Java Essentials

Hoofdstuk 2

Constructors, klassevariabelen en klassemethoden

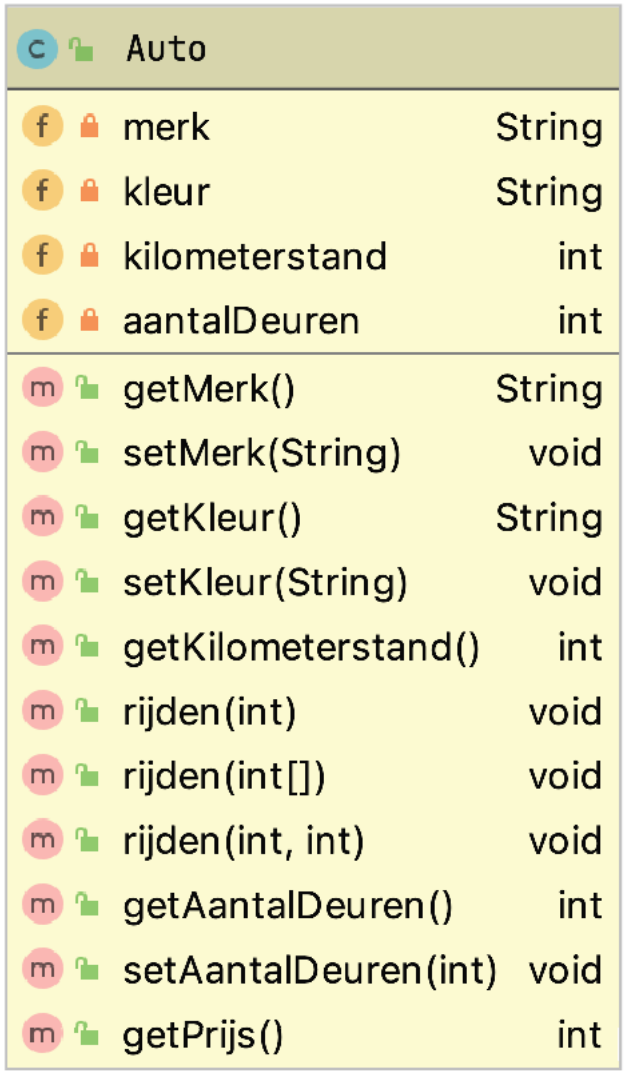
DE HOGESCHOOL MET HET NETWERK

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

Inhoud

1. Inleiding
2. Constructors
3. Instantievariabelen en klassevariabelen
4. Instantiemethoden en klassemethoden

1. Inleiding


Auto
merk String
kleur String
kilometerstand int
aantalDeuren int
getMerk() String
setMerk(String) void
getKleur() String
setKleur(String) void
getKilometerstand() int
rijden(int) void
rijden(int[]) void
rijden(int, int) void
getAantalDeuren() int
setAantalDeuren(int) void
getPrijs() int
eigenschappen
getters, setters en andere methoden

1. Inleiding

```
public class Auto {
  private String merk;
  private String kleur;
  private int kilometerstand;
  private int aantalDeuren;

  public String getKleur() {
    return kleur;
  }

  public void setKleur(String kleur) {
    this.kleur = kleur;
  }

  /** overige getters en setters worden niet getoond */
}
```

1. Inleiding

```
public class Auto {
  private String merk;
  private String kleur;
  private int kilometerstand;
  private int aantalDeuren;

  /** getters en setters worden niet getoond */

  public void rijden(int afstand) {
    kilometerstand += afstand;
  }

  public int getPrijs() {
    int prijs = 20000;
    if (aantalDeuren == 3) {
      prijs = prijs - 1000;
    }
    return prijs;
  }
}
```

```
public class AutoApp {

  public static void main(String[] args) {
    Auto auto = new Auto();
    auto.setMerk("Tesla");
    auto.setKleur("rood");
    auto.setAantalDeuren(5);

    auto.rijden(1250);
    auto.rijden(320);
    auto.rijden(2, 60);

    System.out.println(auto.getKilometerstand());
  }
}
```

Object aanmaken en initialiseren

2. Constructors

```
public class AutoApp {

  public static void main(String[] args) {
    Auto auto = new Auto("Tesla", "rood", 5);

    auto.rijden(1250);
    auto.rijden(320);
    auto.rijden(2, 60);

    System.out.println(auto.getKilometerstand());
  }
}
```

2. Constructors

```
public class Auto {
  private String merk;
  private String kleur;
  private int kilometerstand;
  private int aantalDeuren;

  public Auto(String merk, String kleur, int aantalDeuren) {
    this.merk = merk;
    this.kleur = kleur;
    this.aantalDeuren = aantalDeuren;
  }
```

Constructor

2. 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.

