

edwin.vanouwerkerkmoria@hku.nl

## Vorige keer

- C++ 11 feature: auto

Laat de compiler uitvogelen wat 't type moet zijn...

Vooral: declaratie variabelen. Sinds C++ 14 ook return-types van functies

## SOLID principes

single responsibility,open/closed,Liskov,Interface segregation,Dependency reversal

## - Design patterns

specifiek: Strategy

## Opdracht vorige keer

Maak een (tekst-mode) implementatie van Conway's *Game of Life* <a href="https://en.wikipedia.org/wiki/Conway%27s\_Game\_of\_Life">https://en.wikipedia.org/wiki/Conway%27s\_Game\_of\_Life</a>

Vul een matrix (25x25) random met een cel. Voor iedere volgende generatie bereken het aantal buren: Cellen met <2 of >3 buren sterven. Cellen met precies 2 of 3 buren overleven. Lege cellen met 3 buren krijgen nieuwe cel. Herhaal...

- Maak classes voor de verschillende entiteiten
- Implementeer de standaardregels (hierboven) volgens 't strategy-pattern.
- Maak 2 alternatieve implementaties van deze strategy

# Vandaag

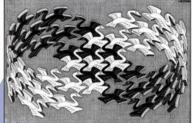
# Templates!

generic programming in C++

Design pattern: **Decorator** 

Elements of Reusable Object-Oriented Software

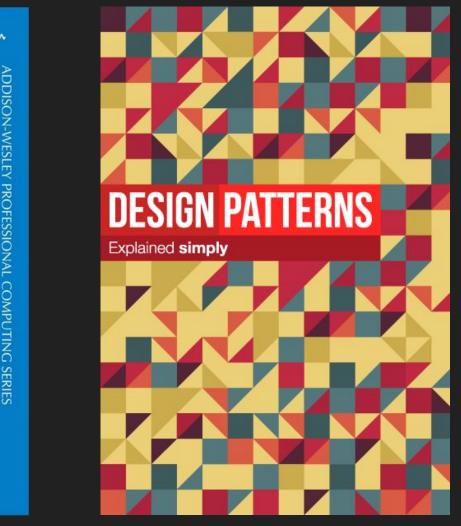
Erich Gamma Richard Helm Ralph Johnson John Vlissides



Course at C 1994 ALC Feches / Corrion Art - Basen - Holland All rights manus

Foreword by Grady Booch





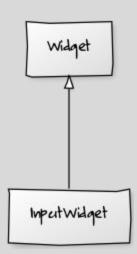


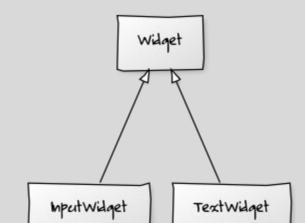
## Design pattern: Decorator

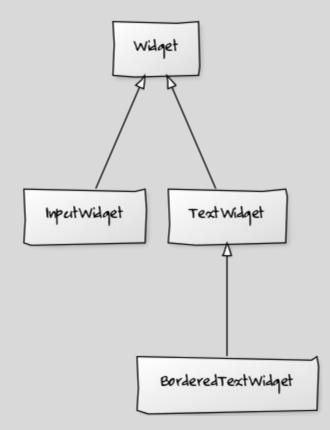
Koppel uitwisselbaar gedrag los van inheritance: voorkom diepe inheritance-hierarchie, voorkom code-duplicatie

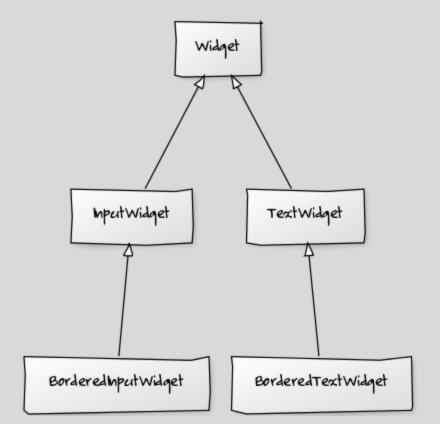
- Maak een baseclass
- Maak een tweede baseclass voor het uitwisselbare gedrag
- Maak subclasses voor specifieke versies van 't gedrag
- Koppel (runtime of compile-time) gedrag aan gebruiker

