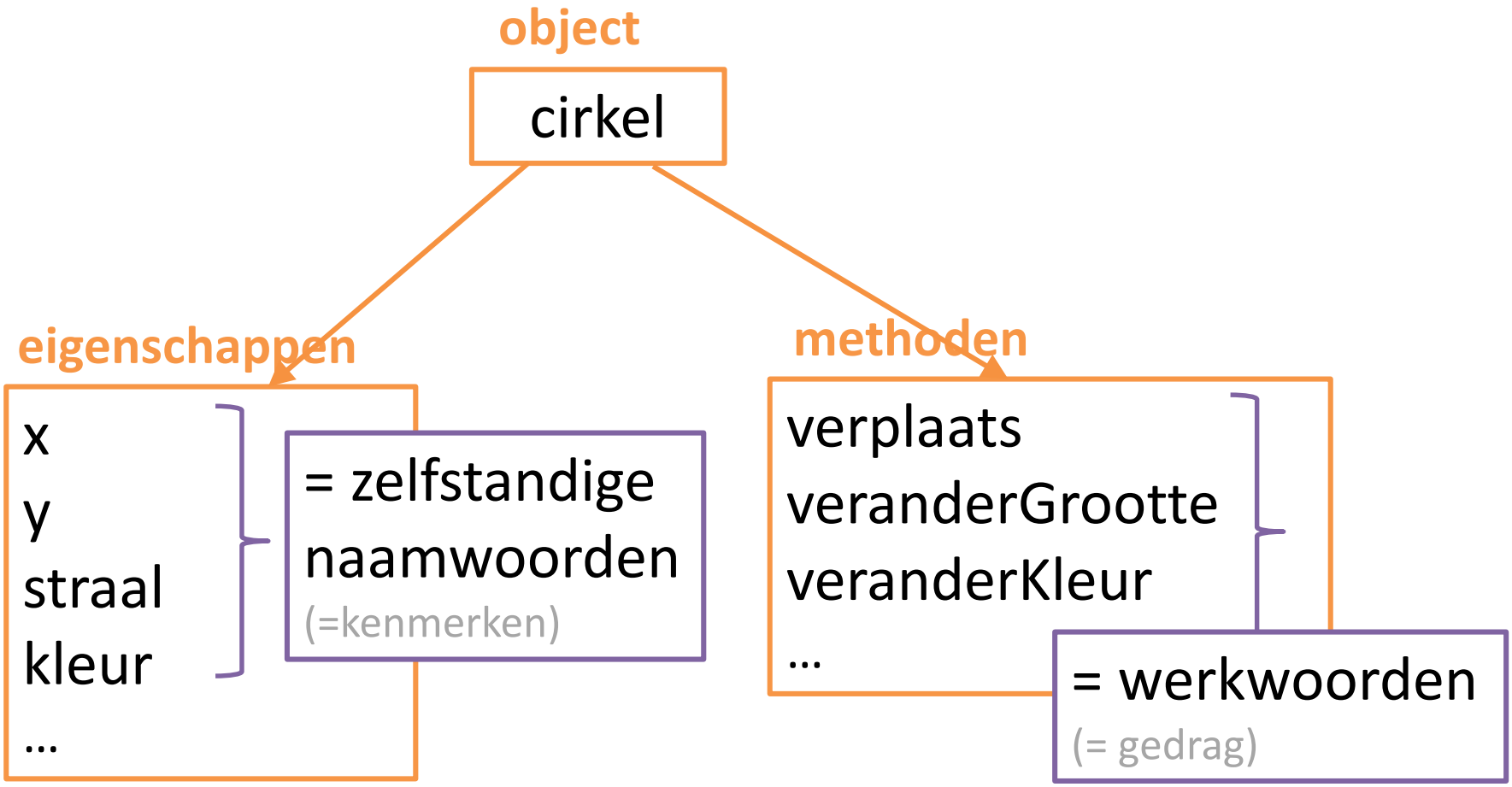
Voorbeeld:



Elke cirkel kan je beschouwen als een **object**.

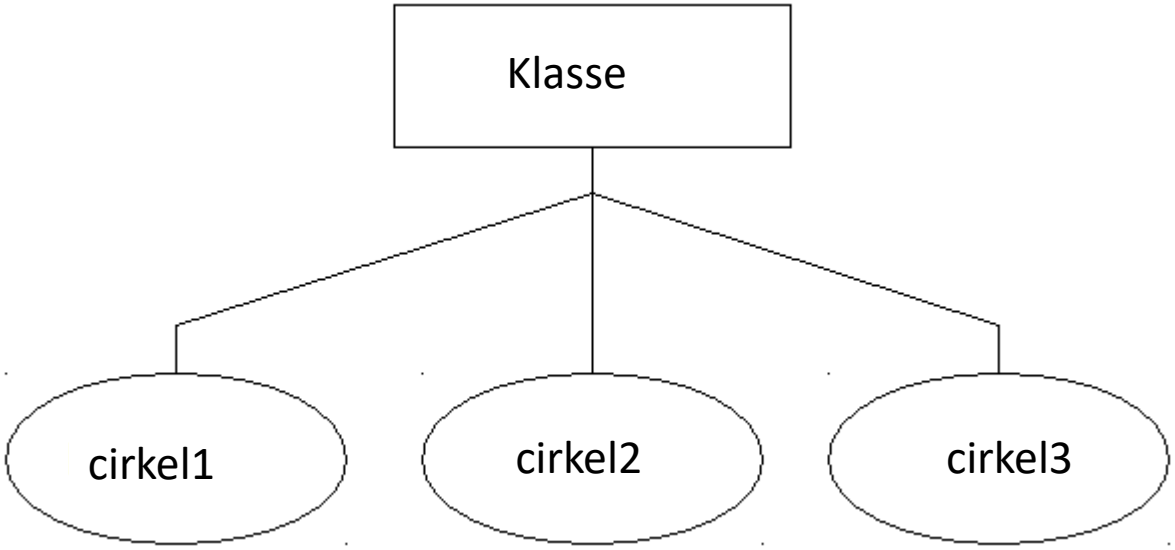
Deze cirkels hebben een aantal gemeenschappelijke kenmerken

1. Inleiding


object
cirkel
eigenschappen
x
y
straal
kleur
…
= zelfstandige naamwoorden
(=kenmerken)
methoden
verplaats
veranderGrootte
veranderKleur
…
= werkwoorden
(= gedrag)

1. Inleiding

Al deze cirkels zijn gefabriceerd o.b.v. een blauwdruk (= klasse).



