# C++ Fokus på sproget #09

in C++ Fokus with 0 Comments

## Klasser og objekter i praksis – THE CLASS

Nu skal sketchen skrives om, således at så meget som muligt bliver lagt ind i en klasse. For god ordens skyld skal det nævnes, at man normalt ville konstruere sin klasse med det samme. Det er kun for at tydeliggøre forskellen mellem at arbejde med klasser/objekter og enkeltstående funktioner, at dette projekt er grebet an som i disse posts.

Allerførst skal klassen have et navn og da jeg allerede ved, at den skal laves til et library, kommer den til at have prefixet js. Klassen kunne derfor hedde jsControlDHT. Det vil være oplagt, at få overført oplysninger om, hvilke pins der skal bruges til dens constructor, som derfor skal udstyres med parametre til at tage imod disse værdier. Ud over det er der kun brug for at kunne tilgå en enkelt metode, som så at sige kan holde tingene i gang. Resten af koden kan derfor gemmes af vejen i klassens Private del. Derfor kommer Public delen kun til at indeholde få elementer og ser således ud:

```
13 class jsControlDHT {
14   public:
15     jsControlDHT(byte DHT11Pin, byte OnesPin,
16                  byte TenthsPin, byte ActivateTrendPin,
17                  byte RunPin);
18     void run(dht& MyDht);
19
```

Bemærk at Constructor hedder det samme som klassen. I erklæringen af void run optræder der en lidt speciel parameter, idet der står (dht& MyDht). Sådan lidt frit oversat betyder det, at parameteren er et objekt af klassen dht. Det hedder MyDht, men er ikke erklæret endnu. Derfor får compileren her besked på, at alle informationer vil dukke op på et senere tidspunkt, så der er ingen grund til at komme med fejlmeddelelser. Det er nødvendigt at gøre det sådan, for objektet MyDht kan først instantieres EFTER erklæringen af vores klasse.

Klassen skal have nogle variable for at kunne lagre de værdier, der skal bruges undervejs. I denne klasse findes f. eks. variable ved navn _DHT11Pin, _OnesPin, _TenthsPin, _ActivateTrendPin og _RunPin. De ligger i Private delen og kan derfor ikke ses udefra, men kan benyttes af alle funktioner inden for klassen. Deres scope er med andre ord selve klassen og det mindsker behovet for globale variable i betydelig grad. Det samme gælder for de øvrige variable, der var globale variable i den rene “funktionsudgave” af sketchen. De øvrige funktioner kan også placeres i Private delen, da vi ikke har brug for at kunne tilgå dem udefra. Definitonen af klassen kommer derfor til at se således ud i sin helhed:

```
13 class jsControlDHT {
14   public:
15     jsControlDHT(byte DHT11Pin, byte OnesPin,
16                  byte TenthsPin, byte ActivateTrendPin,
17                  byte RunPin);
18     void run(dht& MyDht);
19
20   private:
21     byte _DHT11Pin = 0,  _OnesPin = 0;
22     byte _TenthsPin = 0,  _ActivateTrendPin = 0;
23     byte _RunPin = 0;
24
25     byte _StoredTemperature = 0, _StoredHumidity = 0;
26     boolean _IsRunning = true, _ShowTrend = true;
27
28     void makeAlert(unsigned int mseconds);
29     void doShowTrend(byte CurrentValue, byte &OldValue);
30     void showDhtValues(byte Value);
31     void processDhtValues(byte Temperature, byte Humidity);
32 };
```

For at vores Class kan gøres færdig, skal Implementation delen skrives. Den ser således ud i delvis foldet form:

```
33
34 // * * IMPLEMENTATION * * *
35
36 jsControlDHT::jsControlDHT(byte DHT11Pin,
37                            byte OnesPin, byte TenthsPin,
38                            byte ActivateTrendPin, byte RunPin) {
39   _DHT11Pin = DHT11Pin;
40   _OnesPin = OnesPin;
41   _TenthsPin = TenthsPin ;
42   _ActivateTrendPin = ActivateTrendPin ;
43   _RunPin = RunPin ;
44   pinMode(_OnesPin, OUTPUT);
45   pinMode(_TenthsPin, OUTPUT);
46   pinMode(_ActivateTrendPin, INPUT_PULLUP);
47   pinMode(_RunPin, INPUT_PULLUP);
48 }
49 void jsControlDHT::run(dht& MyDHT) {
57
58 void jsControlDHT::makeAlert(unsigned int mseconds) {
71
72 void jsControlDHT::doShowTrend(byte CurrentValue, byte & OldValue) {
87
88 void jsControlDHT::showDhtValues(byte Value) {
114
115 void jsControlDHT::processDhtValues(byte Temperature, byte Humidity) {
127 /// end of class jsControlDHT
```