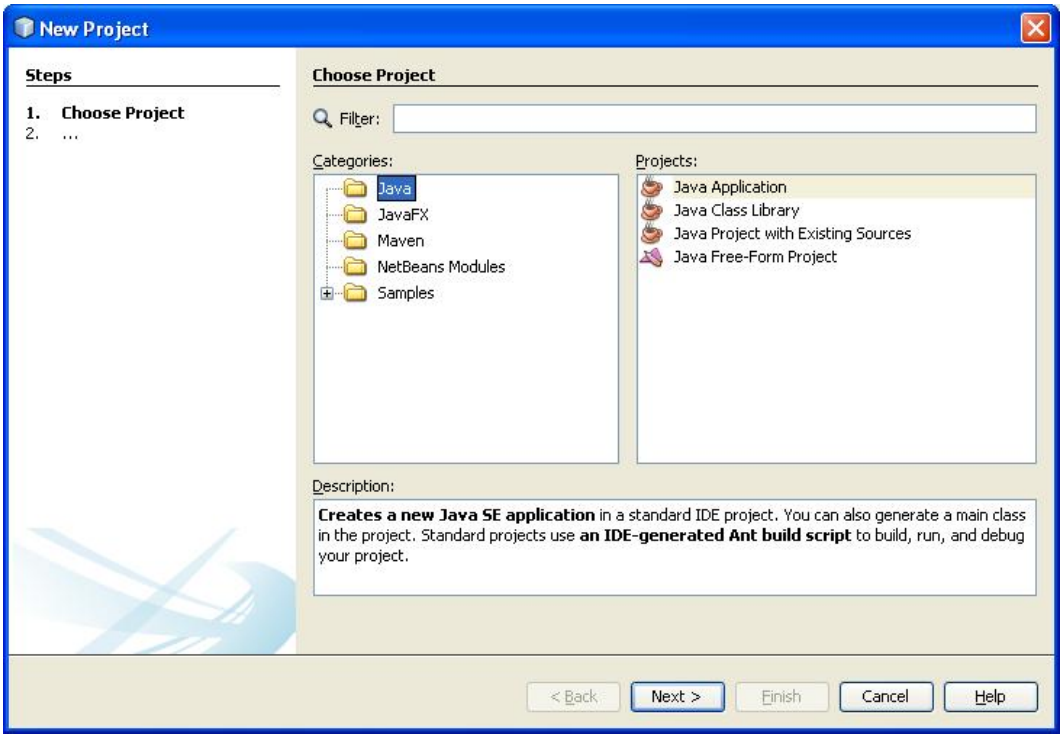
Oefenopgaven hoofdstuk 1

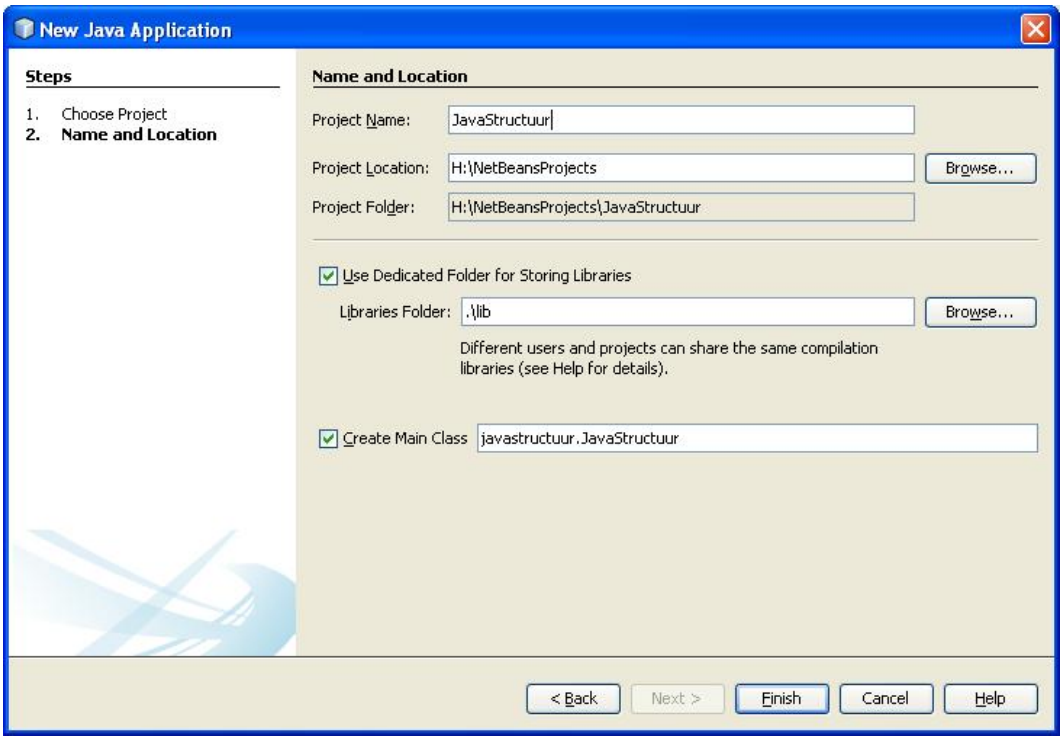
Oefenopgave 1
Wanneer u de eerste twee paragrafen van hoofdstuk 1 bestudeerd hebt, gaat u de stof toepassen door een programma te maken. U gebruikt Netbeans om het programma te ontwikkelen. Netbeans is een integrated development environment of IDE om software te ontwikkelen.
U start Netbeans op.

