

# Diensten en Domein

een blik op domein gedreven ontwikkelen

**Ralf Wolter**

[rwolter@sogyo.nl](mailto:rwolter@sogyo.nl)

- Het centrale probleem van software ontwikkelen is het beheersen van complexiteit.
  - Complexiteit van bedrijfsproces
  - Complexiteit van platform
  - Complexiteit van onderhoud

- Principes om complexiteit te beheersen.
  - Minimale koppeling
  - Maximale cohesie
  - Schaalbaarheid.

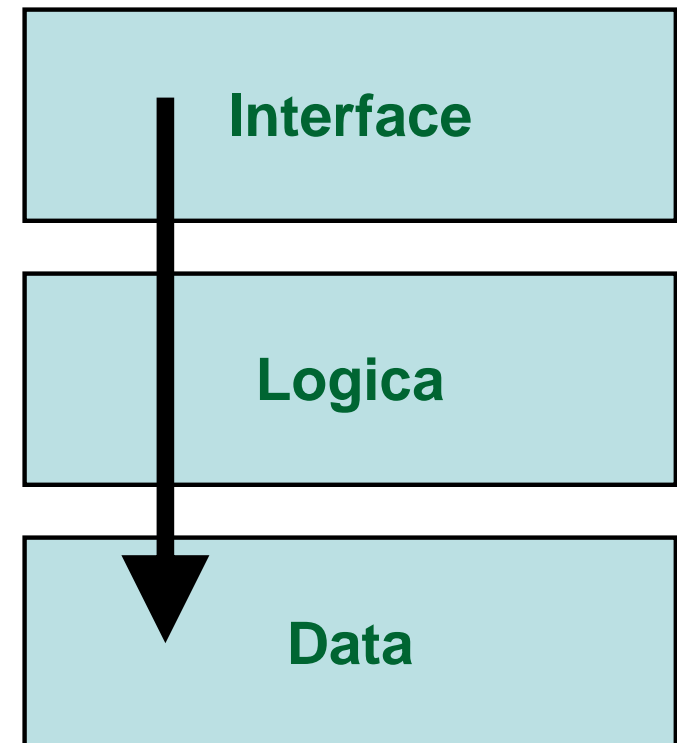
- Een model met drie of meer lagen.
- Genoemde Redenen:
  - Opsplitsen van interface, logica en data
  - Betere onderhoudbaarheid
  - Betere schaalbaarheid

