Design patterns

W. Oele

16 september 2014

Deze les

- Factories
- Het abstract factory patroon

Factories: het probleem

Code:

- houdt zich bezig met het creëren van objecten
- houdt zich bezig met het gebruiken van objecten

Voorbeeld

```
public class Client
{
  public void maakenGebruik()
  {
     Auto a = new Auto();
     a.startMotor();
     a.rijNaar(Rotterdam);
     a.stopMotor();
  }
}
```

Factories

Code die zich bezig houdt met het creëren *en* gebruiken van objecten:

Factories

Code die zich bezig houdt met het creëren *en* gebruiken van objecten:

- client moet weten *wat* er gecreëerd moet worden en *hoe* het gecreëerde werkt.
- high coupling
- lastig te onderhouden
- derhalve ongewenst

Factories

Uitdaging: Hoe het *hoe* te scheiden van het *wat*?

Factories

Uitdaging: Hoe het *hoe* te scheiden van het *wat*?

- Client tegen verkoper: *"Ik heb een schroevendraaier nodig. . . "*
- Verkoper: *"Wat voor één (groot/klein/kruiskop/torq/flat)?"*
- Client: *" Eén die werkt (kan mij wat schelen hoe ie werkt, als ie het maar doet)!"*

Factories: de client

Het client object:

- wil op bepaalde momenten kunnen beschikken over objecten die iets doen.
- is niet geïnteresseerd in de interne werking van die objecten (het hoe).
- is geïnteresseerd in *wat* en *wanneer*.

Factories

De maker:

- houdt zich bezig met de vraag *hoe* een object gemaakt moet worden.
- is belast met varianten
- is geïnteresseerd in de eisen die aan een object worden gesteld en *hoe* daaraan te voldoen.

Ontwerpen

Bij het ontwerpen:

- de *hoe* vraag scheiden van de *wat* vraag.
- eerst nadenken over het wat:
  - entiteiten:
  - attributen
  - operaties
  - etc.
- daarna: nadenken over het *hoe* en wie in je programma daar voor opdraait.

Ontwerpen

Algemeen:

- Denk eerst na over *wat* je nodig hebt in je programma en aan welke eisen dat moet voldoen.
- Denk *daarna* pas na over wie dat gaat bouwen en hoe dat moet gebeuren.

Factories

- Client: *"Ik heb een auto nodig!"*

Factories

- Client: *"Ik heb een auto nodig!"*
- Verkoper: *"Wat voor één meneer?"*

Factories

- Client: *"Ik heb een auto nodig!"*
- Verkoper: *"Wat voor één meneer?"*
- Client: *"eentje met: "*
  - *"vierwielaandrijving"*

Factories

- Client: *"Ik heb een auto nodig!"*
- Verkoper: *"Wat voor één meneer?"*
- Client: *"eentje met: "*
  - *"vierwielaandrijving"*
  - *"bestand tegen stof"*

Factories

- Client: *"Ik heb een auto nodig!"*
- Verkoper: *"Wat voor één meneer?"*
- Client: *"eentje met: "*
  - *"vierwielaandrijving"*
  - *"bestand tegen stof"*
  - *"airco die aan en uit kan"*

Factories

- Client: *"Ik heb een auto nodig!"*
- Verkoper: *"Wat voor één meneer?"*
- Client: *"eentje met: "*
  - *"vierwielaandrijving"*
  - *"bestand tegen stof"*
  - *"airco die aan en uit kan"*
  - *"kooiconstructie"*

Factories

- Client: *"Ik heb een auto nodig!"*
- Verkoper: *"Wat voor één meneer?"*
- Client: *"eentje met: "*
  - *"vierwielaandrijving"*
  - *"bestand tegen stof"*
  - *"airco die aan en uit kan"*
  - *"kooiconstructie"*
  - *"motor die aan en uit kan"*
  - *"variabele snelheid"*
  - *"...waarmee ik Parijs-Dakar kan rijden"*

Ontwerpen

- Hoe een object te bouwen dat *gegarandeerd* voldoet aan de eisen die een client object daaraan stelt?

Ontwerpen

- Hoe een object te bouwen dat *gegarandeerd* voldoet aan de eisen die een client object daaraan stelt?
- Antwoord:

INTERFACES

Door na te denken over de interfaces, garandeer je bepaald gedrag bij de objecten die in je programma gebruikt worden.

Het abstract factory pattern

- naam: het abstract factory pattern
- doel: families van objecten aanbieden aan clients
- hoe: gebruik van objecten scheiden van implementatie
- gevolgen: nette boedelscheiding tussen wat/wanneer en hoe

Het abstract factory pattern

Een verkoper verkoopt verschillende soorten gereedschap:

- hamers
- boren
- schroevedraaiers
- etc.

Voor elk van deze soorten gereedschap bestaan er verschillende kwaliteiten:

- goed/duur(zaam?)
- standaard
- goedkoop

Het abstract factory pattern

- De verkoper wenst verschillende voordeelpakketten met gereedschap aan te bieden in zijn winkel.
- De verkoper koopt zelf zijn gereedschap in bij een fabriek die die gereedschappen maakt.