File → New Project
U krijgt dan het volgende scherm:

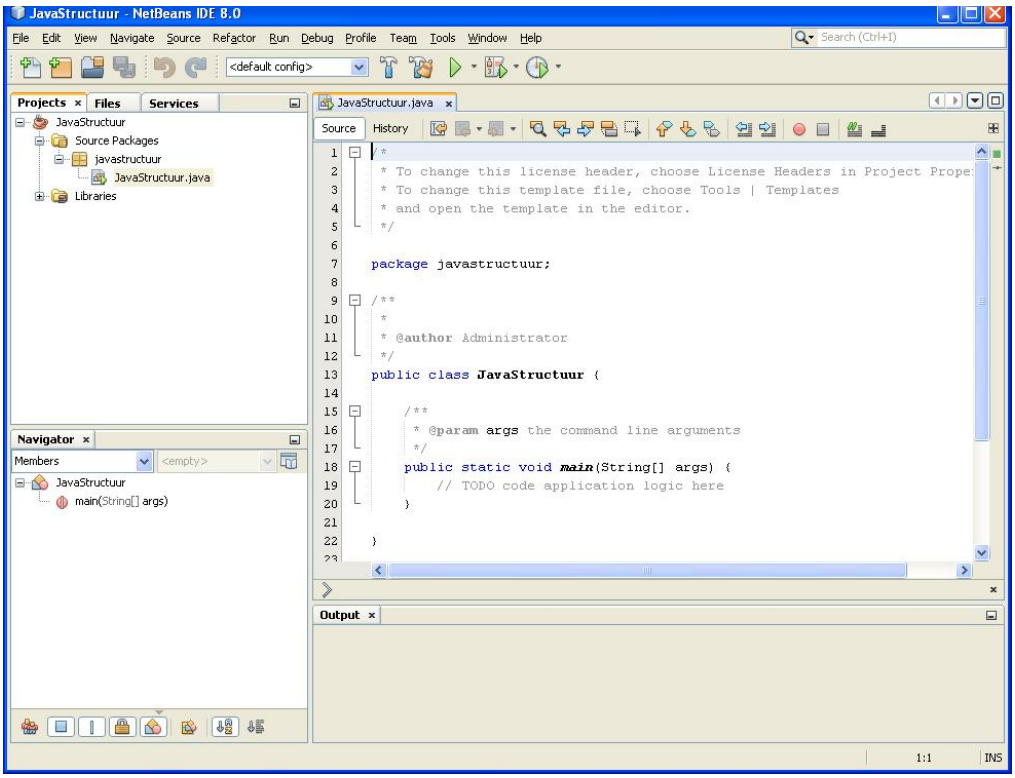


U klikt op Java Application.

U krijgt dan het volgende scherm:



U vult in:
Project Name: JavaStructuur
Eventueel een Project Location
Daarna klikt u op Finish. U krijgt dan het volgende scherm:



Links ziet u de projectstructuur met daarin de package javastructuur en de class JavaStructuur, daaronder de Navigator. In de Navigator ziet u de classNaam met zijn variabelen en methoden.

Rechts ziet u de Javacode die u kunt bewerken.
Onder dit invoerscherm verschijnt de output wanneer u het programma uitvoert.

