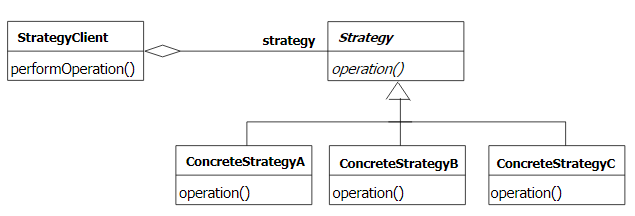
Design patterns

Definities niet per se kennen, enkel de essentie

# Strategy pattern

Familie van algoritmen en maakt ze uitwisselbaar. Mogelijk om algoritme los van de client te gebruiken. Dit ontwerp heeft nog een setter nodig in de StrategyClient, zodanig er indien nodig veranderd kan worden van strategy. Er wordt van **buitenaf** van klasse veranderd.

# Simple Factory pattern

Creëren van klasses

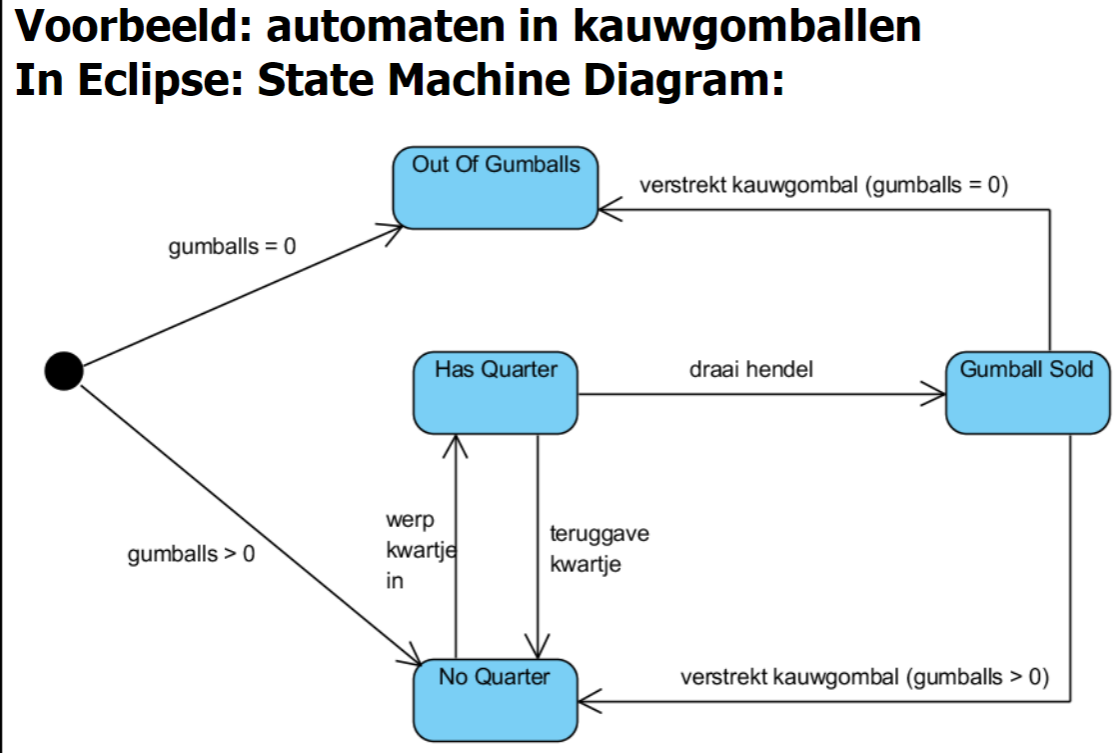
# Observer pattern

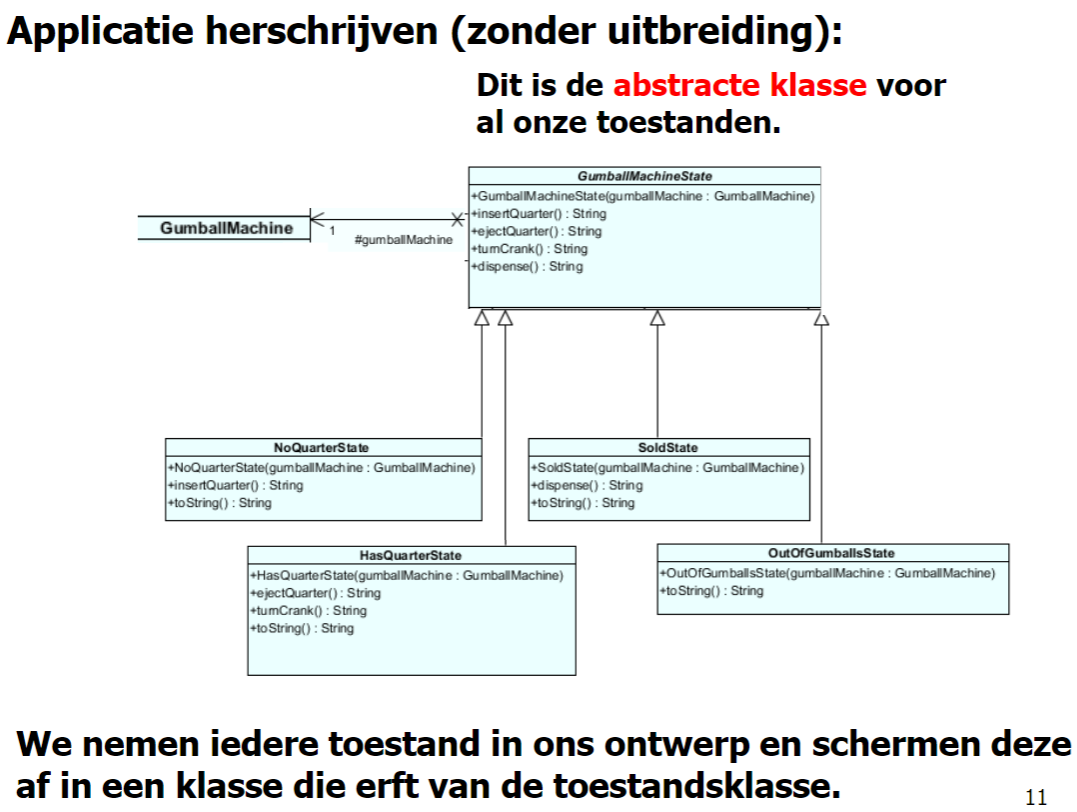
Afbeelding met tekst

Automatisch gegenereerde beschrijving

# State pattern

Eerst toestandsdiagram maken, daarna domain class diagram ! Dit is enkel bij state pattern!