Interface

Logica

Data

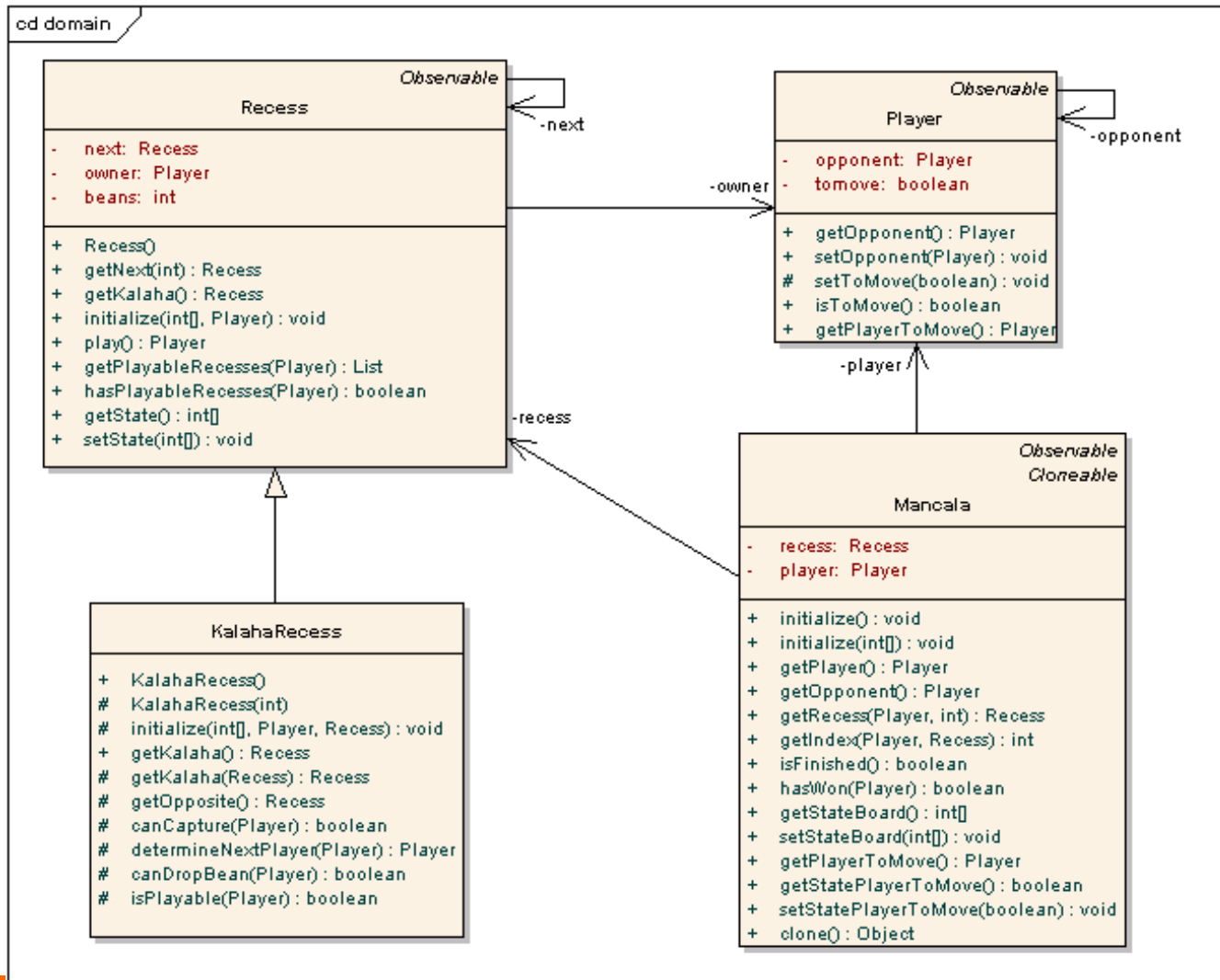
- In praktijk:
  - Aanroepen van boven naar beneden
  - Hechte koppeling tussen lagen
  - Focus op presentatie en opslag andere aspecten passen moeilijk in het ontwerp.



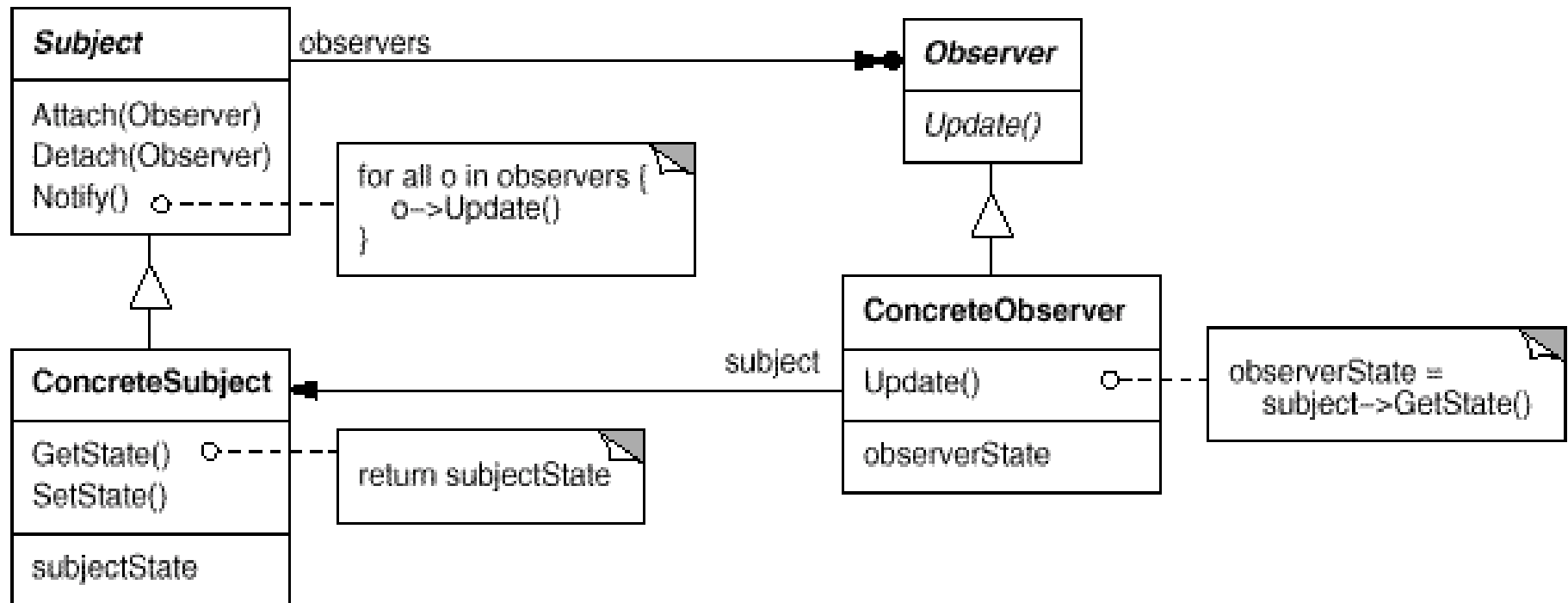
- Het belangrijkste onderdeel van ontwikkelen is het modelleren van het domein en de domein logica
- Complexe domein ontwerpen moeten gebaseerd zijn op een model.