= Constructor-overloading

```
public class Auto {
  private String merk;
  private String kleur;
  private int kilometerstand;
  private int aantalDeuren;

  public Auto() {
  }

  public Auto(String merk, String kleur, int aantalDeuren) {
    this.merk = merk;
    this.kleur = kleur;
    this.aantalDeuren = aantalDeuren;
  }
```

= default constructor
(= parameterloze constructor)

= constructor met parameters

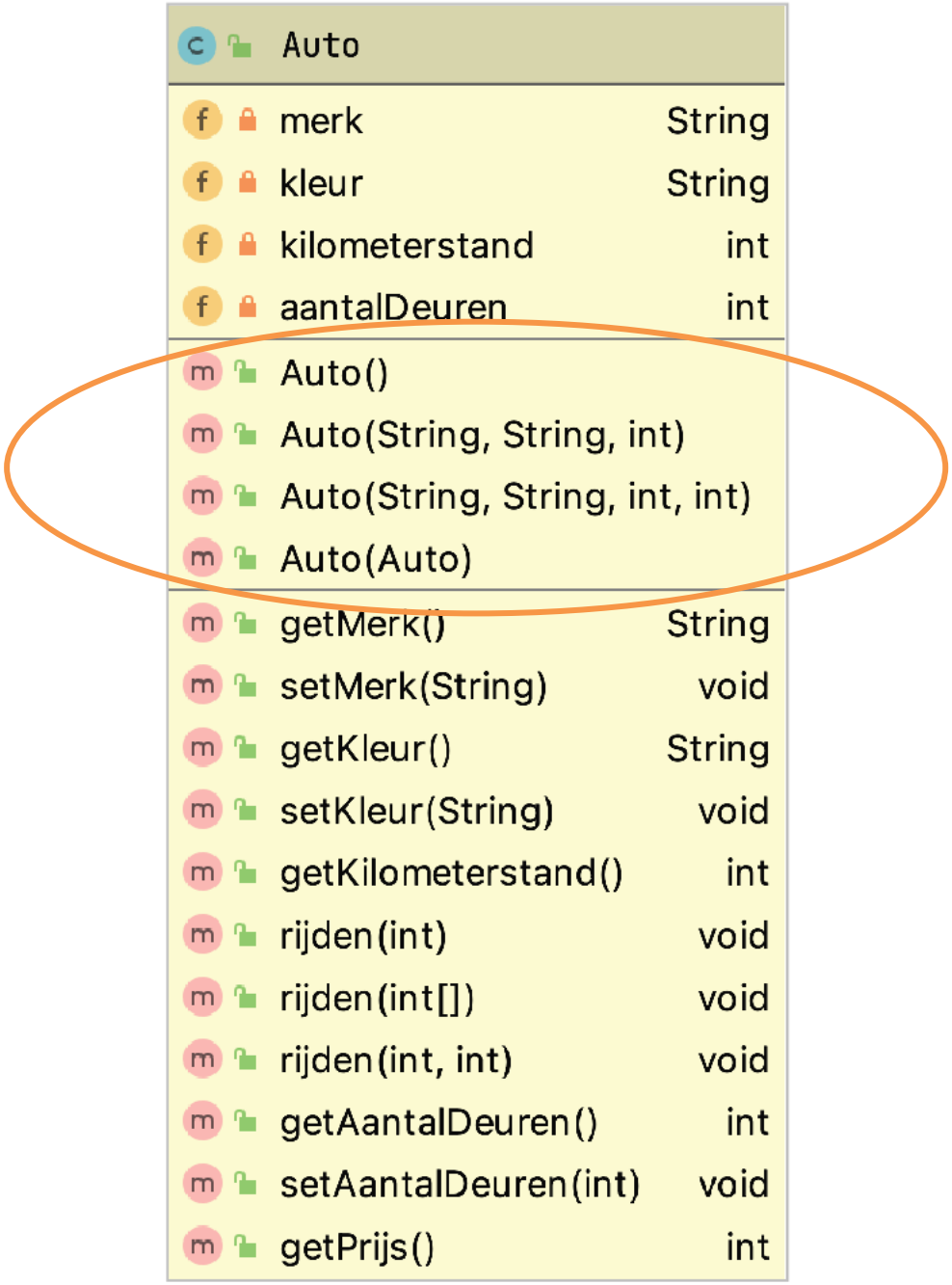
```
public class AutoApp {
   public static void main(String[] args) {
     Auto auto1 = new Auto();
     Auto auto2 = new Auto("Tesla", "rood", 5);
   }
}
```