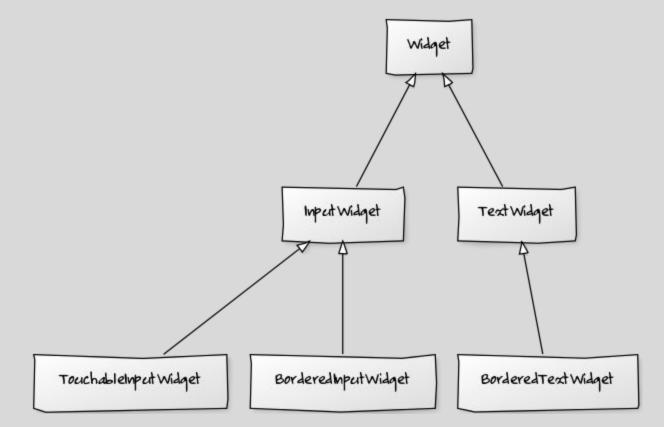
Widget

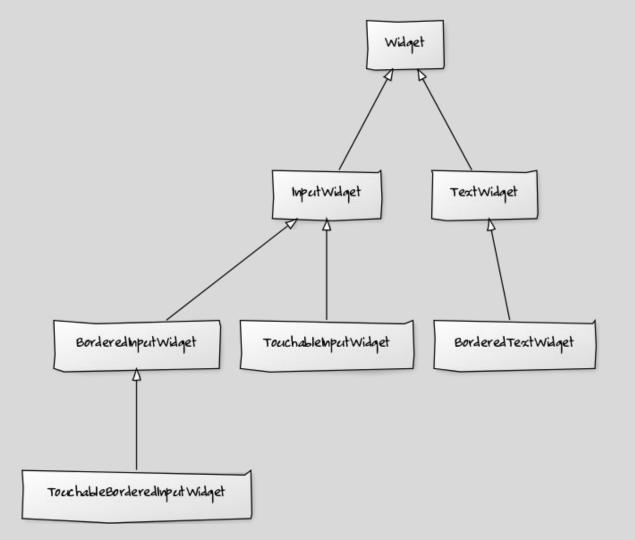


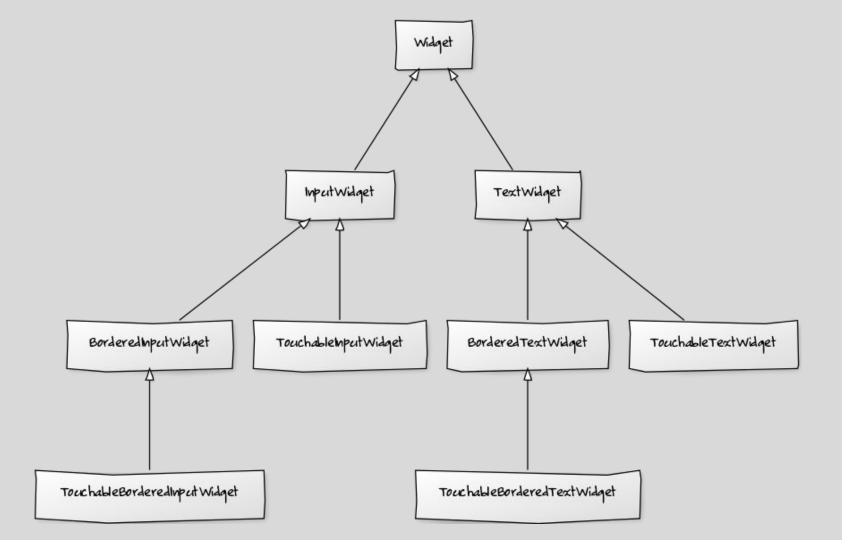




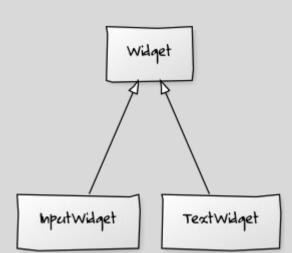


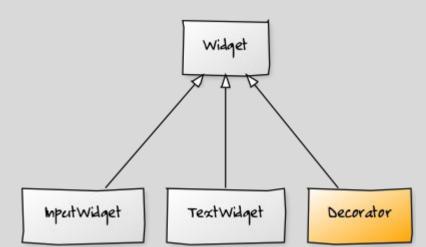


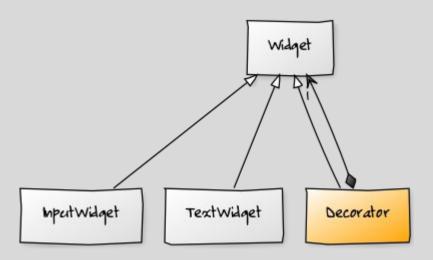


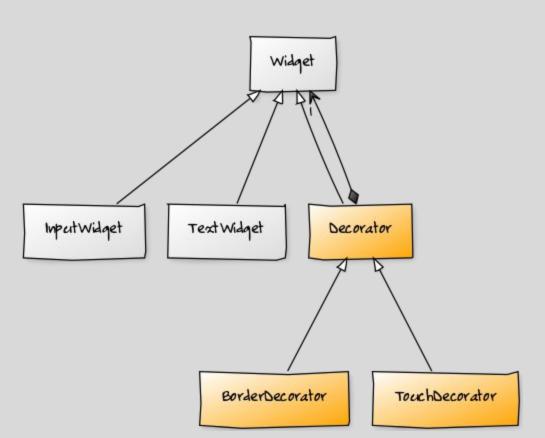














### LOREM IPSUM

Lorem Ipsum-dolor sit armet, consectious adipscrip ett, sed dam honummy nich eutsmod strodunt ut lacreet delore magna sitiguam ett coutpat. Ut wiel einem ad minim venlam, quis nostrud einem steon ultamooper suspipit labortan riels utalgulp ex di commodo consequa. Duri autem et eum insue dodn'i in herostint in et eum insue dodn'i in herostint in hendrer's in sulputate velit esse molecule opinisquat, veli flum dobce eu licepular nulta facilisis at vero eron et accumian et lustro odo dignissimi qui blanda praesient luptatum zoni deleni auque dus dolore le leugati nulta facilisi. Nami liber tempor oum soluta nobis elederd option, conque nitri imperdett doming er Lorem bysom dolor sit amer, consectation may vista adjoiscing ells, and dans nonummy with maismoders tracidant of lacered dolore magna aliquam eratisellin vollutest. Un wisi entim ed minim verbane, quistranceir mostrud exect ration ultimoroper exactopis bolomisellin mai ut aliquip ex ex commodo consequat. Dississani autem sel eximitare dolor in herativit in volgutation veila; esse molestite consequat, veil illum dolore eurox, fergair multis facilistis at vere more et accument eratification dolore della propositional dell



TEXT HERE