Een **klasse** is een blauwdruk voor objecten
die de eigenschappen en methoden van objecten van dezelfde soort bepaalt.

1. Inleiding

Zelf klassen maken: vb Rechthoek

Eigenschappen:

```
hoogte
breedte
x
y
```

Methoden:

```
setHoogte()
setBreedte()
setPositie()
getHoogte()
getBreedte()
getOppervlakte()
getOmtrek()
```

2. De declaratie van de klasse

```
package …;

public class Rechthoek {
…
}
```

Dit is de ‘meest eenvoudige’ klasse-declaratie. Hier kunnen ook nog andere (optionele) componenten staan

Geen `main()`-methode. Een programma bestaat meestal uit meerdere klassen waarvan er maar 1 de methode `main()` heeft.

Opdracht 1: *Een klasse maken*

Maak een klasse met de naam **Rechthoek.java** in het pakket be.pxl.h2.opdracht

```
package be.pxl.h2.opdracht;
public class Rechthoek {
    …
}
```

(de klasse Rechthoek gaan we steeds verder uitbreiden met eigenschappen en methoden)

Opdracht 2: *Een hoofdprogramma maken*

- Maak een hoofdprogramma met de naam **RechthoekApp**
- Maak een object van de klasse *Rechthoek* (m.b.v. *new*).

```
public class RechthoekApp {
    public static void main(String[] args) {
        System.out.println("Dit programa maakt een rechthoek");
        Rechthoek rechthoek = new Rechthoek();
    }
}
```

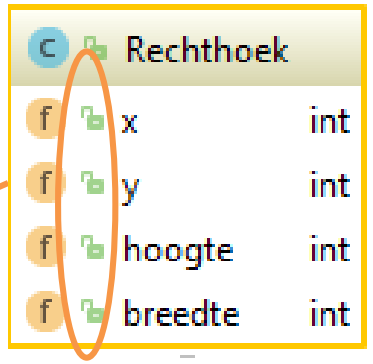
(RechthoekApp gaan we ook steeds verder uitbreiden: met code die gebruik maakt van de klasse Rechthoek)

3. De klasse-omschrijving (body)

- Body: tussen accolades
- Bevat:
  - Eigenschappen (instantievariabelen, velden, …)
  - Methoden
  - Constructors

3.1 Eigenschappen

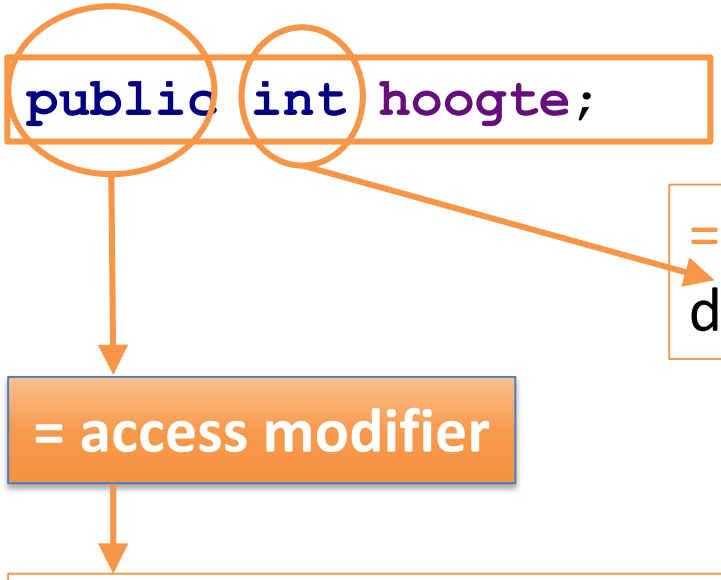
```
public class Rechthoek
{
    public int x;
    public int y;
    public int hoogte;
    public int breedte;
}
```



= public

Instantievariabelen

= UML-schema van de klasse Rechthoek



Indien de variabelen niet expliciet geïnitialiseerd worden, krijgen ze automatisch de waarde `0`, `null` of `false` naargelang het datatype.

Eigenschappen (instantievariabelen) gebruiken:

objectNaam.eigenschapNaam

```
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek = new Rechthoek();
        System.out.println(rechthoek.hoogte);
}
```

Opdracht 3: *instantievariabelen toevoegen*

- Voeg de variabelen *hoogte*, *breedte*, *x* en *y* toe aan de klasse-definitie.
- Druk de hoogte, breedte en de positie van de rechthoek af op het scherm.
- Ken in het hoofdprogramma expliciet een waarde toe aan de variabelen.
- Druk de hoogte, breedte en de positie van de rechthoek af op het scherm.
- Maak een 2e rechthoek en geef deze andere afmetingen en een andere positie. Druk de gegevens van deze 2e rechthoek af.
- Verander het toegangsniveau van de variabelen eens in private. Werkt dit?

3.2 Methoden

- Naam voor stuk code bvb. getOppervlakte()
- Heeft 1 taak bvb. oppervlakte berekenen
- Kan andere methoden gebruiken
- lowerCamelCasing

Voorbeeld:

Rechthoek

- *rechthoek1* (object) is een *Rechthoek* (class)
- *hoogte* en *breedte* zijn eigenschappen van *rechthoek1*
- *getOppervlakte()* is een methode (taak) om de oppervlakte van *rechthoek1* te berekenen

3.2.1 Declaratie van methoden

```
[public | private] returntype methodeNaam (parameters)
{
    …
}
```

- Vet gedrukt = verplicht
- Een methode kan geen waarde of 1 waarde teruggeven
  - geen waarde => terugkeertype = void
  - 1 waarde => terugkeertype is het datatype van hetgeen teruggeven wordt
- Voor elke parameter moet worden opgegeven van welk datatype de parameter is.

Voorbeeld: methode zonder terugkeerwaarde zonder parameters

```
public void vbMethode() {
…
}
```

Er wordt geen waarde teruggegeven

De methode is toegankelijk vanuit andere klassen

⟺ *private*: enkel toegankelijk vanuit de klasse zelf

Voorbeeld: methode met terugkeerwaarde met parameters

```
public int vbMethode(int par1, double par2){
  …
}
```

Er wordt een int teruggegeven

Bij het oproepen moeten er 2 argumenten meegegeven worden van het juiste type.