Wanneer u nu het boek blz. 15 Java components erbij pakt, ziet u:
Comments 3a regel 1 t/m 5 enz.
Package 1 regel 7
Class declaration regel 4
Methods regel 18

Wat ontbreekt, zijn:

- Import statements 2 wordt op een later tijdstip behandeld in paragraaf 1.3.3 en 1.3.4.
- Variables 5
- Constructors 6 wordt op een later tijdstip behandeld in paragraaf 3.5.

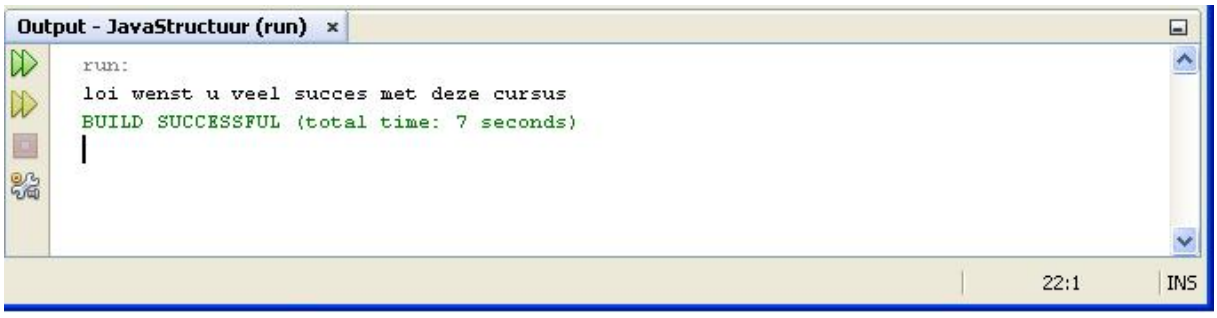
Wanneer u nu het boek paragraaf 1.2.2 Main method erbij pakt, kunt u dit in uw eigen programma nagaan.

We gaan nu bij de methode main een variabel s toevoegen. De variabele s is van het type String en we initialiseren de variabele met “loi wenst u veel succes met deze cursus”. Tevens roepen we een methode aan: **System.out.println**().
System is de package-name, in de directory out roepen we de methode **println**() aan.
Als argument kunt u bij deze methode een String gebruiken.

Voeg de twee dikgedrukte regels toe in het programma:

```
public static void main(String[] args) {
        String s = "loi wenst u veel succes met deze cursus";
        System.out.println(s);
}
```

Nu moeten we het programma uitvoeren. Dit kan op verschillende manieren.
Eén manier is: rechtermuisklik op: JavaStructuur.java (in het Projects-venster)
Dan: Run file.
U ziet nu in de Output het volgende:



Nu kunt u het programma uitbreiden met de twee regels die boven figure 1.8 in het boek staan:

```
public static void main(String[] args) {
        String s = "loi wenst u veel succes met deze cursus";
        System.out.println(s);
        System.out.println(args[0]);
        System.out.println(args[1]);
 }
```

Wanneer u het programma uitvoert, krijgt u de volgende melding:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
```

```
   at javastructuur.JavaStructuur.main(JavaStructuur.java:21)
Java Result: 1
```

Exception handling is een mechanisme in Java om op een uitzonderlijke gebeurtenis (excepties) te reageren. Dit is nu het geval, omdat we geen argumenten aan het programma meegegeven hebben en het programma die wel had verwacht:

```
System.out.println(args[0]);
System.out.println(args[1]);
```

We kunnen deze argumenten meegeven door:
Run → Set Project Configuration → Customize
In het scherm ziet u Arguments. Daar vult u in: Hallo Jan
Nu: Run → Run Project
U ziet dat u drie regels krijgt en de laatste zijn: Hallo Jan.

Oefenopgave 2

- Maak een nieuw project met de naam Hfdst1Oef2
- Zet in het tweede scherm het vakje Create Main Class uit.
  U ziet nu geen java file staan.
- Maak nu een class “Oef2” met een main methode.
  Met de rechtermuisknop opSource Pacakage → new → Java Main Class
  Class Name: Oef2
  Package: myPackage
- Zet de volgende code in main

```
System.out.print("ik geef Java ");
for(int i=0;i<5;i++) System.out.print("*");
System.out.println(" sterren");
```

- Zet uw naam als commentaar in de code (zie paragraaf 1.1).

Oefenopgave 3

We willen nu een package maken en importeren in ons eerste programma.

Met de rechtermuisknop op Source Packages
New → Java Package
Package Name: com.loi.student

Met de rechtermuisknop op com.loi.student
New → Java Class
Class Name: Persoon
Package: com.loi.student (dat staat er al)

We gaan nu eerst de variabelen toevoegen:

