# ISU: 6. Resource Handling

Håndtering af ressourcer

* Ressourcer er alt det er begrænset – CPU, båndbredde men i denne sammenhængen hovedsagtligt **memory**
* Det bliver et problem når ansvaret for at deallokere hukommelse er uklart

## RAII - What and why?

Resource acquisition is initialization

* Går ud på at indpakke alle ressourcer i deres eget objekt der håndterer deres levetid, og putter objektet på stacken
* For at garantere deallokering af ressourcer når de er out of scope eller ved exceptions
* Subscriber skal også pakkes ind!!
* For ikke at tage stilling til om noget skal nedlægges

## Copy construction and the assignment operator

* Copy constructor – A a = A(B) – ved nyt objekt - Object::Object(const Object&)
* Assignment operator ­– b = a – ved objekt der allerede er lavet
* Disse skal være private, således man ikke kan compile hvis man prøver at tilgå dem
  + På den måde undgår man at flere objekter har pointere til det samme objekt
  + Dette vil man undgå, da man så ikke ved hvem der ejer det originale objekt

## What is the concept behind a Counted SmartPointer ?

* Det er ”dyrt” at kopiere SmartPointer objekter, da det involverer allokering af hukommelse og copy construction
* Ideen med counted smartpointer er så at bruge reference og pointere – men så er problemet, hvem skal så rydde op?
* Med Counted SmartPointer er deletion sikret når reference counteren bliver 0 – altså når den ikke bruges af nogen

Ved brug af counted smart pointer, skal copy constructor og assignment operator implementeres således at

* Copy: kopierer en klasse og tæller counteren op
* Assignment: skal dekrementere counteren, eftersom der nu peges på noget nyt – den nye counter tælles så 1 op. Hvis den originale når 0 skal det selvfølgelig slettes
* Objektet selv der tager stilling til om det skal nedlægges
  + Shared pointer som en attribut 🡪 bruges til at pege på sig selv

## What is boost :: shared ptr and how do you use it?

* boost::shared\_ptr er et eksempel på en counted smart pointer
* Delete er garanteret når den sidste reference til objektet ”dør” eller når member function reset() bliver kaldt

**Best practice:**

Godt:

* shared\_ptr<int> p(new int(2));
* f(p, g());

Skidt:

* f(shared\_ptr<int>(new int(2)), g());
* Her er det skidt, da parametrene ikke bliver kaldt i nogen bestemt rækkefølge – hvis g er først og smider en exception – then what?