3.2.2 De body van de methode

```
public void vbMethode() {
    int nummer = 6;
    System.out.println(nummer);
}
```

Binnen codeblok kunnen variabelen gedeclareerd worden

lokale variabelen

moeten geïnitialiseerd worden
enkel gekend binnen de methode vbMethode

Instantievariabelen (moeten niet geïnitialiseerd worden)
Instantievariabelen zijn in elke methode van de klasse gekend

Een lokale variabele mag dezelfde naam hebben als een instantievariabele ➔ instantievariabele wordt als het ware verborgen door de lokale variabele **= shadowing**

```
public class MijnKlasse {
    private int nummer = 5;

    public void vbMethode() {
        int nummer = 6;
        System.out.println(nummer);          ➔ 6
        System.out.println(this.nummer);     ➔ 5
    }
}
```

Om de instantievariabele toch te benaderen
➔ gebruik maken van het ***this***-object
(= referentie naar het huidig object)

Een parameter mag ook dezelfde naam hebben als een instantievariabele ➔ de instantievariabele wordt als het ware verborgen door de parameter

this gebruiken als je de instantie variabele wil benaderen

```
public class MijnKlasse {
    private int nummer;

    public void setNummer(int nummer) {
        this.nummer = nummer;
    }
}
```

3.2.3 Gegevens doorgeven aan een methode

```
public class Rechthoek {
    public int x;
    public int y;
   …
    public void setPositie(int xpos, int ypos) {
        x = xpos;
        y = ypos;
    }
   …
}
```

= parameters

Bereik van *xpos* en *ypos*

waarde 7 wordt gekopieerd naar xpos

waarde 9 wordt gekopieerd naar ypos

```
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek = new Rechthoek();
        rechthoek.setPositie(7, 9);
    }
}
```

= argumenten

```
public void setPositie(int xpos, int ypos)
{
    x = xpos;
    y = ypos;
}
```

= call-by-value

```
rechthoek.setPositie(7, 9);
```

Voor primitieve datatypes wordt de waarde doorgegeven.

```java
public class Rechthoek {
    public int x;
    public int y;
    …
    public void setPositie(int xpos, int ypos) {
        x = xpos;
        y = ypos;
        xpos = 0;
        ypos = 0;
    }
    …
}
```

```java
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek = new Rechthoek();
        int a = 5;
        int b = 7;
        rechthoek.setPositie(a, b);
        System.out.println(a);
        System.out.println(b);
    }
}
```

➔ 5

➔ 7

➔ Het wijzigen van de waarde van *xpos* in de methode heeft geen invloed op de waarde van *a*. De waarde van de variabele *a* wordt immers doorgegeven (gekopieerd) naar de variabele *xpos*.

```
public class Rechthoek {
    public int x;
    public int y;
    …
    public void setPositie(int [] pos) {
        x = pos[0];
        y = pos[1];
    }
    …
}
```

Bereik van *pos*

de referentie van positie wordt gekopieerd naar pos

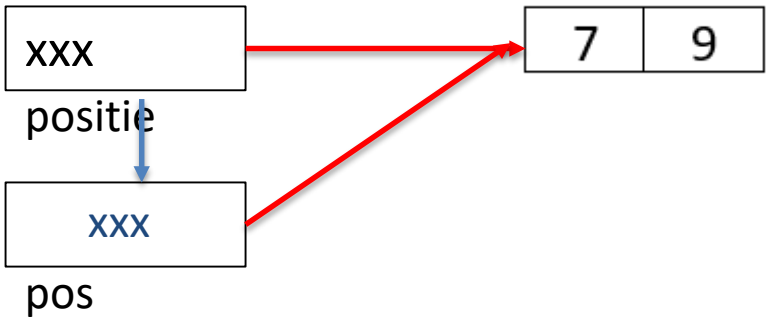
```
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek = new Rechthoek();
        int [] positie = {7, 9};
        rechthoek.setPositie(positie);
    }
}
```

```
public void setPositie(int [] pos) {
    x = pos[0];
    y = pos[1];
}
```

= call-by-reference

Voor referentietypes wordt de referentie doorgegeven.

```
rechthoek.setPositie(positie);
```



```java
public class Rechthoek {
    public int x;
    public int y;
    …
    public void setPositie(int [] pos) {
        x = pos[0];
        y = pos[1];
        pos[0] = 0;
        pos[1] = 0;
    }
    …
}
```

```java
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek = new Rechthoek();
        int [] a = {5, 7};
        rechthoek.setPositie(a);
        System.out.println(a[0]); ➔ ?
        System.out.println(a[1]); ➔ ?
    }
}
```

```
public void setPositie(int [] pos) {
    x = pos[0];
    y = pos[1];
    pos[0] = 0;
    pos[1] = 0;
}
```

```
rect.setPositie(positie);
```

= call-by-reference

Voor referentietypes wordt de referentie doorgegeven.

```
public class Rechthoek {
    public int x;
    public int y;
   …
    public void setPositie(int [] pos) {
        x = pos[0];
        y = pos[1];
        pos[0] = 0;
        pos[1] = 0;
    }
   …
}
```

```
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek = new Rechthoek();
        int [] a = {5, 7};
        rechthoek.setPositie(a);
        System.out.println(a[0]);    → 0
        System.out.println(a[1]);    → 0
    }
}
```

Opdracht 4: *Methoden met parameters toevoegen*

- Voeg aan de klasse `Rechthoek` de nodige methoden toe om de hoogte en de breedte van de rechthoek in te stellen.
- Voeg methoden toe om de positie van de rechthoek in te stellen:

```
setPositie()
setX()
setY()
```

- Voeg een methode toe om de grootte van de rechthoek met een bepaalde waarde te laten toenemen:

```
groei(int dw, int dh)
```

3.2.4 Waarden teruggeven via een methode (return)

```
return expressie;
```

```
public class Rechthoek {    …
    public double getOppervlakte() {
        return breedte * hoogte;
    }
}
```

returntype

- *return*: meestal op het einde van een methode.
- Indien het *return*-statement in het midden van een methode staat, wordt de methode op de plaats van het *return*-statement beëindigd.

```
public static void main(String[] args) {
    …
     double oppervlakte = rechthoek.getOppervlakte();
     System.out.println("Oppervlakte: " + oppervlakte);

}
```

Opdracht 5: *Methoden toevoegen die waarden teruggeven*

- Voeg aan de klasse Rechthoek de nodige methoden toe om de oppervlakte en de omtrek te berekenen:

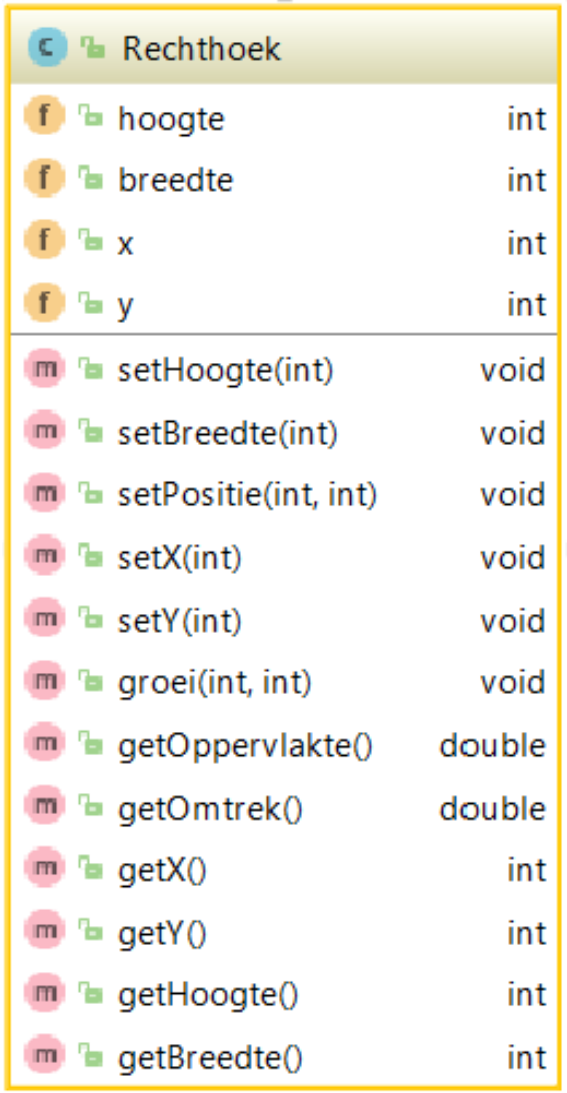
```
getOppervlakte()
getOmtrek()
```

- Druk in het hoofdprogramma de oppervlakte en omtrek van de rechthoeken op het scherm af.

- Voeg methoden toe om de hoogte de breedte en de positie van de rechthoek op te vragen:

```
getBreedte()
getHoogte()
getX()
getY()
```

- Gebruik deze methoden om de hoogte, de breedte en de positie in het hoofdprogramma af te drukken.



private of **public** instantievariabelen?

```
public class Rechthoek {
    public int hoogte;
    public int breedte;

    public void setHoogte(int hoogte){
        this.hoogte = hoogte;
    }
}
```

```
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek = new Rechthoek();
        rechthoek.hoogte = 5;
    }
}
```

De hoogte van de rechthoek kan eender welke integer waarde krijgen. Wat als je de mogelijke waarden wil beperken?

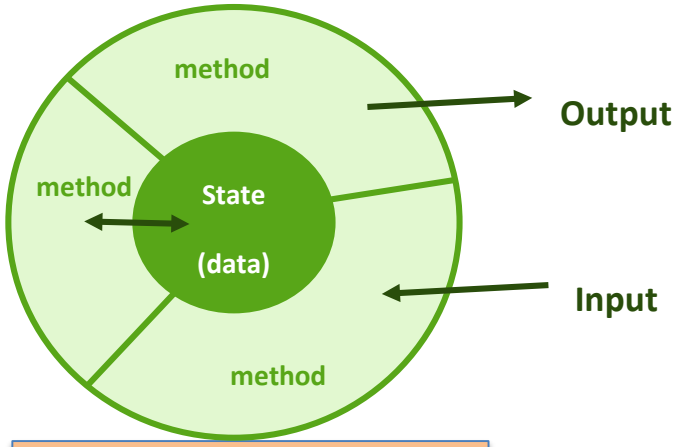
```
private int hoogte;
private int breedte;

public int getHoogte() {
    return hoogte;
}

public void setHoogte(int hoogte){
    if (hoogte < 0){
        this.hoogte = -hoogte;
    } else {
        this.hoogte = hoogte;
    }
}
```

= getter

= setter

We kunnen bijvoorbeeld beslissen om negatieve getallen te vervangen door hun absolute waarde (of te vervangen door 0)

- Het is een slechte gewoonte om instantievariabelen **`public`** te maken
  → **`private`** (= afschermen van de buitenwereld)



= encapsulation (inkapseling; data hiding)

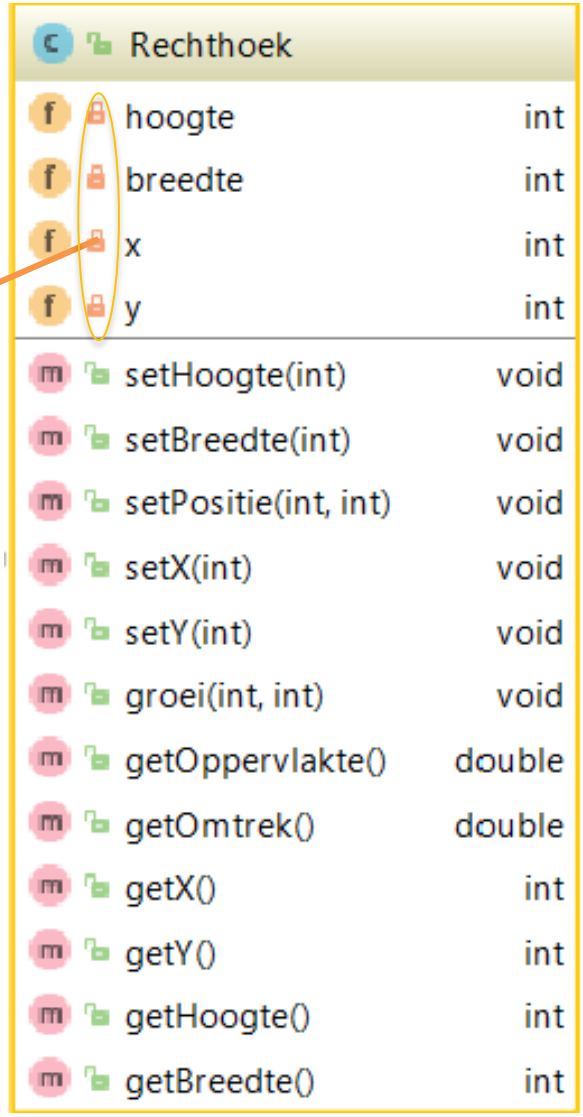
- Je maakt deze variabelen toegankelijk via aangepaste methoden

- Via de **`public`** methode kan je dan controle uitoefenen op de waarde van de **`private`** instantievariabelen.