```
int id;
String naam;
String woonplaats;
```

Constructor
Een constructor kan gebruikt worden om aan de variabelen een default waarde te geven. Bij de class Persoon wordt id geïnitialiseerd op 0. De constructor is te herkennen doordat deze dezelfde naam heeft als de class. Er kunnen verschillende constructors gedefinieerd worden, allen met dezelfde naam, maar met verschillende argumenten (parameters die u meegeeft). Een constructor geeft nooit iets terug. Daarom staat er geen type voor de constructor en komt er binnen de constructor geen return voor. De constructors worden behandeld in paragraaf 3.5.

Nu gaan we een constructor toevoegen:
Source → Insert Code → Constructor
Vink alle drie de hokjes aan en klik op Generate.
U ziet dat de constructor is toegevoegd.

Methoden
Methoden gebruikt u om te communiceren met andere objecten en de buitenwereld. De methodes verzorgen het gedrag van het object. Andere objecten kunnen aan het gedefinieerde object vragen 'iets' te doen. De methodes worden behandeld in paragraaf 3.3. Getters en setters zijn gewone methodes die u automatisch kunt laten genereren door de IDE. Ze horen bij de variabelen die u reeds gedefinieerd hebt.

Nu gaat u de getter- en settermethoden toevoegen:
Source → Insert Code → Getters and Setters
Vink alle hokjes aan, en klik dan op Generate.

Nu hebt u een Java class gemaakt waar u weinig voor hoefde te doen.

Nu gaan we weer naar de class Java Structuur.java.

In de methode main voegen we de volgende code toe:
```
Persoon pers = new Persoon(12,"Klaas","Lutjebroek");
```

U ziet in de kantlijn een lampje met een rood bolletje. Er is kennelijk iets verkeerd gegaan.
Wanneer we daarop gaan staan, zien we de melding: can't find symbol class Persoon.
De class Persoon wordt niet gevonden, omdat we het import statement nog niet toegepast hebben.
Onder package zet u:
```
import com.loi.student.Persoon;
```

Vervolgens wilt u zien of het object Persoon bestaat door de inhoud ervan op het scherm (in de output) te tonen. Het is niet zo netjes om in elke class een methode print op te nemen. U wilt dat meestal alleen doen in de GUI. Nu we geen GUI hebben, willen we alleen printen in de class JavaStructuur. De andere classes houden we dan algemeen. Nu hebben we twee mogelijkheden:

- De getters van de class Persoon gebruiken
- Een methode aan de class Persoon toevoegen toString();

U doet beide.

Voeg toe in de methode main:
```
System.out.println("id = " + pers.getId() + ", naam = " + pers.getNaam()
+ ", woonplaats = " + pers.getWoonplaats());
```

Voer het programma uit.

De tweede manier:
Ga naar Persoon.Java
Voeg de volgende methode onderaan voor de laatste accolade toe:

```
public String toString() {
        return "id = " + id + ", naam = " + naam + ", woonplaats = " +
woonplaats;
}
```

Ga naar de methode main:
Voeg toe:

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

Zet u tussen de bovenstaande haakjes pers, dan ziet u een scherm met de methodes van Persoon verschijnen. Kies toString();

Voer het programma uit. U ziet nu 2 keer id = enz. staan.

De volledige code wordt hier getoond, het commentaar is weggehaald.

```
package javastructuur;

import com.loi.student.Persoon;

public class JavaStructuur {

    public static void main(String[] args) {
        String s = "loi wenst u veel succes met deze cursus";
        System.out.println(s);
        System.out.println(args[0]);
        System.out.println(args[1]);
        Persoon pers = new Persoon(12,"Klaas","Lutjebroek");
        System.out.println("id = " + pers.getId() + ", naam = " +
pers.getNaam() + ", woonplaats = " + pers.getWoonplaats());
        System.out.println(pers.toString() );
    }
}
package com.loi.student;

public class Persoon {
    int id;
    String naam;
    String woonplaats;

    public Persoon(int id, String naam, String woonplaats) {
        this.id = id;
        this.naam = naam;
        this.woonplaats = woonplaats;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNaam() {
```

```
        return naam;
    }

    public void setNaam(String naam) {
        this.naam = naam;
    }

    public String getWoonplaats() {
        return woonplaats;
    }

    public void setWoonplaats(String woonplaats) {
        this.woonplaats = woonplaats;
    }

    public String toString() {
        return "id = " + id + ", naam = " + naam + ", woonplaats = " +
woonplaats;
    }
}
```

Haal de import bij de class JavaStructuur weg. U krijgt nu een foutmelding. Gebruik de fully qualified name (paragraaf 1.3.4)

Oefenopgave 4a

U gaat nog even terug naar ons oude programma.
Ga naar Persoon.java.