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, verdwijnt de default constructor en moet dan eventueel expliciet gedeclareerd worden *(indien deze aangeroepen wordt vanuit het programma)*

Opdracht 1: *Constructors toevoegen*

- Voeg in de klasse Auto een constructor toe met 3 parameters: het merk, de kleur en het aantal deuren Maak een auto met deze constructor en druk de kleur af op het scherm.
- Voeg een constructor toe zonder parameters.
- Voeg een 3e constructor toe die naast merk, kleur en aantal deuren ook de kilometerstand als parameter heeft. Maak in het hoofdprogramma een auto met deze constructor en druk de gegevens af op het scherm.
- Voeg een constructor toe die een auto maakt o.b.v. een bestaande auto. Geef als parameter een referentie naar de bestaande auto mee.


Auto
merk String
kleur String
kilometerstand int
aantalDeuren int
Auto()
Auto(String, String, int)
Auto(String, String, int, int)
Auto(Auto)
getMerk() String
setMerk(String) void
getKleur() String
setKleur(String) void
getKilometerstand() int
rijden(int) void
rijden(int[]) void
rijden(int, int) void
getAantalDeuren() int
setAantalDeuren(int) void
getPrijs() int

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

```
public class Auto {

  public Auto(String merk, String kleur, int aantalDeuren) {
    this(merk, kleur, aantalDeuren, 0);
  }

  public Auto(String merk, String kleur, int aantalDeuren, int kilometerstand) {
    this.merk = merk;
    this.kleur = kleur;
    this.kilometerstand = kilometerstand;
    this.aantalDeuren = aantalDeuren;
  }
}
```

this() verwijst naar een constructor binnen de klasse

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 2: *Het gebruik van this*

Laat de constructors met 3 parameters de constructor met de 4 parameters aanroepen.

Zorg er ook voor dat een auto geen negatieve kilometerstand krijgt! Wanneer een negatieve waarde wordt ingegeven, neem je de absolute waarde.

Test uit!

3. Instantievariabelen en klassevariabelen

3.1 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 default waarde van primitieve datatypen).

manier 1: Initialisatie instantievariabelen via constructor

```
public class Auto {
   private String merk;
   private String kleur;
   private int kilometerstand;
   private int aantalDeuren;

   public Auto(String merk, String kleur, int aantalDeuren, int kilometerstand) {
      this.merk = merk;
      this.kleur = kleur;
      this.kilometerstand = kilometerstand;
      this.aantalDeuren = aantalDeuren;
   }
}
```

manier 2: Initialisatie instantievariabelen tijdens de declaratie

```
public class Auto {
   private String merk = "Tesla";
   private String kleur = "rood";
   private int kilometerstand;
   private int aantalDeuren = 5;
}
```

3.2 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 Auto

```
public class Auto {
    public static final int AANTAL_WIELEN = 4;
    private String merk;
    private String kleur;
    private int kilometerstand;
    private int aantalDeuren;
```