- Een domein model is een representatie van 'real-world' conceptuele klassen. *(bron: [Eliens])*
- Geen software onderdelen
- Geen afhankelijkheden





- Communicatie naar model:
  - Aanroepen: Directe communicatie met model, dit vereist dat de aanroeper complete of gedeeltelijke kennis heeft van het model.
- Communicatie van model:
  - Events: Een indirecte aanroep vanuit het model. Hiervoor is geen kennis nodig en blijft onafhankelijkheid behouden.
    - Implementatie: observer pattern



Bron: [Gamma]

- Onafhankelijk
- Standalone
  
- Domein Model *is* de applicatie en kan functioneren zonder andere onderdelen.

- Demonstratie gebruik domein model zonder toevoegingen.
- *Gebruik Eclipse Scrapbook page om mancala te spelen.*

```
SpeelMancala.jspage x
Mancala mancala = new Mancala();

mancala.initialize();

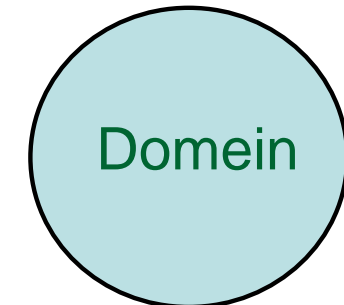
System.out.println(mancala.toString());

mancala.getRecess(mancala.getPlayerToMove());
System.out.println(mancala.toString());
```

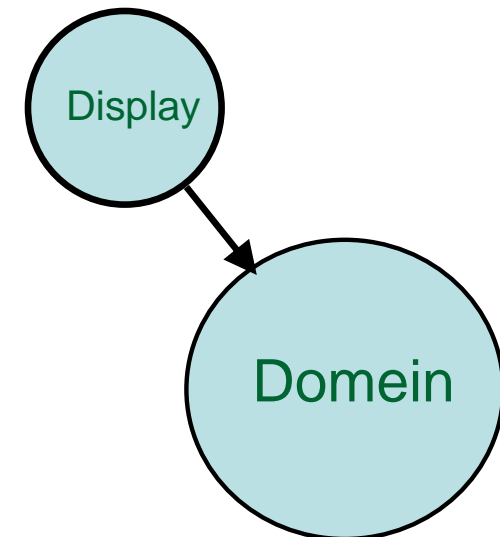
- Het domein model is de essentie van de applicatie, maar een applicatie moet vaak meer kunnen.
- Alle andere onderdelen zijn diensten.
  - Persistentie
  - Logging
  - Notificatie
  - Gebruikers Interface