### HEADLINE

i germ lopum dollor sid amet, conspecteur a dipriscing ells, sedio denn nonumm with quaimod finach at la toered deliche magne sliquam erat volutgas. Ut visil erinn ad minim veniam, quistiliam notatud leseri statu ultiercorper suspect toboris melli di sliquim es es commodo consequat. Duis sudem vell eum inure dollor reta deliche in vulgutate velst case moloste consoquat, del filmini, doller eu livugit hulla fiscilias at versi error et accumismin els visib

## **HEAD** LINE

Lorent (sourn Gilvor (ri) siene; connectativer adoptacing ett, sed diem nonumente mith eutemod finci, hir at lacered dolore magna allquam enat volutgat. Ut visid entim ad mir im verbiim, gipis notatud exent lation ultamoorper sussight bloomisnial eri aliqo(hir ir is commodo consequat. Duis autem vel eum time dolor insite hir middent eri uglatare vielt exes modeste consequat. Ett illem dolore ere fregist malle hir libias at vero eros et accumaan et lustro odio digrissim qui bianditerani gressen hystemm zoril debent augun duis dolore te fregist milla facilità. Nation promocom solution arbite elettre opport orius promoter domining del pur lacere del mitter del promoter del promoter del promoter del promoter tempor cum solutar arbite elettre opport orius promoter del promoter promoter con solution arbite elettre opport orius promoter del promoter del promoter del promoter del promoter promoter con solution arbite elettre orius promoter del promoter del promoter promoter del promoter promoter del promoter promoter del promoter promoter



