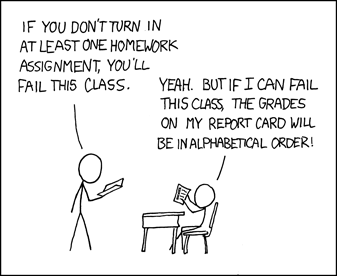
# Practicumopgaven week 1



Bestudeer de hoofstukken van de reader zoals aangeven in het weekoverzicht.

## Installeer

Run het Linux image onder VMware. Maak in de v1oopc-examples directory een kopie van de subdirectory van een vergelijkbaar voorbeeld om de een opgave in te maken. Run de update\*.bat file om je nieuwe directory zichtbaar te maken in CodeLite.

## Boeken

Onderstaant zie je de code van de class *book*. De code is verdeelt over een .hpp bestand en een .cpp bestand. Lees de code en beantwoord de volgende vragen:

1. Wat doet de contructor van de klasse book?
2. Hoe noemen we de tekst tussen de : (dubbelepunt) en de { (krulhaak) bij de constructor?
3. Welke functies van de klasse book zijn van buitenaf toegankelijk?
4. Waarom denk je de schrijver van deze code er voor heeft gekozen sommige functies private te maken?
5. Waarom staat er in de .cpp voor elke functie “book::”?

|  |
| --- |
| #ifndef BOOK\_HPP  #define BOOK\_HPP  #include <ostream>  class book {  private:  string text;  string author;  string title;  void print\_text();  void print\_author();  void print\_title();  public:  book(  const std::string & text,  const std::string & author,  const std::string & title  ):  text( text ),  author( author ),  title( title )  {}    void print();  };  #endif // BOOK\_HPP |
| book.hpp |

|  |
| --- |
| #include <ostream>  #include "book.hpp"  void book::print(){  print\_title();  print\_author();  print\_text();  }  void book::print\_text(){  cout << text << "\n";  }  void book::print\_title(){  cout << title << "\n";  }  void book::print\_author(){  cout << author << "\n";  } |
| book.cpp |

## Main

Maak een project aan met de book.cpp en book.hpp, en een main.cpp waarin de main functie een book aanmaakt en deze print.

## Meubels

1. Maak een project aan met de klassen: *stool*, *table* en *furniture*. *Stool* heeft als member variabelen n\_legs en n\_seats. *Table* heeft als member variabelen n\_legs, length en width. De constructors krijgen de aantallen mee en slaan die op. De membervariabelen zijn allemaal private.
2. *Furniture* bestaat uit 4 variabelen van het type *stool* en 1 variabele van het type *table*. De constructor van *Furniture* initializeert de meubelen met een redelijke waarden. De membervariabelen van *Furiniture* zijn ook private.
3. *Furniture* heeft de methode *makeMoreHipster*(). Deze functie haalt van alle 4 de stoelen één poot af en geeft de tafel deze 4 poten. Schrijf deze functie en de benodigde functie’s bij *stool* en *table.*
4. Schrijf de main functie. Deze maakt een *Furniture* aan, print zijn eigenschappen, roept zijn *makeMoreHipster*() aan, en print weer de eigenschappen. Om dit te kunnen doen moet je print functies toevoegen aan alle klassen.
5. Teken een UML-diagram met *stool*, *table* en *furniture*.

## Gevulde rechthoek

|  |
| --- |
| D:\2017-2018\vakken\V1OOPC\images\h1\filled-rectangle.png |
| UML diagram van een opgevulde rechthoek (en de window waarop hij wordt afgebeeld) |

Neem de 04-07-window voorbeeldcode als uitgangspunt. Schrijf (voeg er aan toe) een klasse (hpp en cpp files) voor een opgevulde rechthoek. De constructor ziet er hetzelfde uit als van de rechthoek in de voorbeelden maar bij het afdrukken moeten alle pixels die deel uitmaken van de rechthoek (inclusief de randen) getekend worden. Je maakt dus geen gebruik van de lijn klasse, gebruik direct de methode(n) van window. Demonstreer je klasse in een applicatie die een aantal gevulde rechthoeken toont. In het UML diagram zijn de attributen van de gevulde rechthoek weggelaten: die moet je zelf bedenken.

## Samengesteld object

Gebruik de code van de vorige opgave als uitgangspunt. Maak een klasse voor een grafisch object (cpp en hpp files) dat (ten minste)

* 1 rechthoek (de bestaande of de in de vorige opgave gemaakte gevulde rechthoek),
* 1 lijn en
* 1 circkel bevat (dus een compositie)
* en van ten minste 1 van deze basis objecten meer dan 1.

Verdere vereisten:

* Het grafische object moet iets herkenbaars zijn, bv. een huis, trein, wolkenlucht, fabriek, pacman.
* De constructor van jouw object moet (ten minste) 1 parameter hebben die iets aan het object aanpast, bv voor een trein het aantal wagons.
* Maak ook een UML diagram met jouw klasse, de basis klassen die jouw object gebruikt en de window klasse.
* Demonstreer jouw klasse in een applicatie die er (ten minste) twee verschillende varianten van afbeeldt.