- Een dienst is alles dat ondersteuning levert voor het domein model.
  - Reageert op domein
  - Initieert acties op domein
  - Onafhankelijk van andere diensten
  - Generiek voor verschillende domeinen

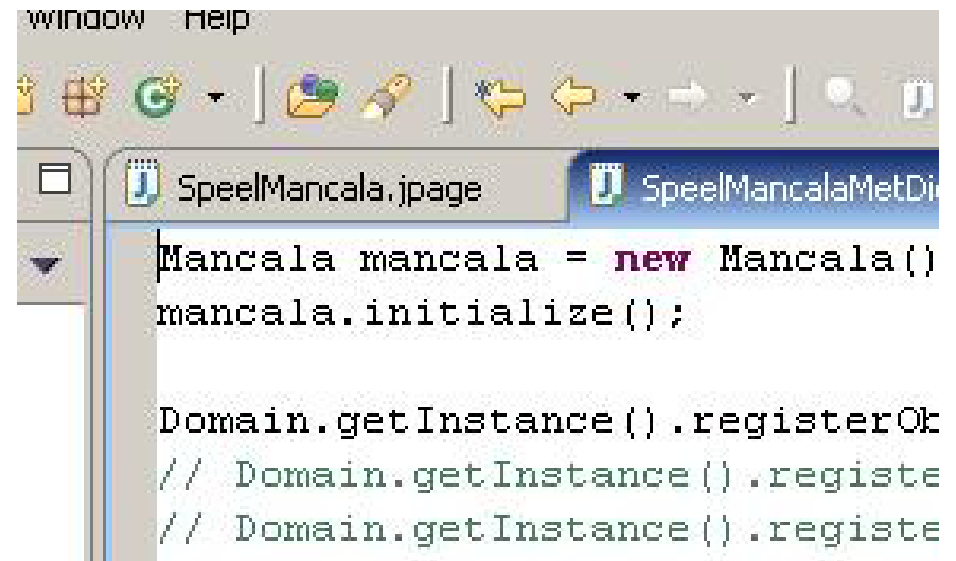
- Domein staat centraal.
- Functioneert zonder diensten.



- Domein staat centraal.
- Functioneert zonder diensten.
- Maar ook met een dienst zoals weergave.



- Vorige voorbeeld was Mancala zonder dienst, nu mancala met weergave dienst.
  - Reageert op veranderingen.
  - Aanwezigheid heeft geen effect op domein.

