Java

Klassen

Inhoud

De basics

- ➔ Wat is een klasse
- ➔ Wat zijn Objecten
- ➔ Objecten maken
- ➔ Constructors
- ➔ Referentie variabele

De klasse String

- ➔ Korte inleiding
- ➔ Gebruik van de klasse String
- ➔ Declaratie en initialisatie van een String object
- ➔ String literal
- ➔ Immutable
- ➔ String constant pool
- ➔ Java API doc

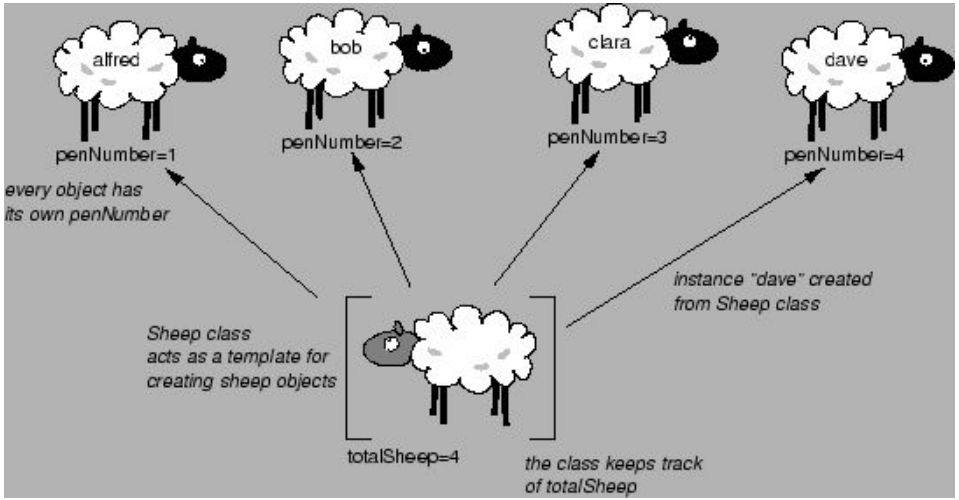
Demo

- ➔ methodes van de klasse String

Klassen

Een klasse

Een blauwdruk die de eigenschappen en methoden van objecten van dezelfde soort bepaalt.



Objecten

Ook wel instanties (instances) van een klasse genoemd.

Door het onderscheid tussen klassen en objecten kan men ook een onderscheid maken tussen het soort variabelen en methoden:

1. Instance-variabelen en methoden

2. Klasse-variabelen en methoden

Objecten maken

Objecten worden gemaakt op basis van een blauwdruk. Deze blauwdruk is de klasse waartoe het object behoort.

Declaratie van het object

Random rand;

➔ Hier gaan we het Random object declareren.

Initialisatie van het object

rand = new Random();

➔ Het toekennen van een waarde aan het object in dit geval door het oproepen van een constructor

Constructors

De constructor is een **speciale methode** die **dezelfde naam** draagt als de **klasse** en die enkel gebruikt wordt om het object tijdens de creatie te **initialiseren.**

Iedere klasse heeft één of meerdere constructors die van elkaar verschillen in het aantal en type van de parameters.

Random()
Random(long seed)

➔ Een object kan daardoor op verschillende manieren geïnitialiseerd worden.

Referentie variabele

Type name = initialValue;

Random rand = new Random();

Tijdens de **creatie** van een **object** wordt er **geheugen** gereserveerd voor het object.

Het resultaat van de creatie is dan ook een **verwijzing** naar dit nieuwe object in het geheugen.

Die verwijzing hebben we later nodig om het object in ons programma te kunnen gebruiken.

Een referentievariabele die naar geen enkel object verwijst, heeft de waarde **null**

We kunnen de compiler ook het datatype laten afleiden, omdat we een declaratie en objecreatie in één statement hebben

var name = initialValue;
var rand = new java.util.Random();

Pakket Importeren

```
Import java.util.Random;
Random rand = new Random();

Import java.util.*;
var rand2 = new Random();
```

Java importeer klassen in een bepaalde volgorde.

Java importeert automatisch volgende pakketten

1. default
2. huidige
3. java.lang

java.lang is een pakket met enkele basispakketten van Java

Zie API

Random gebruiken

Geen publieke eigenschappen => **data hiding.**

Wel publieke Methoden

Naam van methode plus evt. gegevens meegeven.

