C++ Fokus på sproget #07

in C++ Fokus with 0 Comments

Class & object – klasse & objekt

Aldrig havde jeg set så mange ledninger i en bil, som da jeg første gang åbnede motorhjelmen på min Auris Hybrid, så jeg skyndte mig at lukke igen. Synet efterlod ingen tvivl om, at der skal rigtig meget kontrol og styring til for at sikre, at man får det optimale ud af sådan et køretøj. Inde i kabinen er tingene derimod ganske normale. Rat, speeder, bremse og en række kontrollys og instrumenter, hvor jeg kan aflæse tingenes tilstand. Selv om bilen er overordentlig kompliceret, er den ikke sværere at køre end en almindelig bil. For dens interface, det som jeg har adgang til, er ganske simpelt. Det er så at sige bag ved facaden, at der sker en hel masse, som jeg ikke har direkte adgang til.

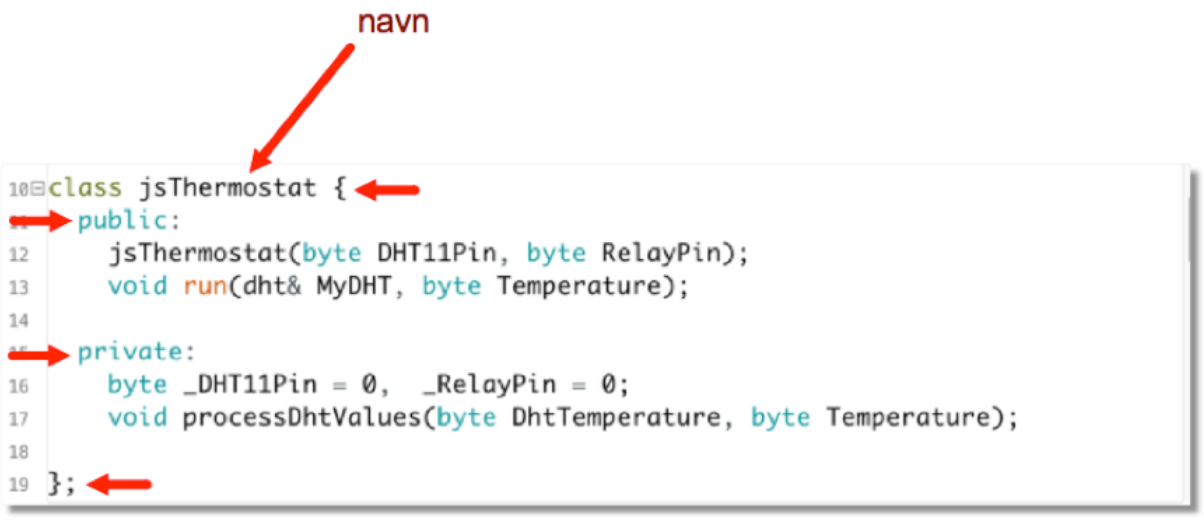
Det minder på mange måder om det, man kalder for et objekt i C++. Objekter er en samling elementer, blandt andet funktioner, der tilsammen kan udføre komplekse opgaver. Objekter har som min bil et interface, som kan tilgås udefra og så har de en helt lukket del, som objektet selv kan sætte i arbejde, men udefra kan man ikke få adgang til den del.

Nøjagtig som min bil. Jeg har mulighed for at træde mere eller mindre på speederen. Men om elmotoren bliver koblet ind eller ud, om der bliver ladet på batteriet eller ej har jeg ingen adgang til. Det afgør systemet ud fra en analyse af fart og vej.

Når man designer sine objekter, tager man stilling til, hvad de skal gøre totalt set. Hvad er det for en opgave, man ønsker at løse? Man tager også stilling til, hvor meget der skal være tilgængeligt direkte og hvor meget der skal ligge under motorhjelmen. Det erklæres med henholdsvis public: og private:

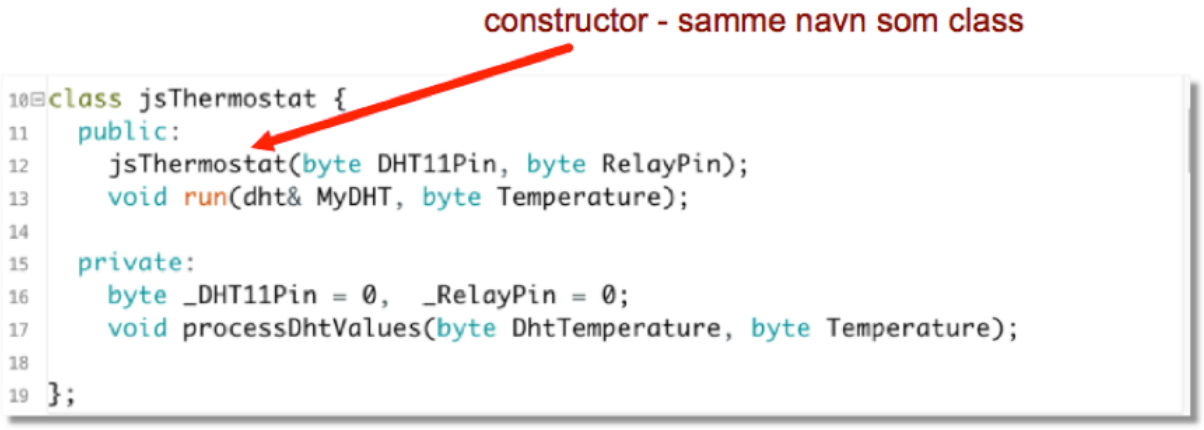
Et sådant design for et objekt kaldes for en klasse, class. Man kan tænke på class som et blueprint eller en byggebeskrivelse. En klasse skal have et navn og som alle andre navne skal navnet være informativt og meningsfuldt i sammenhængen. Navne på klasser skrives i CamelCase som variabelnavne men foran navnet er det god skik at sætte et lille prefix. Jeg plejer at sætte mine initialer, js, ind som prefix. Som al anden kode i C++ er der helt faste krav til syntaksen. Det er lettest at demonstrere ved at kigge på et konkret eksempel.

Den første del af klasse er en definition af, hvilke elementer der er henholdsvis public og private. I eksemplet ser denne del således ud:



I denne del skriver man klassens navn og viser med ordene public og private, hvad der hører til hvor. Det hele kapsles ind af to krøllede parenteser.

I public-delen skal man vise, hvad der skal gøres, når objektet bliver bygget. Det kalder man for en constructor og den skal have samme navn som klassen. Den behøver ikke at have parametre, men den må gerne. Den har INGEN type, – heller ikke void.



Dernæst viser man, hvilke funktioner, der må kaldes “udefra” og man viser, hvorledes de skal erklæres, – dvs. hvilken type de er og hvilke parametre, de i givet fald er udstyret med.

Når man har defineret sin klasse, dvs. vist hvilke metoder og variable som er public: og hvilke der er private: skal man for hver funktion skrive den kode, som skal udføres, når funktionen kaldes. Det kaldes for implementeringsdelen. Implementation Når det er gjort, er klassen klar til at fungere som blueprints for de objekter man opretter.