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

Voorbeeld: klasse Auto

Bijhouden hoeveel auto-objecten gecreëerd werden

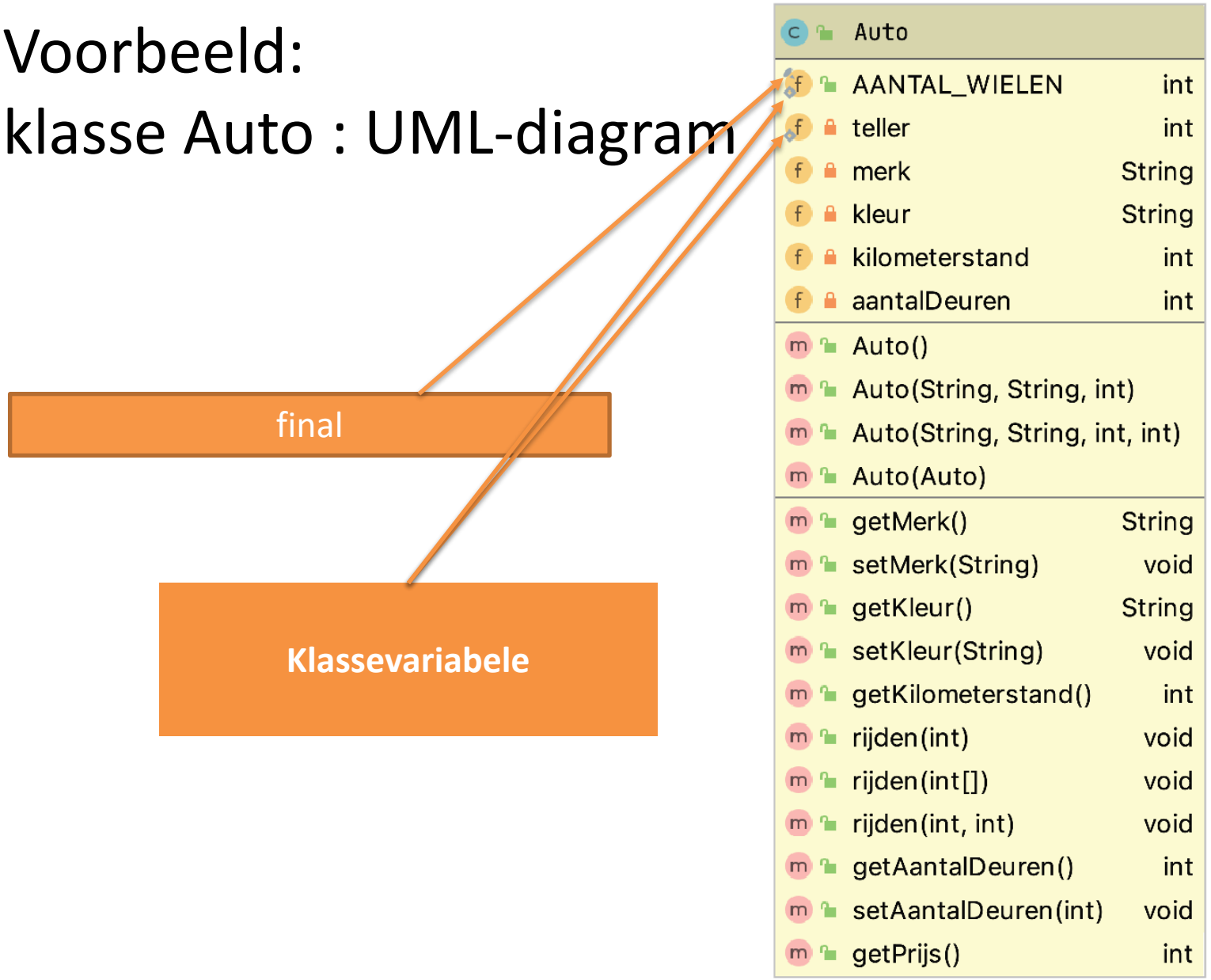
```
public class Auto {
   public static final int AANTAL_WIELEN = 4;
   private static int teller;
   private String merk;
   private String kleur;
   private int kilometerstand;
   private int aantalDeuren;
}
```

Telkens een object gecreëerd wordt,
moet het geteld worden in de constructor.
teller heeft initieel de waarde 0.

```
public class Auto {
  public static final int AANTAL_WIELEN = 4;
  private static int teller;
  private String merk;
  private String kleur;
  private int kilometerstand;
  private int aantalDeuren;

  public Auto(String merk, String kleur, int aantalDeuren) {
    this(merk, kleur, aantalDeuren, 0);
  }

  public Auto(String merk, String kleur, int aantalDeuren, int kilometerstand) {
    this.merk = merk;
    this.kleur = kleur;
    this.kilometerstand = kilometerstand;
    this.aantalDeuren = aantalDeuren;
    teller++;
  }
```

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

Voorbeeld:
klasse Auto : UML-diagram



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(Auto.AANTAL_WIELEN);
```

Opdracht 3:

1. Maak een klasse ‘Klasgroep’ met als eigenschappen de naam van de klasgroep en het aantal studenten.
2. Maak 2 constructors: een klasgroep 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 instantievariabelen getters en setters aan.
4. Voeg een klassevariabele toe die het maximum aantal studenten bevat dat in een klasgroep 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 klasgroep-object rekening gehouden wordt met het maximum.
7. Maak een array en voeg hierin al je klasgroep-objecten toe. Overloop alle objecten en druk van alle klasgroepen naam en aantal studenten af.

4. Instantiemethoden en klassemethoden

4.1 Instantiemethoden

Instantiemethode

→ moet opgeroepen worden op een concreet object van de klasse

Voorbeeld: `auto.rijden(135);`

→ 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 Auto {
   public static final int AANTAL_WIELEN = 4;
   private static int teller;
   private String merk;
   private String kleur;
   private int kilometerstand;
   private int aantalDeuren;
   private String[] banden = new String[AANTAL_WIELEN];

   public Auto(String merk, String kleur, int aantalDeuren, int kilometerstand) {
      this.merk = merk;
      this.kleur = kleur;
      this.kilometerstand = kilometerstand;
      this.aantalDeuren = aantalDeuren;
      teller++;
   }

   public void verwisselBanden(String typeBand) {
       System.out.println(AANTAL_WIELEN + " banden verwisselen.");
       for (int i = 0; i < AANTAL_WIELEN; i++) {
            banden[i] = typeBand;
       }
   }
}
```