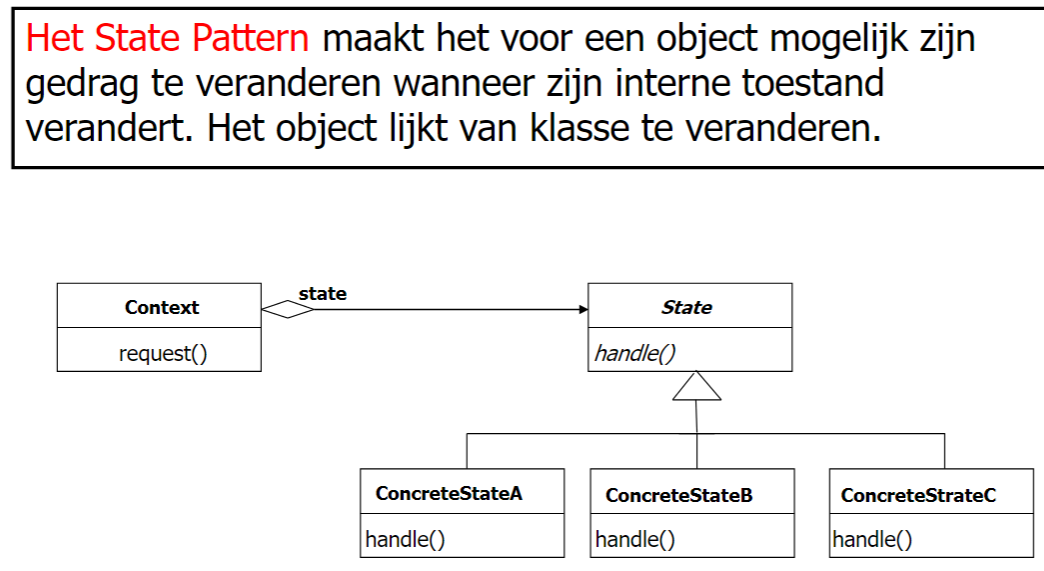




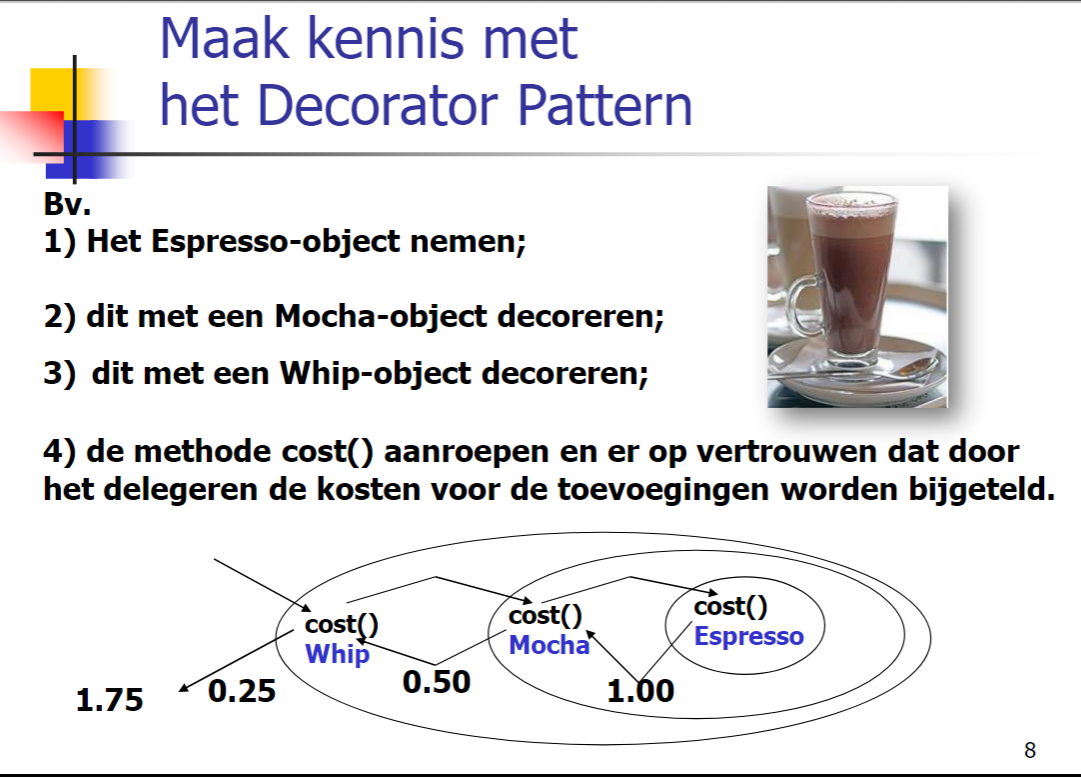
Abstract class om als basis te dienen. Er moet nog altijd de effectieve state (subklassen) gedefinieerd worden.

Klasse gumballmachinestate is beveiligd in package, de “buitenwereld” kan er niet aan.

De klasse beslist **zelf** of hij van state verandert of niet.

