# Implementatietemplate Pattern

## Naam pattern: Strategy Pattern

## Omschrijving pattern

Geef in 3-5 zinnen aan hoe het pattern **concreet** werkt en welk probleem het pattern oplost.

|  |
| --- |
| Elke kamer kan verschillende soorten opdrachten hebben. Om te voorkomen dat we voor elk type opdracht veel duplicaatcode moeten schrijven, hebben we een interface Opdracht gemaakt. Dit stelt ons in staat om op een strategische manier te werken: we kunnen eenvoudig één van de drie opdrachtklassen kiezen (PuzzelOpdracht, MeerkeuzeOpdracht of OpenOpdracht) zonder extra aanpassingen aan de rest van de code. Dit maakt het systeem flexibeler, herbruikbaar en gemakkelijker uitbreidbaar. |

## Classes

Noem hier de voor het pattern relevante classes in je code en geef per class aan wat zijn verantwoordelijkheid/functie is.

|  |  |
| --- | --- |
| **Class** | **functie** |
| StandaardKamer | Dit is een instantie van de interface Kamer |
| Opdracht | Opdracht is een interface die het gedrag definieert dat elke opdrachtklasse moet implementeren. Het wordt gebruikt om op een strategische manier verschillende soorten opdrachten (zoals OpenOpdracht, MeerkeuzeOpdracht en PuzzelOpdracht) te kunnen inzetten. |
| MeerkeuzeOpdracht | Dit is een instantie van de interface Opdracht. Hiermee kunnen er meerdere mogelijke antwoorden in een lijst worden gezet. |
| OpenOpdracht | Dit is een instantie van de interface Opdracht. Hiermee wordt een vraag en het antwoord meegegeven |
| PuzzelOpdracht | Dit is een instantie van de interface Opdracht. Hiermee kunnen de antwoorden met elkaar worden gematcht |
|  |  |

## Inspanning

Geef aan per student wat zijn bijdrage was in het realiseren van dit pattern

|  |  |
| --- | --- |
| **Naam student** | **Bijdrage** |
| Dami Kastaneer | Interface Opdracht gemaakt met de bijbehorende klassen. |
| Anson Mok | Verschillende vragen gemaakt en geimplementeerd in de code. |
| Nathaniel Rasmijn | Standaard kamers gemaakt en geimplementeerd in de code. |
|  |  |