In een instantiemethode zijn toegelaten:

- lokale variabele
- klassevariabele
- instantievariabele

4.2 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 Auto

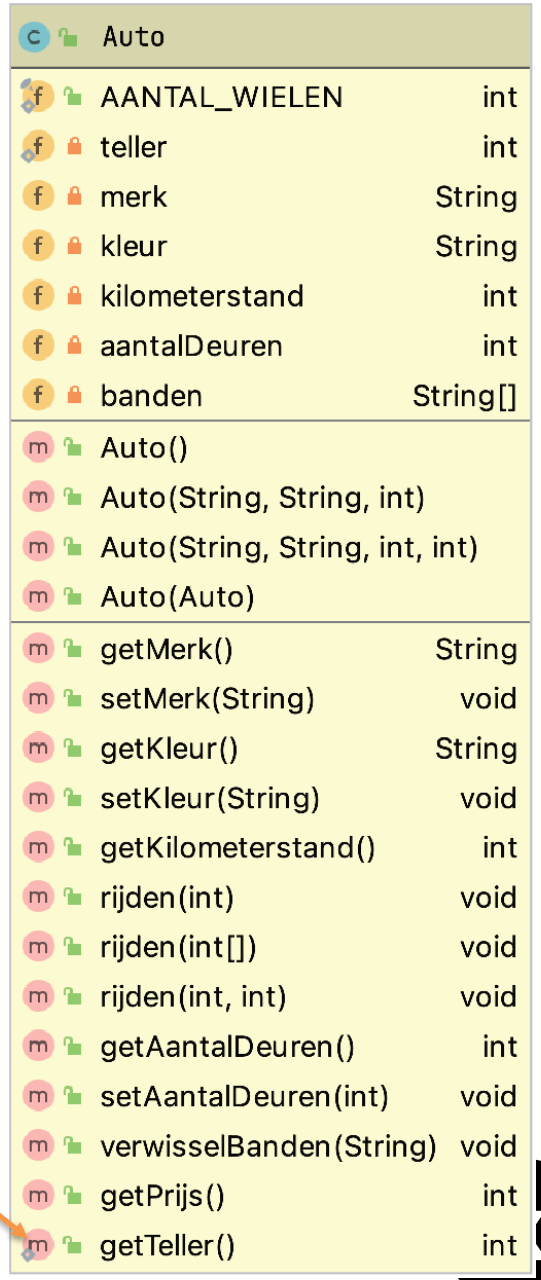
Klassemethode die het aantal objecten teruggeeft

```
public class Auto {
  public static final int AANTAL_WIELEN = 4;
  private static int teller;
  private String merk;
  private String kleur;
  private int kilometerstand;
  private int aantalDeuren;

  public Auto(String merk, String kleur, int aantalDeuren, int kilometerstand) {
    this.merk = merk;
    this.kleur = kleur;
    this.kilometerstand = kilometerstand;
    this.aantalDeuren = aantalDeuren;
    teller++;
  }

  public static int getTeller() {
    return teller;
  }
}
```

Voorbeeld: klasse Rechthoek

Klassemethode die het aantal objecten teruggeeft

static methode getTeller()



```
public static int getTeller() {
   System.out.println(this.merk);
   return teller;
}
```

‘this’ cannot be referenced from a static context.

In een klassemethode zijn toegelaten:

- lokale variabele
- klassevariabele

Hoe een klassemethode gebruiken in de main?

```
KlasseNaam.methode()
```

Opvragen van het aantal gecreëerde Auto-objecten

In de main-methode:

```
int teller = Auto.getTeller();
```

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.Auto.*;

public class AutoApp {
    public static void main (String [] args) {
        System.out.println(getTeller());
        System.out.println(AANTAL_WIELEN);
    }
}
```

Opdracht 4:

8. Voeg een private klasse-variabele toe die het aantal objecten van de klasse Klasgroep bevat. Voeg een methode toe die het aantal Klasgroep-objecten geeft.
Maak een aantal klasgroep-objecten aan en druk telkens het aantal klasgroepen af op het scherm.
9. 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.
10. In de main-methode druk je het aantal aangemaakte klasgroepen af en het totaal aantal studenten.
11. Voeg 1 student toe aan een bepaalde klas en controleer of het totaal nog steeds klopt.
12. Bereken het gemiddeld aantal studenten per klasgroep.
Druk dit gemiddelde af met 1 cijfer na de komma.