Soms gegevens terug voor verdere verwerking.

naamObject.naamMethode(int par1, int par2);

rand.nextBoolean();
rand.nextInt();
rand.nextInt(int Bound);
rand.nextLong();
rand.nextFloat();
rand.nextDouble();

Garba

This is Jav

Java does not le
mess for others

It collects it's ow

Be like Java.
Keep your country

creatie betekent geheugen reservatie.

talen zoals c ++ moeten zelf hun objecten
en op Memory OverFlow tegen te gaan.

imt zelf de ongebruikte objecten op uit memory

d in een afzonderlijke Thread en bijna geen
op rest van programma.

ırten:
System.gc();

ker dat ze meteen start.

Inleiding

De klasse String

De String klasse is één van de meest gebruikte klassen binnen Java. Het is dan ook een ietwat speciale klasse, waarom zien we later.

Instanties van deze klasse zullen voornamelijk gebruikt worden voor het opslaan en bewerken van tekst.

Strings zijn **immutable,** en kunnen dus na het creëren van het object niet meer gewijzigd worden.

We kunnen een object van deze klasse op verschillende manieren aanmaken en zullen deze later in de cursus zien.

Waarvoor kunnen we de klasse String gebruiken ?

Zoals reeds eerder vermeld gaan we de String klasse gebruiken voor het bewerken en opslaan van tekst.

➔ onveranderlijke tekenreeksen.

Enkele voorbeelden :

➔ het bewerken van een individuele character.
➔ het vergelijken van Strings onderling.
➔ een deel van de String kopiëren in een nieuwe String.
➔ het zoeken in een String intern.
➔ Een kopie van de String maken en alle characters omzetten naar lower/uppercase.

Declaratie & initialisatie

Het maken van een String object

Een String object kan op verschillende manieren gecreëerd worden.

We zien hier enkele voorbeelden:

String s = “Hello World”;

String s = new String(); **//nutteloos zien later waarom**

String s = new String(“Hello World”);

Naargelang het type en aantal parameters wordt de juiste constructor aangeroepen om het object te initialiseren.

String literal

String literals bestaan uit een reeks characters tussen **dubbele** aanhalingstekens.

enkele voorbeelden:

- ➔ “abc”
  Deze kan dus bestaan uit letters.
- ➔ “H3ll0”
  Er kunnen ook cijfers in een String Literal
- ➔ “m0n3y$”
  Net als speciale tekens

Java zal elke String Literal automatisch omzetten in een object van de klasse String

Immutable

Wanneer we in Java een object van het type String aanmaken wordt aan die String een initiële waarde toegekend. Deze waarde kunnen we nadien niet meer veranderen.

String text = new String(“Hello World!”);

- ➔ We creëren hier een object van de klasse String en we kennen via de constructor onmiddellijk de waarde “Hello World” toe.

- ➔ De variabele text is een referentie naar het object.

Aangezien een String literal behandeld kan worden als een String object kan men gebruik maken van de eigenschappen en methoden van een String object

De String constant pool in java

Doordat strings onveranderbaar zijn [**immutable**] is het **mogelijk** een **pool bij te houden** van String objecten.

➔ Deze pool noemen we dan ook de **String constant pool**.

Wanneer een nieuw String object gecreëerd wordt d.m.v een String literal, gaat de VM na of deze zich reeds in de pool bevindt.

➔ Indien de VM een referentie vindt in de String Pool geeft hij deze referentie terug.

➔ Dit **vormt geen probleem omdat de strings onveranderlijk zijn**. We **kunnen** deze referentie **dus delen**.

Opgelet!
Indien we echter de constructor gebruiken om een String object te maken, wordt er wel een nieuw object gemaakt

String.equals()

Door het bestaan van canonieke strings in de string pool is het gedrag van == onbetrouwbaar.

Vergelijken met == **IS NOT DONE!**

De .equals methode zal de inhoud van de String vergelijken ongeacht hun plaats in geheugen.

Java API

Oracle heeft een online documentatie waarin we de Java klassen eens dichter kunnen bekijken.

Om te zien welke publieke eigenschappen en methoden de klasse String heeft kunnen de Java API documentatie eens bekijken.