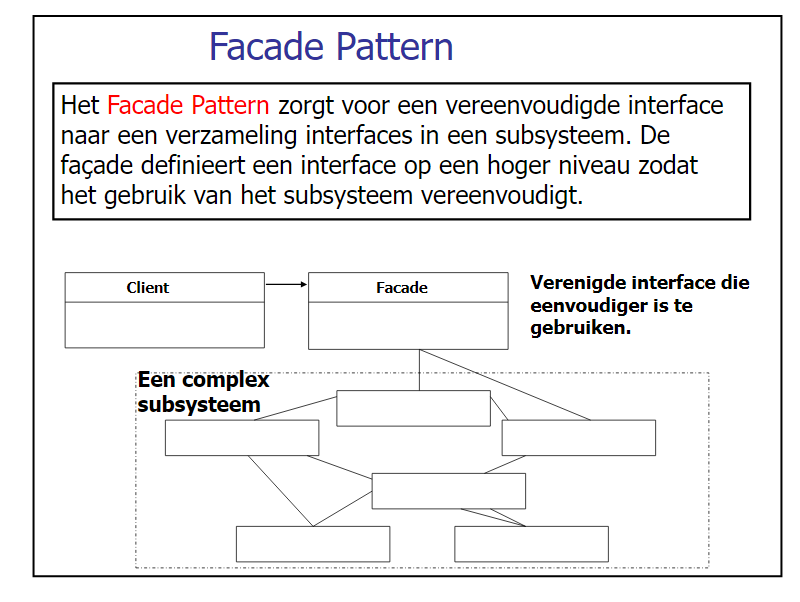


# Decorator pattern



Bestaat uit concrete componenten & constructor die iets verpakt.

# Facade pattern

Wordt niet uitgewerkt

# MVC

Model view controller

Het zijn een paar patterns die zijn samengevoegd.

MVC: uit welke patterns bestaat dat?

* Niet echt DP, zal ook niet echt gevraagd worden om uit te werken