000

#### **FLYER TITLE**

Lorem (paum dider ett amet, ennecetebaer adjacing ett), end tim nonummy nibbut, exismod fincishert (\*\*Lorent delorestrand magna allquis' n'en' judupat. Ut sisi enin ad miss' vien'an, qui anostud eserotion taleiu-yillumoopse suurigis lidomis nisil ill allquis et ae commodo consequet. Duvis autem sel sum inius dolor in handwit hall vulputata veilli essa molestis donorquistot veil illum dolore un fangist nulla facilista al veno erso et soumsan et lustro odirumnes dignissim qui allandi paesen luptumini. Zirili delemit sugar duis dolore la fangista nulla facilisti. Nam liber tempor cumsoon soluta noble siellend option conque nibile



## HEADLINE

Lorieri (psum dolor sit amet, consectetuer adiptiscing ellit, sed dilam nonummy nilihut, culsmod disciduant ut lacered dolorestandi magna aliquam erat volutgast. Ut wisi edin adminim veniam, quis nostrud exercitora tation ullamoroper suscipit loborits nilai illi aliquip ax ea commodo consequat. Duivet ustem vel eum iniure dolor in hendevit inil vulgutate velt esse molestie consequator veli illum dolore su feuglist nulla facilissia al vero esse si accumsant et susci dolorumes.



## OO: hergebruik van code

Hergebruik kan via

- classes / objecten
- inheritance

Allemaal vaste data-types, met bijbehorend wisselend gedrag

Wat als je nu hetzelfde gedrag wil hergebruiken, maar dan met wisselende datatypes?

Templates!

## Gewone functie

```
int functionname(int parameter) {
    // rest van de functie
}
```

## Template functions

```
template<typename T> T functionname(T parameter) {
   // rest van de functie
                              of
template<typename T>
T functionname (T parameter) {
    // rest van de functie
```



## template-functies met meerdere types

```
template <typename T, typename U>
void doSomething(T first, U second) {
   ...
}
```

maar wat als er wel een returntype is?

# template-functies met meerdere types

```
template <typename T, typename U>
int multiply(T first, U second) {
   return first * second;
}
```

return type is afhankelijk van template types T & U

## template-functies met meerdere types

```
template <typename T, typename U>
T?/U? multiply(T first, U second) {
   return first * second;
}
```

return type is afhankelijk van template types T & U

## template-functie met auto return type

```
template <typename T, typename U>
auto multiply(T first, U second) {
    return first * second;
}
```

return type is afhankelijk van template types T & U

## template-functie met auto return type: decltype

```
template <typename T, typename U>
auto multiply(T first, U second) -> decltype( first*second) {
          return first * second;
}
```

return type is afhankelijk van template types T & U: **decItype** verteld de compiler wat er voor **auto** gebruikt moet worden

## Template classes

Zelfde idee als voor een template functie, maar nu voor een hele class

Veel gebruikt bij container classes: vector<string>

```
template <typename T>
class MyClass {
    T someMethod(const T& someType);
}
```

# Voorbeeld: ContainerShip



## Template gotcha's

voor iedere template instantie, nieuwe class
 meer code om te builden

- template definiëren in *header* file

veel templates: langere compile

warrige foutmeldingen van compiler

VStudio: I'm looking at you...:-(

# Opdrachten volgende keer (1)

- Maak één template-functie die array's kan sorteren:
  - Een array van strings
  - Een array van floats

- Maak een template-class die Queue-semantics implementeert
  - put() functie: element toevoegen aan einde
  - get() functie: element ophalen van 't begin
  - size() functie: aantal elementen

print tussendoor de queue om te testen

# Opdrachten volgende keer (2)

Voor een spel modelleer je een serie classes die twee soorten non-playing characters gaan implementeren: elves en orcs. Beide classes hebben 1 attribute: name, en 1 methode: render().

Beide typen NPC zijn er in 3 rollen: Farmer, Soldier, Shaman.

In plaats van 6 subclasses te maken, implementeer je deze varianten via het Decorator pattern. De render () methode print de naam, en de rol.