Hvad har jeg lært i vores første uge af H2.

Af Nikolaj Hoggins Jacobsen

Det er nu ved at være længe siden at vi var på H1, hvor vi lærte grund niveauet, såsom datatyper og loops. Siden da har jeg haft travlt på arbejde og lært en masse ting inden for programmering. Så det var spændende at se hvilke ting jeg ville have nemt ved, og hvilke ting der ville være svære når vi skulle i gang.

Det viste sig at alting gav ret god mening, men der stadig var en masse nye ting at lære.

//Libraries

Vi startede ugen ud med at få en velkomst tilbage i C# og skulle lære hvad alt var igen. Efter at vi havde fået styr på det grundlæggende blev vi introduceret til libraries, og blev sat til at lave vores egen. Libraries giver super god mening, da det gør din kode modulær, og funktionerne du bygger ind i dit library, kan, og vil, ofte være noget der kan bruges i flere programmer, hvor du kan hive dit library ind.

//Dictionaries

Dictionaries er en smart form for liste i c#. Den bedste måde jeg, med baggrund i ikke-type strict scripting languages, kan forklare hvad dictionaries er, er ved at forklare dem som en array med key value pairs. Dictionaries er smarte til at f.eks. have en liste objekter som value, og nemt kunne søge efter dem med f.eks en streng som key. Du kunne også have id's i key, som en nem måde at holde styr på brugere.
Dictionaries bliver initialized således:

```
Dictionary<string, Person> people = new Dictionary<string, Person>();
```

Det man så kan gøre for at finde en person i sin dictionary er bare at skrive

```
people["Nikolaj Hoggins"]
```

//Overloading

Overloading er noget jeg har set til en del før, men aldrig rigtigt har dykket ned i. Det bliver brugt en del i spil programmering, og er noget jeg er løbet ind i, når jeg har lavet spil i Unity og Unreal.
For at bruge overloading skal man sådan set bare lave to metoder med samme navn, og forskellige parametre. Således.

```
public static int Add(int a, int b)
{
    return a + b;
}
public static float Add(float a, float b)
{
    return a + b;
}
```

Det gør at man kan kalde den funktionen man ønsker ved at indsætte de parametre der er tilsvarende den ønskede funktion.

```
MathClass.Add(2, 2);
MathClass.Add(2.0f, 2.0f);
```

Det var fedt at lære at lave overloading efter kun at have brugt dem tidligere, og sådan set meget ligetil at bruge.

//UML Class diagrams

UML Class diagrams er en super smart måde at planlægge sit program på inden man faktisk begynder at kode.
Jeg er en person der rigtigt godt kan lide bare at gå i gang med at kode, og ofte har tankegangen at “det kan jeg sagtens like lave”, denne mentalitet har dog ofte bidt mig i røven, når jeg rammer en blokering i mit “flow” og lige pludseligt ikke rigtigt kan tænke mig til hvad jeg mangler eller hvor jeg har lavet noget forkert.

Det har så lidt efterhånden lært mig at det er smart at lave et UML Class diagram, inden man begynder at kode. Dog har jeg lagt mærke til at man sjældent slipper for at have glemt nogle ting, men det giver en bedre overblik over hvad man har glemt, hvor man så kan tilføje dem til en ny version af diagrammet.

//Hjemmearbejde

Vi havde en dag med hjemmearbejde, hvor vi fik en opgave vi skulle løse, med nogle ret løse termer. Hvilket jeg synes var en super fed opgave.
Jeg gik ind i dagen tænkende at den opgave havde jeg hurtigt klaret, men da jeg begyndte at tegne UML Class diagrammet, kunne jeg hurtigt se, at der var mange ting jeg gerne ville tilføje udover opgave, og jeg kunne dermed udvide tidshorisonten. Jeg endte dog uanset hvad med at have travlt da dagen var ved at være omme. Og netop af den grund, synes jeg det var en super god opgave, der bredte sig lidt efter evne.

//Delegates

Delegates synes jeg var lidt specielle. Hvor at i JavaScript, kan du bare få lov til at smide funktioner i variabler, som du har lyst til, skal du i C# lave en delegate som du kan gemme dine funktioner i. Du kan lave en delegate således:

```
delegate void VoidDelegate();
```

du kan derefter sætte den til en funktion:

```
VoidDelegate del = MyFunction;
```

Eller bruge en lambda funktion

```
VoidDelegate del = () => Console.WriteLine("Ran delegate");
```

//Lambda

Lambda funktioner er en fed, anderledes måde at skrive funktioner på.
Som får funktioner vi kender sådan her:

```
void TestFunc(){
    Console.WriteLine("Testing");
}
```

Til at se sådan her ud

```
void TestFunc = () => Console.WriteLine("Testing");
```

Og kan bruges super godt i sammenhæng med Delegates. Da jeg har kodet en del JavaScript og React specifikt, kender jeg ret godt til lambda funktioner, og synes ret så godt om dem.