# Design patronen in C#: Abstract factory

Het design patroon dat hier besprekon wordt, is de ‘abstract factory’.

Een ‘abstract factory’ is een creatie patroon waarmee je groepen van gerelateerde objecten kunt maken zonder concreet de klas op te geven. Dit patroon wordt ingezet bij een volgend probleem:

Stel, je maakt een applicatie voor een autohandelaar welke verschillende typen auto’s aanbiedt: personenauto, vrachtauto en bussen. Binnen deze categorieën bestaan ook weer varianten: je kunt bijvoorbeeld een elektrische variant krijgen van de auto, vrachtauto en bus of een extra snelle sport uitvoering. In de toekomst wil de autohandelaar nog meer varianten gaan verkopen, dus de code moet makkelijk aanpasbaar zijn. Het moet niet zo zijn dat als er een variant bijkomt, dit betekent dat de hele applicatie herschreven moet worden.

Hier is een ‘abstract factory’ een goeie oplossing. Je maakt per hoofd categorie een interface aan met daarin de gedeelde eigenschappen die elke variant heeft, bijvoorbeeld;

1. HasWheels()
2. HasEngine()

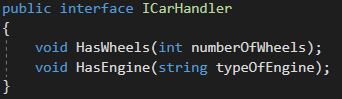
Elke afleidende variant moet deze eigenschappen overnemen, maar kan hier extra eigenschappen toevoegen, zoals extra zitplaatsen als het gaat om een familie auto bijvoorbeeld.

De volgende stap is het bouwen van de ‘factory’, welke de gevraagde klas initialiseren en teruggeeft. Deze fabriek geeft een klasse terug van elke categorie, dus een auto, een vrachtauto en een bus.

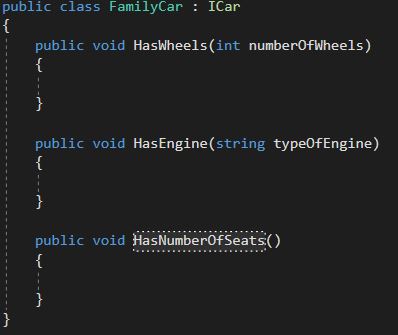
Voor elke variant die weer binnen deze categorieën valt, maak je een aparte fabriek aan. Dus je hebt een fabriek voor de elektrische variant, welke van alle categorieën een elektrische klasse teruggeeft.

## **Voorbeeld in code**

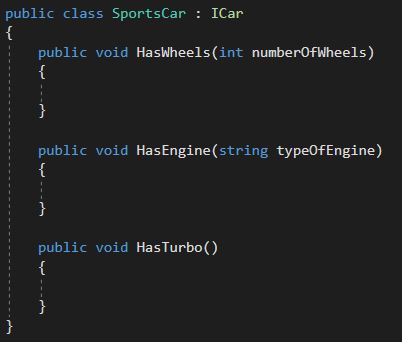
We hebben allereerst de interface voor het hoofdelement, de auto. Deze heeft twee functies, HasWheels en HasEngine:



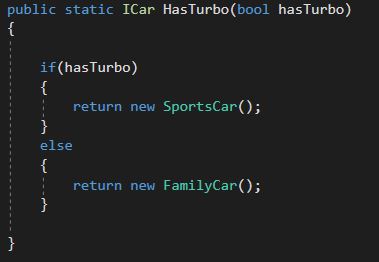
Daarnaast hebben we de respectievelijke modellen voor zowel de familie auto:



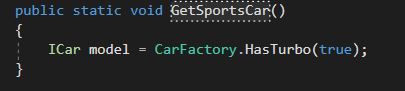
als de sport auto:



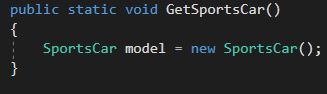
Hiernaast hebben we de eigenlijke factory die de gevaagde class initialiseren en teruggeeft op basis van de parameters:



Als we nu een nieuwe sport auto class willen hebben dan roepen we deze op de volgende manier aan:



Je ziet dat je niet meer expliciet opgeeft welke klasse je aanmaakt maar dat de factory dat beoordeelt op basis van de parameter die je meegeeft. Als je geen gebruik maakt van een factory zou je op de volgende manier een nieuwe klasse initialiseren:



Mochten er nu varianten worden toegevoegd, dan hoeft niet de klasse Car aangepast te worden met de nieuwe variant, maar kun je simpel een nieuw model maken die afstamt van ICar en de factory uitbreiden met de benodigde code.

Een nadeel van dit patroon is dat het enige complexiteit met zicht mee brengt. Er moeten namelijk als je gebruik maakt van dit patroon een hoop interfaces en factories aangemaakt worden.

Voor meer informatie kun je onderstaande links gebruiken:

1. <https://refactoring.guru/design-patterns/abstract-factory/csharp/example>
2. <https://www.dofactory.com/net/abstract-factory-design-pattern>
3. https://nl.qaz.wiki/wiki/Abstract\_factory\_pattern