```
14.  public class Persoon {
15.  int id;
16.  String naam;
17.  String woonplaats;
18.  //
19.  public Persoon(int id, String naam,String woonplaats){
20.  this.id = id;
21.  this.naam = naam;
22.  this.woonplaats = woonplaats;
23.  }
24.  //
25.  public int getId() {
26.  return id;
27.  }
```

Op Regel 14 ziet u public staan. U weet vanuit het boek dat dit betekent dat eenieder deze class mag benaderen.
Op regel 15, 16 en 17 staan geen access modifiers. Hier geldt de default.
De default voor een class, variabelen en methodes is niet public voor verschillende packages.

Probeer het eens uit: public weghalen.
Om de data te beschermen (encapsulation) is het beter om niet de default te gebruiken, maar bij de regels 15-17 private voor de variabelen te zetten.

```
    private int id;
    private String naam;
    private String woonplaats;
```

Maar hoe zit het wanneer u in dezelfde package bent?
Dat gaan we onderzoeken.
Maak een nieuwe class Student met dezelfde gegevens als de class Persoon in de package javastructuur.

Nu hebt u dus de volgende variabelen:

```
public class Student {
    int id;
    String naam;
    String woonplaats;
```

U moet ook de constructor, de getters, de setters en de methode toString() maken.

We gaan een nieuwe student aanmaken in de methode main van de class JavaStructuur onder Persoon pers = new Persoon(12,"Klaas","Lutjebroek");
Student student enz.
Nu willen we op de volgende manier id, naam en woonplaats tonen:

```
System.out.println("id = " + student.id + ", naam = " + student.naam +
", woonplaats = " + student.woonplaats);
```

De theorie uit het boek klopt dus: In dezelfde package is de default public.
Even testen: haalt u de acces modifier public voor de class Student weg, dan krijgt u geen foutmeldingen in de class JavaStructuur. Doet u hetzelfde bij de class Persoon, dan krijgt u een foutmelding bij de import. Misschien viel u nog een verschil op? We hebben geen import `javastructuur.Student` geschreven, dat komt omdat u in dezelfde package zit.
In de class Student hoort u voor de variabelen private te zetten, daarmee beschermt u de variabelen. De variabelen kunt u alleen benaderen door de methodes (bij voorbeeld getters en setters) die bij deze class horen. Wanneer u de variabelen private maakt, moet u de println aanpassen.

Oefenopgave 4b

- Maak in de package com.loi.student een nieuwe class Student.
- Dit is een afgeleide class van Persoon: Student extends Persoon {
- Zet daarin de volgende variabele: opleiding (welk type kiest u en welke modifier?)
- Maak een constructor met de argumenten: id, naam, woonplaats en opleiding.
- Maak nu een override methode toString, precies hetzelfde zoals u dat bij Persoon gemaakt hebt, maar nu uitgebreid met het veld opleiding. Waar loopt u dan tegenaan? Maak de modifiers van de variabelen in de class Persoon Protected!
- Test wat u gemaakt hebt door een student aan te maken en toString() aan te roepen.

Oefenopgave 5

U gaat nu een programma maken waarin de drie modifiers abstract, final en static verwerkt worden.

- Maak in Netbeans een nieuw project met de naam Dierentuin en zet het vinkje bij Create Main Class uit.
- Maak nu een abstracte class dier. Dit doen we door New → Java Class met als naam: Dier en als package: dieren
- Voor “class” zetten we abstract.
- U voegt de volgende private variabelen toe: naam, voedsel.
- U voegt de constructor toe.
- Daarna kunt u de getters en setters toevoegen.
- U voegt de abstracte methode kenmerken toe: public abstract String kenmerken();

U hebt nu een abstracte class gecreëerd met vier methodes en een abstracte methode.

Nu gaat u de afgeleide classes maken: Vogel, Zoogdier en Vis. Waarom wordt voor de namen enkelvoud gebruikt en niet meervoud?

- Maak nu een class Vogel
- Typ achter Vogel: extends Dier
- U krijgt nu een foutmelding: klik op het rode bolletje en klik op “Implement all abstract methods”

Het volgende wordt toegevoegd:

```
@Override
    public String kenmerken() {
        throw new UnsupportedOperationException("Not supported yet.");
//To change body of generated methods, choose Tools | Templates.
    }
```

- De foutmelding blijft (rood onderstreepte Vogel). Wanneer u de melding leest, blijkt dat u een default constructor moet toevoegen.
- Voeg een constructor toe aan de abstracte class dier met insert code (geen vakjes aanvinken)

Nu kunt u de class Vogel afmaken.

- Voeg de private variabelen boolean tam en String soort toe.
- Voeg de constructor toe. Vink het vakje van Dier aan, waar staat: Dier(String naam, String Voedsel) en de vakjes tam en soort.

Nu kunt u de methode kenmerken afmaken.
Een voorbeeld van code wordt gegeven. In de volgende hoofdstukken zult u deze code zelf bedenken.

```
   @Override
    public String kenmerken() {
        if (tam)
        return getNaam() + "behoort tot de soort: " + soort + " eet " +
getVoedsel() + "is tam";
        return getNaam() + "behoort tot de soort: " + soort + " eet " +
getVoedsel() + "leeft in het wild";
    }
```

Nu test u de classes die u gemaakt hebt. Daarvoor schrijft u een class TestDier. Dat doet u in hetzelfde project en in dezelfde package.

- Maak een class TestDier. Dat doet u met new, kiezen other. U krijgt een scherm en daarin kiest u Java Main Class: naam: TestDier, package: dieren
- Schrijf nu methode main

```
public static void main(String[] args) {
      Dier dier = new Vogel(true, "duif", "Tortelduif", "mais");
      System.out.println(dier.kenmerken());
}
```

- Voer de code uit en u ziet de kenmerken in de Output.

Hieronder worden een aantal eigenschappen van abstract gegeven. De dikgedrukte eigenschappen staan als commentaar in de code die u aan het einde van dit hoofdstuk terug kunt vinden.

- Van een abstracte class kan geen instantie worden gemaakt, dus er kan geen object van worden gemaakt.

```
Dier dier = new Dier("Tortelduif", "mais");// kan niet want dier is
abstract!
```

- Een **abstract method** is een method **zonder body**.
- **Abstracte methods moeten altijd geïmplementeerd** worden in een **concrete subclass**.
- Een **abstracte class** mag wel **als type** worden gebruikt.
- Een abstracte class kan zowel abstract als concrete methods bevatten. **Ga dat na in de code van Dier.**

- Een concrete class kan geen abstracte methods hebben. Probeer maar!

Nu rest nog aandacht te besteden aan final en static.

- Voeg de volgende variabele toe aan de abstracte class dier.

```
static int aantal = 0;
```

- Voeg de volgende final methode toe aan de abstracte class dier.

```
final int geefAantal() {
        return aantal;
}
```

Zodra een nieuwe instantie van een class afgeleid van dier wordt gemaakt, dient dit aantal te worden verhoogd. Daarom moet de volgende code aan de default constructor toegevoegd worden:

- Voeg de volgende code toe aan de default constructor: . aantal++;
- In de andere constructor moet op de eerste regel toegevoegd worden: this();
- In TestDier voegt u toe:
  `System.out.println("het aantal = " + dier.geefAantal());`
- Test nu de code en u ziet in de output: het aantal = 1
- Probeer nu een tweede vogel aan te maken met `Vogel diera = new Vogel(`

Hieronder worden een aantal eigenschappen gegeven van final en static.

- De static variabelen en methods behoren bij een class en niet bij een object. Er is maar één kopie. U kunt dat nagaan. U hebt in het programma 2 instanties gemaakt, maar het variabele aantal is maar 1x aangemaakt en de output geeft 2 aan.
- Static final-variabelen zijn constanten.
- Een final method kan niet overrided worden. Probeer de methode “geefAantal()” maar te maken in Vogel. U krijgt dan een foutmelding.
- Van een final class kan geen subclass gemaakt worden. M.a.w. wanneer de class dier final zou zijn, kunt u daarvan niet de class vogel afleiden.

Wanneer u meer wilt oefenen, kunt u op gelijke wijze de Zoogdier en Vis maken.
Veel succes!