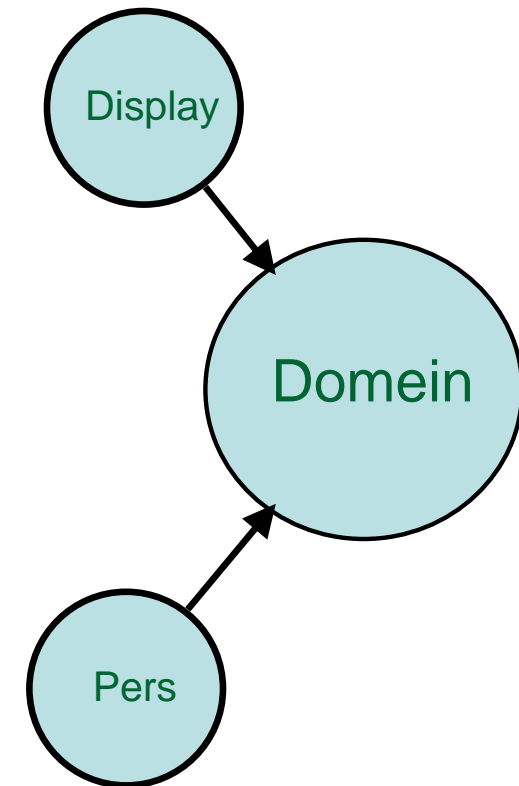


```

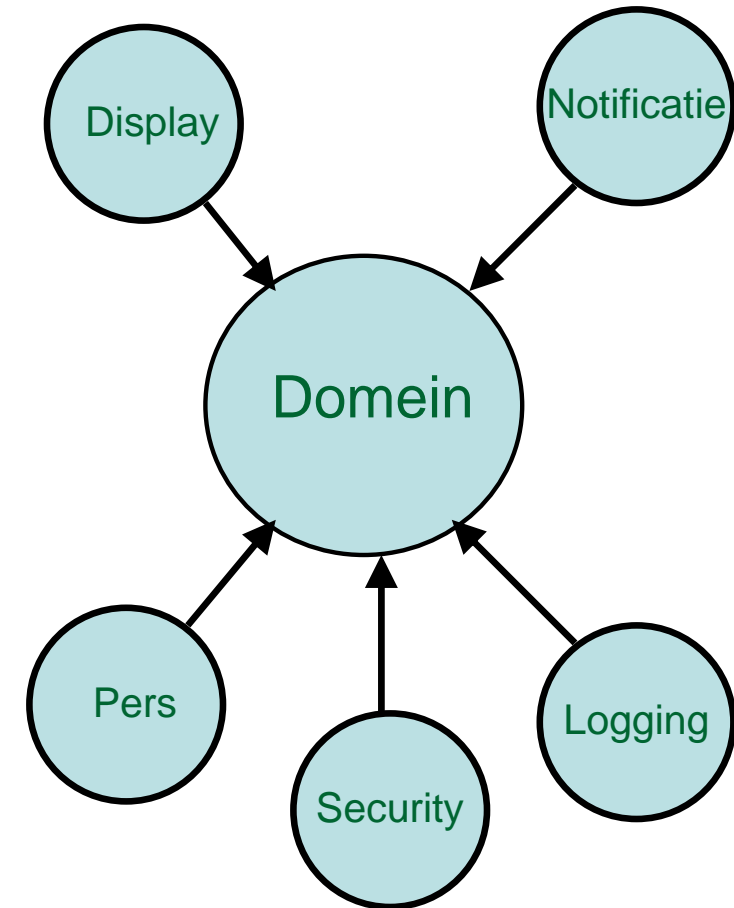
window Help
+ + + | + | * ← → | |
SpeelMancala.jpage | SpeelMancalaMetDi
Mancala mancala = new Mancala()
mancala.initialize();

Domain.getInstance().registerOk
// Domain.getInstance().registe
// Domain.getInstance().registe
    
```

- Een dienst is alles wat *niet* tot het domein behoort.
- Presentatie is een dienst.
- Persistentie is een dienst.
- Tussen diensten is geen verschil, presentatie of persistentie zijn niet belangrijker dan andere diensten.



- Diensten zijn eenvoudig toe te voegen.
- Maar ook eenvoudig te verwijderen.
- Ze staan immers los van het model en de andere diensten.



- Diensten voor mancala.
  - User Interface.
  - Persistentie.
  - AI Speler.
  
- *Voorbeeld mancala diensten.*



```
Mancala mancala = new Mancala()
mancala.initialize();

Domain.getInstance().registerOk
// Domain.getInstance().registe
// Domain.getInstance().registe
```

- Door domein gedreven ontwikkelen is de koppeling tussen domein en diensten lossier.
- Domein onafhankelijk van diensten.
  - Transporteerbaar.
  - Testbaar.
  - Onderhoudbaar.
  - Evolveerbaar.

- Door domein gedreven ontwikkelen leveren diensten een dienst aan het domein, maar is het domein daar niet van afhankelijk.
- Diensten onafhankelijk van domein.
  - Plugbaar.
  - Isoleerbaar.
  - Robuust.
  - Onderhoudbaar.

- **[Gamma]: Design Patterns: Elements of Reusable Object-Oriented Software, Gamma et al, Addison-Wesley (1994)**
- **[Evans]: Domain-Driven Design: Tackling Complexity in the Heart of Software, Evans, Addison-Wesley (2003)**
- **[Vens]: <http://www.sepher.nl/>**
- **[Eliens]: Principles of Object-Oriented Software Development, Eliens, Addison-Wesley (2000, 2th edition)**



Landgoed Sandwijck  
Utrechtseweg 301  
3731 GA De Bilt  
Tel:030 220 22 16  
Fax:030 220 55 06  
Mail:[rwolter@sogyo.nl](mailto:rwolter@sogyo.nl)