Opdracht 6: *Controle op variabelen via methoden*

- Pas de methoden voor het zetten van de hoogte en de breedte aan zodat de absolute waarde genomen wordt. Zorg ook dat de methode *`groei()`* de waarden niet negatief maakt.
- Maak de eigenschappen *`hoogte`*, *`breedte`*, *`x`* en *`y`* van de klasse *`Rechthoek`* private en gebruik enkel de overeenkomstige methoden om de eigenschappen in te stellen en op te vragen.
- Maak in het hoofdprogramma een rechthoek met een negatieve hoogte en breedte en controleer het resultaat.

= private


Rechthoek
hoogte int
breedte int
x int
y int
setHoogte(int) void
setBreedte(int) void
setPositie(int, int) void
setX(int) void
setY(int) void
groei(int, int) void
getOppervlakte() double
getOmtrek() double
getX() int
getY() int
getHoogte() int
getBreedte() int

3.2.5 Method overloading

= methoden met dezelfde naam maar met verschillende (types/aantal) parameters

```
private int hoogte;
private int breedte;

public void setHoogte(int hoogte){
    if (hoogte < 0){
        this.hoogte = -hoogte;
    } else {
        this.hoogte = hoogte;
    }
}
public void setHoogte(double hoogte){
    if (hoogte < 0){
        this.hoogte = -(int)hoogte;
    } else {
        this.hoogte = (int)hoogte;
    }
}
```

De compiler kiest de juiste methode op basis van de argumenten die worden doorgegeven.

De inhoud van de body mag voor beide methodes verschillend zijn.

```
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek = new Rechthoek();
        int intHoogte = 10;
        double doubleHoogte = 20.5;
        rechthoek.setHoogte(intHoogte);
        rechthoek.setHoogte(doubleHoogte);
    }
}
```

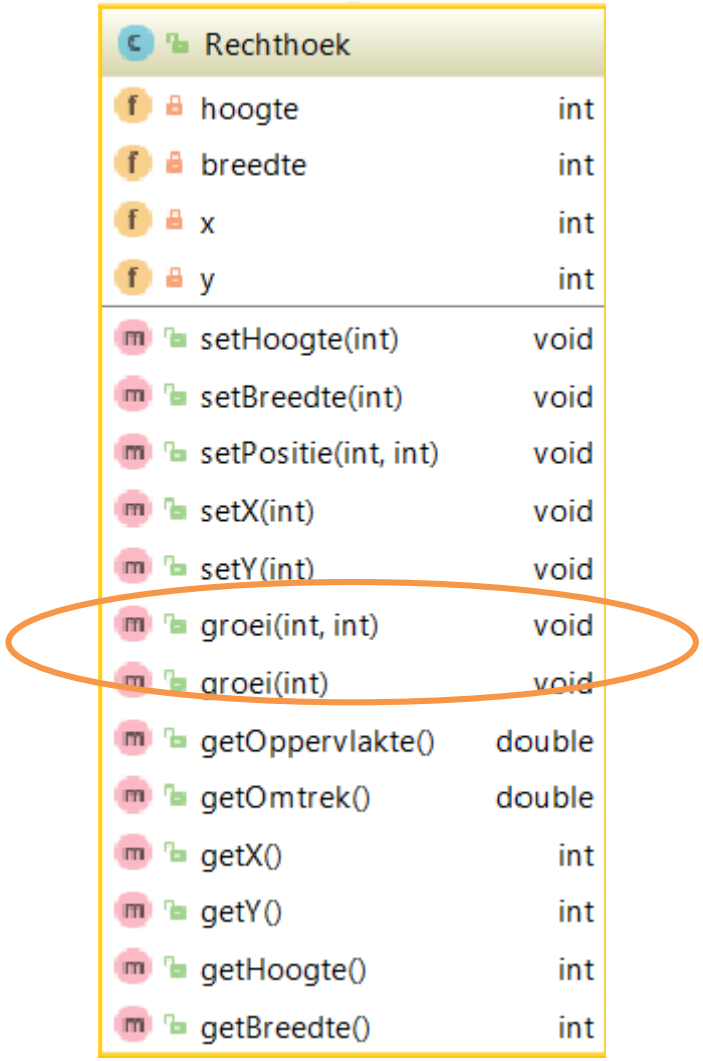
```
public void setHoogte(int hoogte){…}
```

```
public void setHoogte(double hoogte){…}
```

= Method-overloading

Opdracht 7: *Method overloading*

- Voeg een tweede methode *`groei()`* toe die slechts één parameter heeft die zowel voor de hoogte als de breedte geldt.
- Test de methode uit in het hoofdprogramma.


Rechthoek
hoogte int
breedte int
x int
y int
setHoogte(int) void
setBreedte(int) void
setPositie(int, int) void
setX(int) void
setY(int) void
groei(int, int) void
groei(int) void
getOppervlakte() double
getOmtrek() double
getX() int
getY() int
getHoogte() int
getBreedte() int

3.3 Constructors

- Iedere klasse heeft minstens één constructor
- Bij het aanmaken van een object wordt de constructor uitgevoerd
- Wordt opgeroepen door gebruik te maken van de new operator
- Wordt gebruikt om het object te initialiseren.
- Lijkt op een gewone methode maar heeft geen return-type en de naam komt overeen met de klassenaam.

```
public class Rechthoek {
    private int breedte;
    private int hoogte;
   …

    public Rechthoek() {
    }

    public Rechthoek(int breedte, int hoogte) {
        this.breedte = breedte;
        this.hoogte = hoogte;
    }
```

= Constructor-overloading

= default constructor
(= parameterloze constructor)

