## Factory Design Pattern

Het **Factory Design Pattern** is een veelgebruikt creational design pattern dat verantwoordelijk is voor het creëren van objecten zonder dat de code expliciet hoeft te weten welk specifiek object moet worden gemaakt. Het hoofddoel is om de afhankelijkheid van concrete klassen te verminderen en om flexibeler te zijn wanneer objecten worden gemaakt.

**Concept van het Factory Pattern**

Bij het Factory Pattern wordt een aparte klasse gemaakt, de "Factory", die verantwoordelijk is voor het maken van objecten van verschillende typen. De clientcode roept de Factory aan, en de Factory bepaalt welk object moet worden gemaakt op basis van de input of parameters.

**Voorbeeld**

public class WeerstandFactory

{

public enum WeerstandType

{

EnkelVoudig, SerieSchakeling, ParallelSchakeling

}

static IWeerstand MaakWeerstand(WeerstandType weerstandstype)

{

}

}

**Opdracht: Wat is een ‘design pattern’? Ken je nog andere ‘design patterns’.**

**Opdracht: Toon aan dat elk electrisch schema opgebouwd is uit een combinatie van weerstanden, parallelschakelingen en serieschakelingen.**

**Opdracht: Implementeer drie klasses ‘EnkelVoudigeWeerstand’, ‘ParallelSchakeling’, en ‘SerieSchakeling’ volgens de interface IWeerstand.**

**Opdracht: Vul de klasses aan met de nodige properties**

* Een enkelvoudige weerstand heeft een weerstandswaarde (in Ohm)
* Een serie- en parallelschakeling heeft is gebouwd uit twee andere weerstanden (of schakelingen).

**Opdracht: Implementeer de functies die de:**

* Equivalente weerstand van het circuit bepalen (RECURSIE!)
* De spanning / stroom berekenen op basis van deze equivalente weerstand.

**Opdracht: Maak een object aan die een schakeling voorstelt en bereken de equivalente weerstand, test jouw klasses uit.**