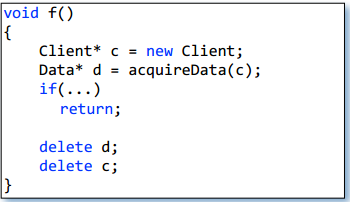
# Disposition06 - Ressource Handling

## RAII – What and why?

RAII er et idiom det står for Resource Acquisition Is Initialization, det går ud på at udnytte at en klasse der bor på stakken bliver nedlagt når den går ud af scope.

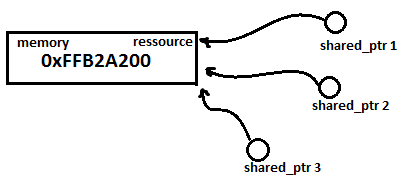
Det gøres ved at ressource bliver allokeret med klassens construktor og derefter bliver nedlagt med destructoren. Vi pakker vores ressource ind og slipper derved for at tænke på deallokering osv.



## Copy construction and the assignment operator

Hvis vi burger RAII er vi nød til at tænke over om det giver mening at kopiere den klasse?

Dertil kan vi som oftest nok svare nej! Med mindre det er noget som en smartpointer



## What is the concept behind a countet SmartPointer?

Ideen med en countet smart pointer er at holde styre på hvor mange der peger på et givent stykke data.

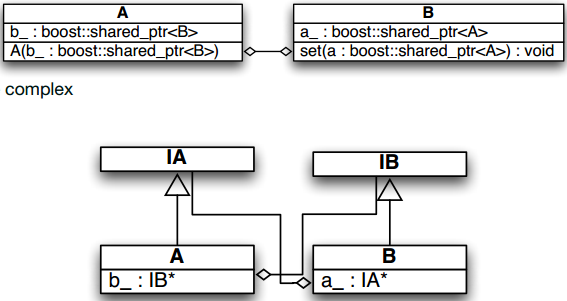
Dette gøres således at vi når der ikke er flere der peger på data, kan vi nedlægge data. Dette gøres ved at have en intern counter der tællers ned når et objekt nedlægges igen. Hvis der når vi nedlægger et objekt er talt ned så det indikeres at der ikke er flere der peger på dataen slettes den.



Det ses i koden at counter er en pointer der bliver kopiere med når objektet bliver kopieret eller assigened. Derved er det den samme der tælles ned og således kan vi holde styr på hvor mange der er tilbage.

## What is boost::shared\_ptr<> and how do you use it?

Det er en countet smartpointer der er lidt smartere, den bruges ligesom en template klasse bruges og den kan derefter holde øje med hvor mange der har fat I det data den peger på. Den har noget tamtam der gør at den kan bryde cykliske afhængigheder.



Dette problem kan løses med boost biblioteket således:

