# Implementatietemplate Pattern

## Naam pattern: Template Method Pattern

## Omschrijving pattern

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

|  |
| --- |
| Door gebruik te maken van een abstracte klasse Kamer kunnen we eenvoudig verschillende soorten kamers implementeren en later uitbreiden en aanpassen naar hun eigen unieke methodes. Dit maakt het systeem flexibel en herbruikbaar. We kunnen meerdere kamertypes definiëren zonder veel extra code te schrijven, omdat de gemeenschappelijke functionaliteit al in de abstracte klasse is vastgelegd. |

## Classes

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

|  |  |
| --- | --- |
| **Class** | **functie** |
| Kamer | Kamer is een abstracte class. Hiermee hebben wij een template van hoe een kamer precies moet werken. Hierdoor kunnen wij eenvoudig veranderingen maken wanneer nodig. Deze class heeft meerdere methods zoals getNummer() die voor elke instantie zo aangepast kan worden. |
| StandaardKamer | Dit is een instantie van de abstracte Kamer class. Hiermee kunnen we een standaard kamer maken met een introtekst en opdracht. |
|  |  |
|  |  |
|  |  |
|  |  |

## Inspanning

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

|  |  |
| --- | --- |
| **Naam student** | **Bijdrage** |
| Dami Kastaneer | Kamer en StandaardKamer in klassendiagram gemaakt een aangegeven. |
| Anson Mok | Abstract Class en StandaardKamer Class gemaakt en geimplementeerd in de code. |
| Nathaniel Rasmijn | Meerdere instanties van de StandaardKamer class gemaakt en geimplementeerd in de code. |
|  |  |