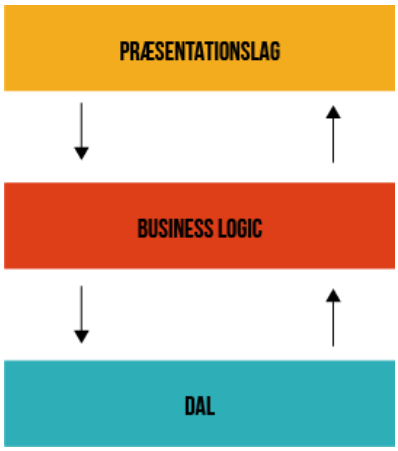
Marker som gennemført



Der er to regler som er gældende her:

1. High level moduler må ikke være afhængig af low level moduler

2. Abstraktioner må ikke være afhængig af detaljer

Inden vi kan gå videre skal vi have styr på definitionerne:

Low level betyder i princippet klasser som implementerer basis og primære operationer som f.eks. skrivning til disk, netværkskommunikation m.m. Husk 3-lags modellen. Faktisk er low-level det der svarer til DAL-laget. High level er så alle de klasser som indkapsler forretningslogikken.

En naturlig måde at kode på, er først at implementere low level klassen og herefter forretningslogikken, her skal man være opmærksom på at man hurtigt kan komme til at lave unødige koblinger og at high level klassen bliver afhængig af low level klassen, derfor inversion!

Kik på eksemplet her

```
class Worker {
  public void work() {
      // ....working
  }
}

class Manager {
  Worker worker;

  public void setWorker(Worker w) {
      worker = w;
  }

  public void manage() {
      worker.work();
  }
}
```

Managerklassen er en high-level klasse og Worker klassen betragtes som en Low-level klasse. Som du kan se er Managerklassen afhængig af Worker klassen og det må den ikke være, hvis vi skal overholde dette princip.

Og lad os lige forstille os vi havde brug for at introducere en ny super klasse af worker..... så ville vi have behov for at ændre på Manager klassen.

En god løsning

?

```
interface IWorker {
        public void work();
}

class Worker implements IWorker{
        public void work() {
                // ....working
        }
}

class SuperWorker  implements IWorker{
        public void work() {
                //.... working much more
        }
}

class Manager {
        IWorker worker;

        public void setWorker(IWorker w) {
                worker = w;
        }

        public void manage() {
                worker.work();
        }
}
```

Nu bliver woker i steded til en IWorker og vi kan altid implementere en ny subtype.

Tips med tips : Programmer alktid mod et interface og ikke en implementering.

Senest ændret: torsdag den 1. februar 2018, 09:17

◄ Hjælp til Lone

Spring til...

Dependency injection ►