Het abstract factory pattern

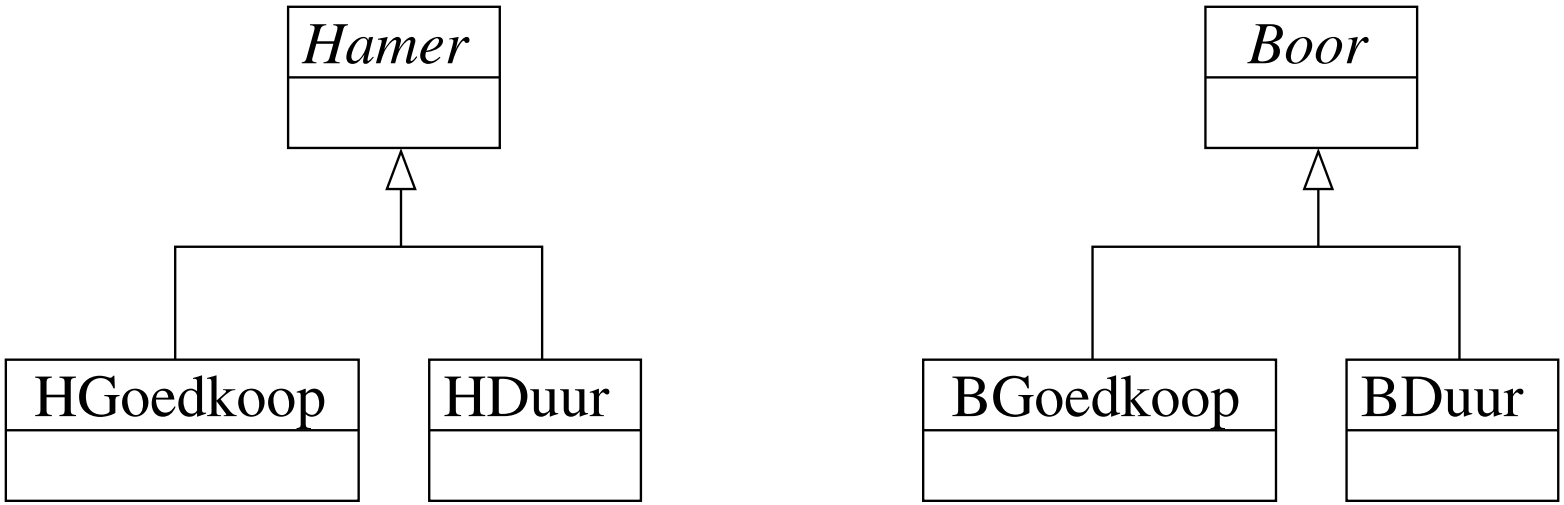
Het abstract factory pattern

- De verkoper wenst te beschikken over verschillende soorten gereedschap. . .

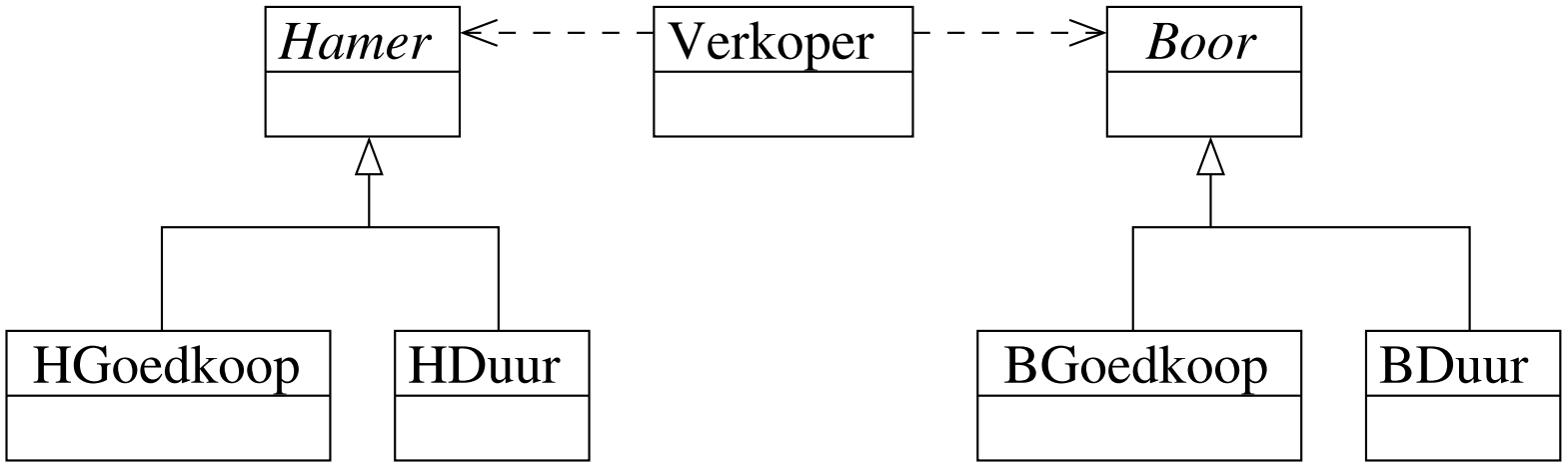
Het abstract factory pattern

- De verkoper wenst van elk soort gereedschap een dure en goedkope variant te kunnen kopen. . .