= constructor met parameters

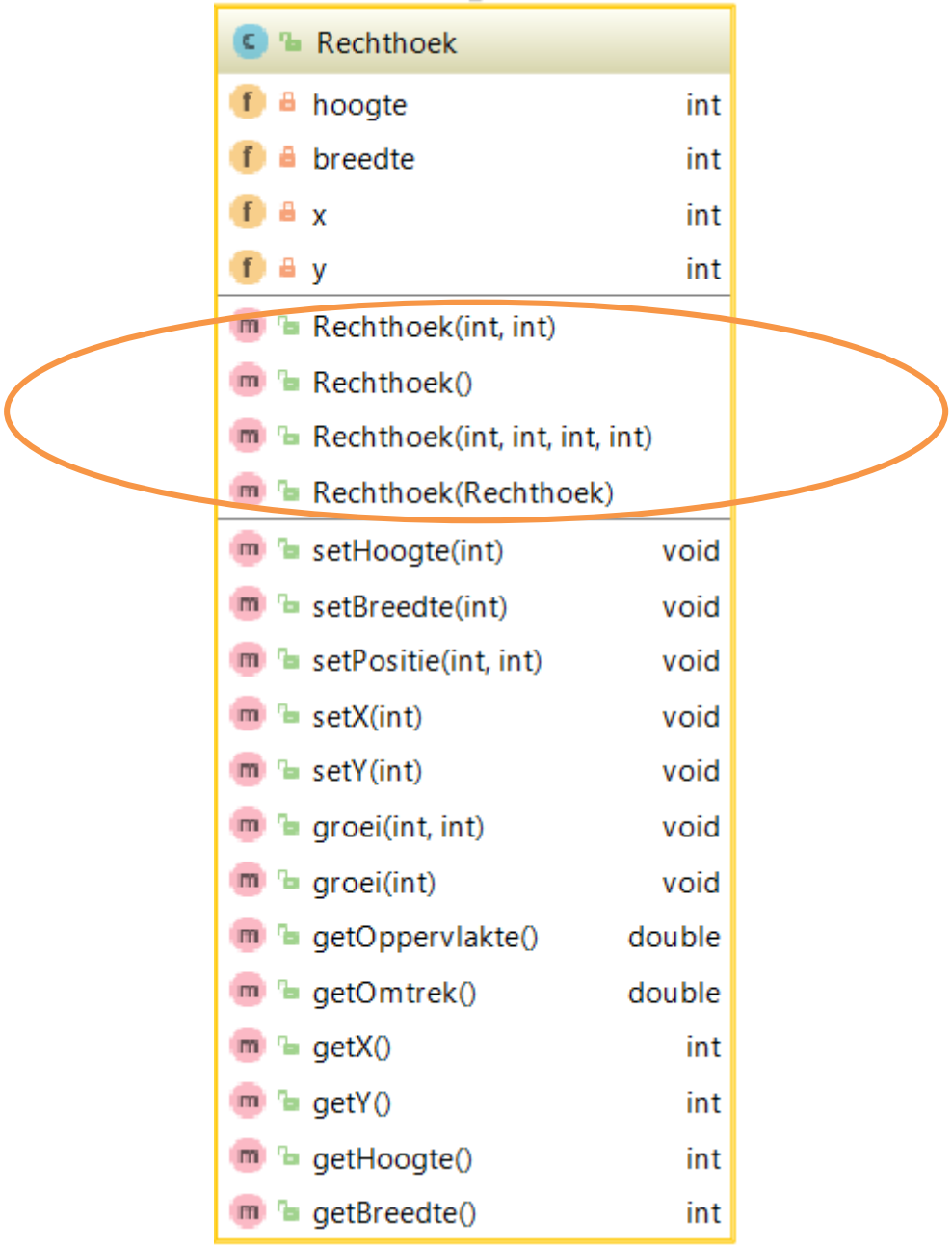
```
public class RechthoekApp {
    public static void main(String[] args) {
        Rechthoek rechthoek1 = new Rechthoek();
        Rechthoek rechthoek2 = new Rechthoek(4, 7);
```

Default constructor

- Indien een constructor niet expliciet gedefinieerd wordt, krijgt de klasse een default constructor zonder parameters *(default constructor moet dus niet geschreven worden)*
- Zodra er een constructor met parameters gespecifieerd wordt, vervalt deze default constructor en moet dan eventueel expliciet gedeclareerd worden *(indien deze aangeroepen wordt vanuit het programma)*

Opdracht 8: *Constructors toevoegen*

- Voeg een constructor toe met 2 parameters voor de initiële hoogte en breedte. Maak een rechthoek met deze constructor en druk de hoogte en breedte van de rechthoek af op het scherm.
- Voeg een constructor toe zonder parameters.
- Voeg een 3e constructor toe die naast de hoogte en de breedte ook nog de x-positie en de y-positie van de rechthoek als parameter neemt. Maak in het hoofdprogramma een rechthoek met deze constructor en druk de gegevens af op het scherm.
- Voeg een constructor toe die een rechthoek maakt o.b.v. een bestaande rechthoek. Geef als parameter een referentie naar de bestaande rechthoek mee.


Rechthoek
hoogte int
breedte int
x int
y int
Rechthoek(int, int)
Rechthoek()
Rechthoek(int, int, int, int)
Rechthoek(Rechthoek)
setHoogte(int) void
setBreedte(int) void
setPositie(int, int) void
setX(int) void
setY(int) void
groei(int, int) void
groei(int) void
getOppervlakte() double
getOmtrek() double
getX() int
getY() int
getHoogte() int
getBreedte() int

Een constructor kan een andere constructor met de nodige parameters aanroepen: met de referentievariabele *this*

```
public class Rechthoek {
    …
     public Rechthoek() {
          this(0, 0);
     }

     public Rechthoek(int breedte, int hoogte) {
          x = 0;
          y = 0;
          this.hoogte = hoogte;
          this.breedte = breedte;
     }
}
```

this verwijst naar het eigen object (= referentievariabele)

Voordeel:

- De eigenlijke code voor het initialiseren van het object moet maar op 1 plaats geschreven worden.
- Code wordt compacter en overzichtelijker
- Risico op fouten daalt: aanpassingen moeten immers gebeuren in 1 constructor i.p.v. in elke constructor.

Indien een constructor een andere constructor aanroept moet dit steeds gebeuren op de eerste regel

Opdracht 9: *Het gebruik van this*

Laat de constructors met minder parameters de constructor met de meeste parameters aanroepen. Zorg er ook voor dat een rechthoek geen negatieve afmetingen krijgt via de constructor.

3.4 Instantievariabelen en klassevariabelen

3.4.1 Instantievariabelen

Instantievariabelen

= ieder object van een bepaalde klasse heeft een eigen kopie van de variabelen met elk hun eigen waarde.

Synoniemen: veld, eigenschap, objectvariabele, kenmerk, attribuut

Indien de variabelen niet expliciet geïnitialiseerd worden, krijgen ze automatisch de waarde `0`, `null` of `false` naargelang het datatype.

