5. GoF Factory Method/Abstract Factory

1 Redegør for, hvad et software design pattern er

Et design pattern er kernen i en løsning til et problem, som opstår gentagende gange. Årsagen til, at det er et mønster, er, at problemet og løsningen af problemet kan anvendes igen og igen uden, at man gør det samme 2 gange.
Da design patterns altid skal tilpasses vores problem i vores kontekst. Et design pattern er derfor aldrig en færdig løsning på et problem, men en skabelon til løsning af et problem.

2 Redegør for opbygningen af GoF Factory Method og Gof Abstract Factory

2.1 Factory Method pattern

Formålet med Factory method er at adskille objekt oprettelsen fra kontrollen. Dette gør den ved at enkapsulere objekt oprettelse ved at lade Abstract Creator have en Factory Method som subklasser nedarver. Det er så op til subklasserne at implementere factory Method. Factory bruger således arv til at skabe et nyt obejkt ved at subklasser implementerer FactoryMethod

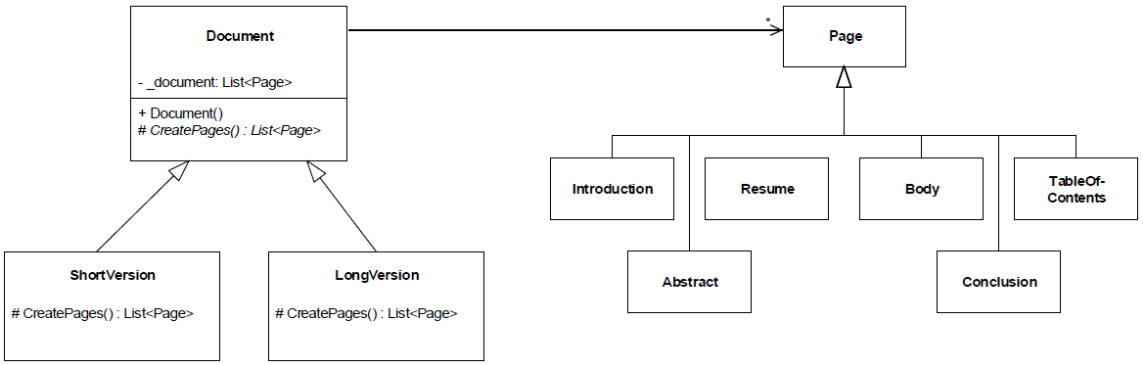
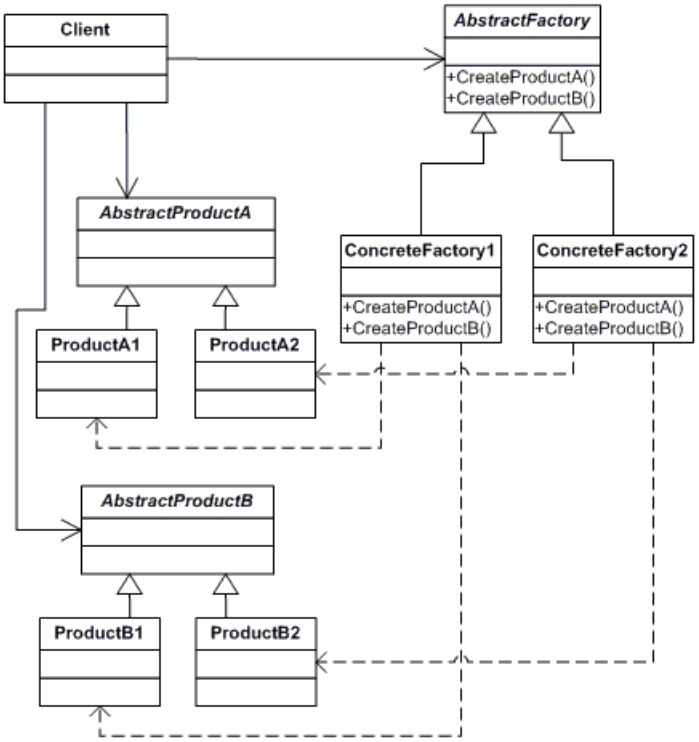


Figure 1: Factory Method

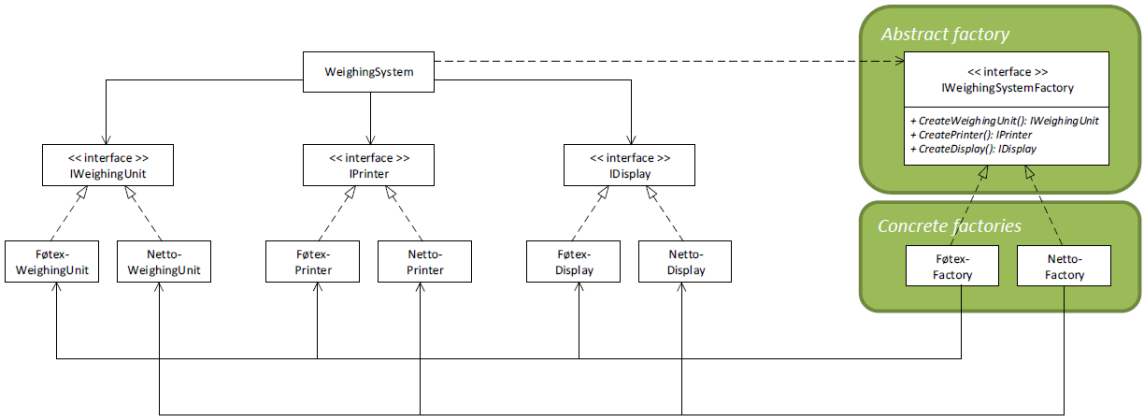
2.2 Abstract factory pattern

Abstract factory pattern er et creational pattern, som står for oprettelsen af objekter. Det særlige ved dette pattern er at alt er indkapslede og det færdige objecte fra factory'en bliver en del af strategy pattern.
Formålet med abstract factory pattern er at lave et interface til oprettelse af familiære objecter UDEN at specificere deres konkrete klasser. Vi vil gerne separere oprettelsen af et objekt fra brugen af objektet. I stedet for en klasse selv skal kende til hvilke afhængigheder den skal oprette, kan den få en abstract factory med i dens constructor, som så skaber en sammensætning af afhængigheder.



3 Giv et eksempel på anvendelsen af GoF Abstract Factory

Et eksempel på Abstract factory pattern er stocking control device.



```
public class Application
{
  public static void Main()
  {
    // Create a Føtex weight
    var føtexWs = new WeighingSystem(new FøtexFactory());
  }
}
```

```
public class FøtexFactory : IWeighingSystemFactory
{
    public IWeighingUnit CreateWeighingUnit() {
        return new FøtexWeighingUnit();
    }
    public IDisplay CreateDisplay() {
        return new FøtexDisplay();
    }
    public IPrinter CreatePrinter() {
        return new FøtexPrinter();
    }
}
```

```
public class WeighingSystem
{
    private IWeighingUnit _weighingUnit;
    private IPrinter _printer;
    private IDisplay _display;

    public WeighingSystem(IWeighingSystemFactory factory)
    {
        _weighingUnit = factory.CreateWeighingUnit();
        _printer = factory.CreatePrinter();
        _display = factory.CreateDisplay();
    }
}
```

Figure 2: Abstract Factory Code