Abstracties



Het abstract factory pattern



- De verkoper verkoopt *een* boor en *een* hamer...

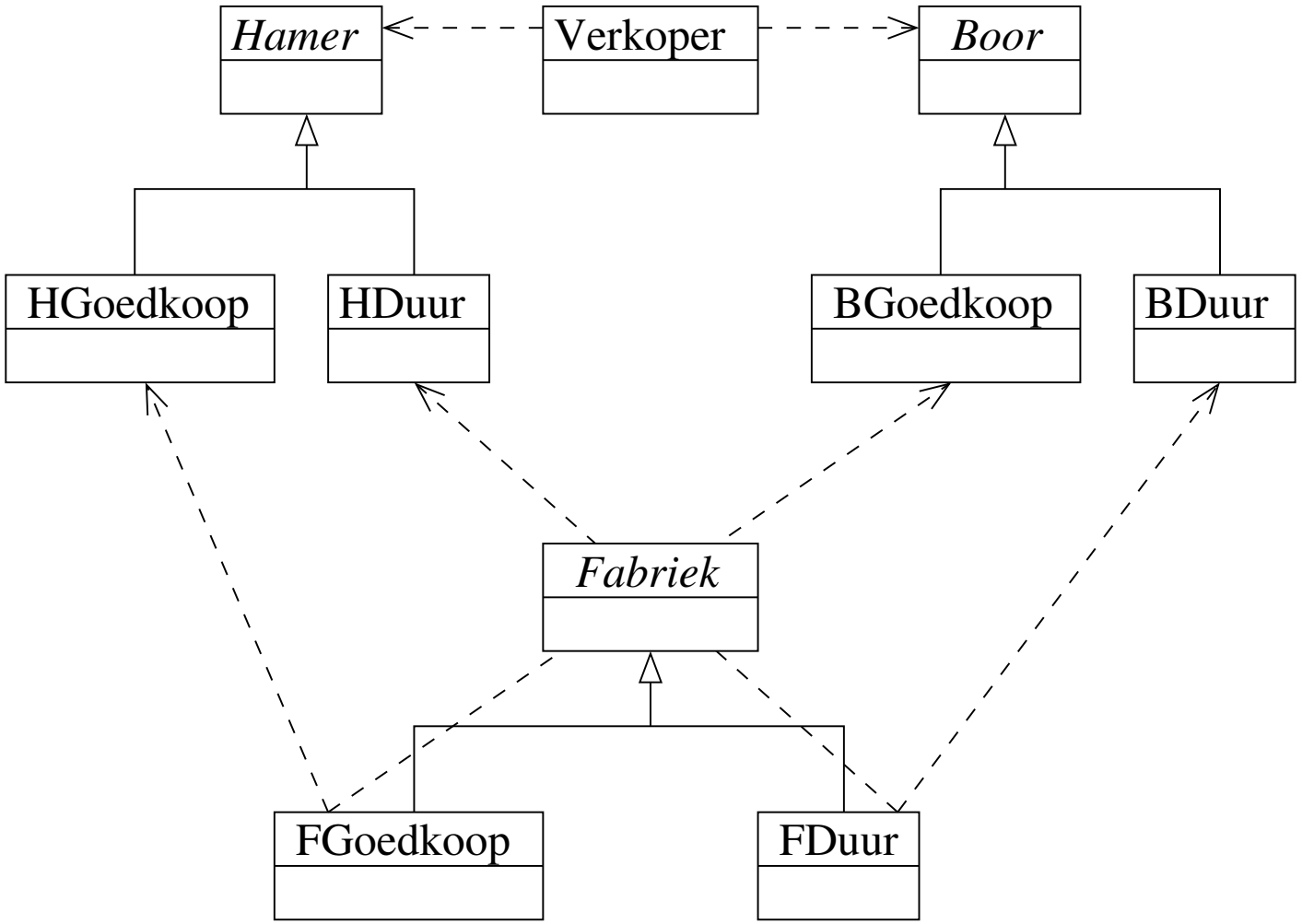
Het abstract factory pattern

De verkoper:

- bestelt gereedschap bij de fabriek
- vertelt de fabriek slechts:
  - welk gereedschap
  - wanneer en
  - van welke soort

Hoe die fabriek die spullen maakt en levert is de verantwoordelijkheid van de fabriek.

Het abstract factory pattern



Het abstract factory pattern

- Naam: abstract factory pattern
- Doel: omgaan met families van objecten
- Hoe: creatie van objecten en *hoe* dat gebeurt, scheiden van gebruik.
- Gevolgen: nette boedelscheiding hoe en wat vragen $\rightarrow$ beter onderhoudbare en uitbreidbare code

Het abstract factory pattern algemeen