De initialisatie van de instantievariabelen kan als volgt:

1. In de constructor (waarden komen binnen als parameters).

2. Tijdens de declaratie (vooral voor primitieve datatypen).

manier 1: Initialisatie instantievariabelen via constructor

```
public class Rechthoek {
    private int breedte;
    private int hoogte;
    private int x;
    private int y;

    public Rechthoek (int breedte, int hoogte){
        this.breedte = breedte;
        this.hoogte = hoogte;
    }
}
```

manier 2: Initialisatie instantievariabelen tijdens de declaratie

```
public class Rechthoek {
    private int breedte = 5;
    private int hoogte = 10;
    private int x = 1;
    private int y;

}
```

3.4.2 Klassevariabelen

Klassevariabelen

- Zijn gemeenschappelijk voor alle objecten van dezelfde klasse.
- Van elke klassevariabele is er slechts 1, die door alle objecten gedeeld wordt.
- Worden gedefinieerd met `static`.
- De initialisatie gebeurt bij de declaratie.

Voorbeeld: klasse Rechthoek

```
public class Rechthoek {
    private int hoogte =3   ;
    private int breedte;
    private int x;
    private int y;
    public static final int HOEKEN =4;

    …
}
```

- Het aantal hoeken is voor elke rechthoek gelijk, → 1 variabele is voldoende.
- static → klassevariabele
- final → constante
- public → mag publiekelijk geraadpleegd worden (kan omwille van 'final' toch niet aangepast worden)

Voorbeeld: klasse Rechthoek

Bijhouden hoeveel rechthoeken gecreëerd werden

```
public class Rechthoek {
    private int hoogte =3  ;
    private int breedte;
    private int x;
    private int y;
    public static final int HOEKEN =4;
    public static int tel = 0;
    …
}
```

- Telkens een object gecreëerd wordt, moet het geteld worden in de constructor
- tel is voorlopig public

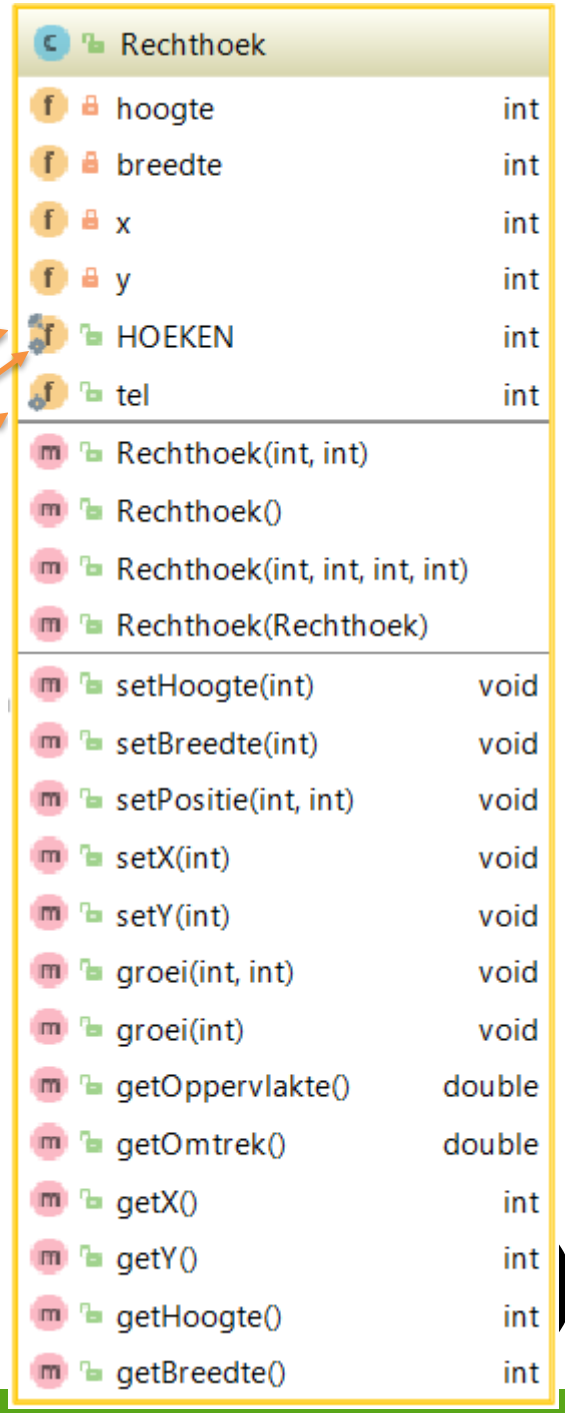
```
public class Rechthoek {
    private int breedte;
    private int hoogte;
    private int x;
    private int y;
    public static final int HOEKEN = 4;
    public static int tel = 0;

    public Rechthoek() {
        this(0, 0);
    }

    public Rechthoek(int breedte, int hoogte) {
        this.hoogte = hoogte;
        this.breedte = breedte;
        tel++;
    }
    …
}
```

Doordat de constructor met minder parameters de constructor met de meeste parameters oproepen moet de code maar 1 keer toegevoegd worden!

Voorbeeld:
klasse Rechthoek : UML-diagram

final

Klassevariabele



Hoe een klassevariabele gebruiken in de main?

KlasseNaam.variabele

→ Zo is het duidelijk dat de variabele bij een klasse hoort

In de main-methode:

```
System.out.println(Rechthoek.HOEKEN);
```

Opdracht 10

1. Maak een klasse ‘Klas’ met als eigenschappen de naam van de klas en het aantal studenten.
2. Maak 2 constructors: een klas die aangemaakt wordt via de default-constructor wordt ‘1TINx’ met 0 studenten. De tweede constructor krijgt 2 waarden binnen die toegekend worden aan de 2 objectvariabelen. De eerste constructor roept de tweede op.
3. Maak voor alle objectvariabelen getters en setters aan.
4. Voeg een klassevariabele toe die het maximum aantal studenten bevat dat in een klas kan. Geef hieraan de waarde 40.
5. Druk (in de main) het maximum aantal studenten af op het scherm.
6. Zorg dat bij creatie of wijziging van een klas-object rekening gehouden wordt met het maximum.
7. Voeg een klasse-variabele toe die het aantal objecten van die klasse bevat.
8. Maak een aantal klas-objecten aan en druk telkens het aantal klassen af op het scherm.
9. Maak een array en voeg hierin al je klas-objecten toe. Overloop alle objecten en druk van alle klassen naam en aantal studenten af.

3.5 Instantiemethoden en klassemethoden

3.5.1 Instantiemethoden

Instantiemethode

→ moet opgeroepen worden op een concreet object van de klasse

Voorbeeld: `rechthoek.setPositie(3, 6);`

→ De methode kan gebruik maken van de instantievariabelen van dat specifieke object

→ De methode kan ook gebruik maken van de klassevariabelen

→ De methode kan gebruik maken van lokale variabelen

```
public class Rechthoek {
    private int hoogte;
    private int breedte;
    private int x;
    private int y;
    public static final int HOEKEN = 4;
    public static int tel = 0;

    public Rechthoek(int x, int y, int h, int b) {
        this.x = x;
        this.y = y;
        setBreedte(b);
        setHoogte(h);
        tel++;

    }

    public setPosition(){
        int hulp = (x + y) / 2
        x = tel * hulp + breedte / 2;
        y = tel * HOEKEN + hoogte / 3;

    } ...
```

In een instantiemethode zijn toegelaten:

- lokale variabele
- klassevariabele
- instantievariabele

3.5.2 Klassemethoden

Klassemethode

→moet opgeroepen worden op basis van de klasse-naam

```
KlasseNaam.methode()
```

→De methode kan gebruik maken van de klassevariabelen

→De methode kan gebruik maken van lokale variabelen binnen de methode

Voorbeeld: klasse Rechthoek

Klassemethode die het aantal objecten teruggeeft

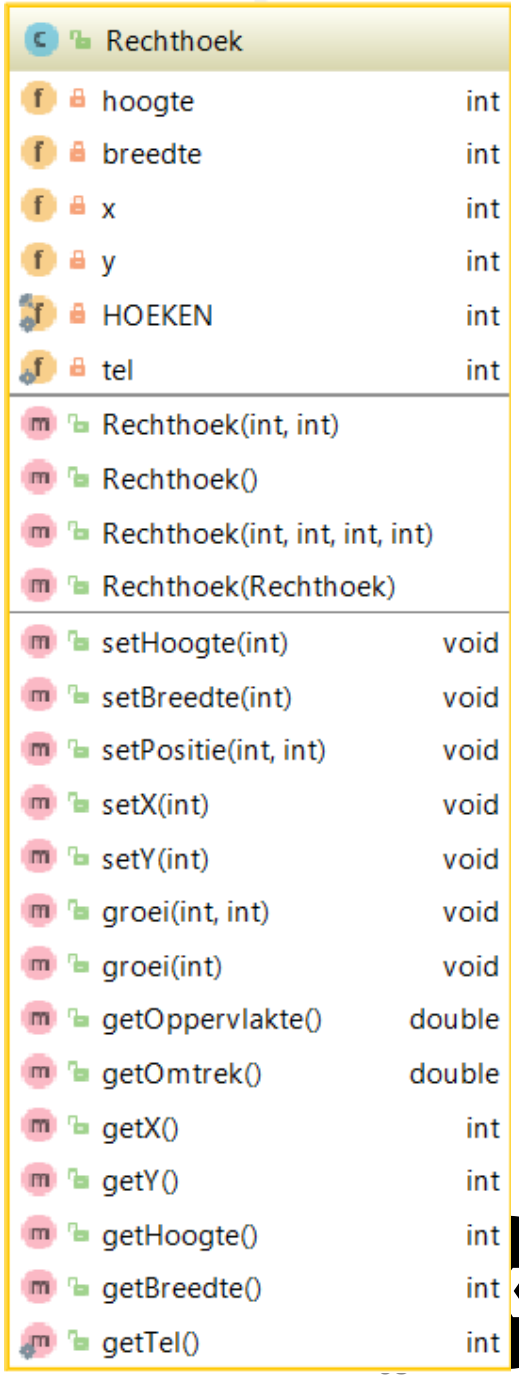
```
public class Rechthoek {
    private int breedte;
    private int hoogte;
    private int x;
    private int y;
    public static final int HOEKEN = 4;
    private static int tel = 0;

    public static int getTel() {
        return tel;
    }
   …
}
```

- tel is nu private!
- Methode getTel() is
  - public
  - static

Voorbeeld: klasse Rechthoek

Klassemethode die het aantal objecten teruggeeft

static methode getTel()



```
public class Rechthoek {
    private int hoogte;
    private int breedte;
    private int x;
    private int y;
    public static final int HOEKEN = 4;
    public static int tel = 0;

    public Rechthoek(int x, int y, int h, int b) {
        this.x = x;
        this.y = y;
        setBreedte(b);
        setHoogte(h);
        tel++;
    }

    public static int getTelSpeciaal(){
        int hulp = HOEKEN * tel * tel;
        return hulp;
    }
}
```

^{2}...

In een klassemethode zijn toegelaten:

- lokale variabele
- klassevariabele

Hoe een klassemethode gebruiken in de main?

```
KlasseNaam.methode()
```

Opvragen van het aantal gecreëerde Rechthoek-objecten

In de main-methode:

```
int teller = Rechthoek.getTel();
```

Opmerkingen:

1. De main-methode is een static method.

   →We kunnen enkel klassevariabelen gebruiken.

   →We kunnen wel lokale variabelen creëren en gebruiken.

2. Binnen een klassemethode is de this-referentie onbestaande.

3. Je kan klassevariabelen en klassemethoden importeren.
   → de vermelding van de klassenaam is niet meer nodig.

Voorbeeld:

```
package be.pxl.h2.opdracht;
import static be.pxl.h2.opdracht.Rechthoek.*;

public class RechthoekApp {
    public static void main (String [] args) {
        System.out.println(getTel());
        System.out.println(HOEKEN);
    }
}
```

Opdracht 11

1. Voeg aan de klasse ‘Klas’ een methode toe die het aantal Klas-objecten geeft.
   Maak de variabele die het aantal klassen bijhoudt private.
2. Maak een variabele om het totaal aantal studenten bij te houden. Klassevariabele of instantievariabele?
   Zorg ervoor dat dit totaal correct bijgehouden wordt.
   Maak eveneens een get-methode aan om dit totaal te kunnen opvragen.
3. In de main-methode druk je het aantal aangemaakte klassen af en het totaal aantal studenten.
4. Voeg 1 student toe aan een bepaalde klas en controleer of het totaal nog steeds klopt.
5. Bereken het gemiddeld aantal studenten per klas.
   Druk dit gemiddelde af met 1 cijfer na de komma.