**Technisch Ontwerp**

Klas 1B, Groep 8

{Foto}

{Namen}

**Inhoud**

[**Inleiding** 2](#_Toc146526979)

[**Technieken uitgelicht** 3](#_Toc146526980)

[**OOP** 3](#_Toc146526981)

[**Polymorphism** 4](#_Toc146526982)

[**MVC** 4](#_Toc146526983)

[**ULM Diagrammen** 5](#_Toc146526984)

[**Systeemomschrijving** 5](#_Toc146526985)

[**WPF** 5](#_Toc146526986)

[**Bibliografie** 6](#_Toc146526987)

# **Inleiding**

# **Technieken uitgelicht**

**A group of columns with text

Description automatically generatedA diagram of a car

Description automatically generated****OOP**  
Object-Oriented-Programming, met als afkorting OOP, is een methode van programmeren die voornamelijk gericht is op Classes, Objects, Functions, Methods, Attributes en Properties. Er zijn verschillende programmeertalen die gebruik maken van deze methode, zoals bijvoorbeeld Python, C#, C++ en Java(Script). Classes zijn een verzameling van objects, alle objects in een class hebben dezelfde attributes, maar de attributes hebben in elk object een andere property. Objects hebben properties en functions, een function die aan een object toebehoord heet een method, en hiermee koppel je dus attributes en methods aan een object (Figuur 1).  
  
**De vier pilaren van OOP**  
OOP maakt gebruik van vier globale principes. Deze vier principes zijn; Encapsulation, Abstraction, Inheritance en Polymorphism, maar wat houden deze termen nou precies in?  
  
**Encapsulation**  
Bij encapsulation maak je gebruik van public en private objects. Public objects zijn toegankelijk vanuit andere objects, en private objects zijn niet toegankelijk vanuit andere objects. Dit doe je voor het beveiligen van de geschreven code.  
  
**Abstraction**  
Bij abstraction laat je de gebruiker alleen de belangrijke details zien die nodig zijn om je code te laten werken, terwijl het “niet belangrijke” gedeelte niet zichtbaar is. Dit doe je om de code gestructueerd en overzichtelijk te houden.  
  
**Inheritance**  
Bij inheritance zorg je ervoor dat je geschreven code herbruikt kan worden. Bijvoorbeeld als je een class hebt met diverse features, en je er nog meer features los bij wil doen. Dit voorkomt dus dat je twee keer dezelfde code opschrijft, en in plaats daarvoor dus tijd en ruimte bespaard. Dit principe gebruik je dus net als abstraction voor het overzichtelijk houden van je code.

Figuur 2 (Doherty, 2023)

Figuur 1, de 4 pilaren van OOP (Yadav, 2023)

**Polymorphism**  
Polymorphism is het moeilijkst begrijpbare onderdeel van object-oriented-programming. Polymorphism betekend letterlijk; veel (Poly) vormen (Morph). Polymorphism laat je besluiten welke functie je kan laten lopen terwijl je programma al loopt (Coding, 2020).  
  
Als je deze 4 “Pillars” goed uitwerkt zijn er veel voordelen aan het gebruiken van OOP, enkele voordelen zijn; de code die je schrijft is makkelijk te onderhouden, door middel van abstraction. Een ander voordeel is, door middel van inheritance, dat je code makkelijk te debuggen is. Je hoeft niet meerdere waarden aan te passen om het probleem op te lossen. Door polymorphism heeft je code class-specifiek gedrag, en door encapsulation is je code beveiligd voor gevoelige informatie die er in zou kunnen liggen. Kortom; een class moet je zien als een blauwdruk en je objecten zijn de instances van je class.

**A diagram of a software process

Description automatically generated****MVC**Model-View-Controller (MVC) is een ontwerppatroon dat word gebruikt in de informatica dat het ontwerp van complexe toepassingen opdeelt in drie verschillende categoriën. Door middel van het splitsen van de geschreven code word de code overzichtelijker en is het makkelijk te herbruiken. De drie verschillende categoriën van het MVC-patroon zijn; het datamodel (Model), de datapresentatie (View) en de applicatielogica (controller(brain)). De cyclus (Figuur 3) begint bij de user zelf. De user klikt op de view (UI) op een knop. Nadat de user op de knop heeft gedrukt, registreerd de controller dat er iets is gebeurt en voert hij een event uit. Op dit event reageert het model weer, en haalt de gegevens op die benodigd zijn voor de actie die de knop waar hij op heeft gedrukt. Nadat het model de gegevens vanuit de dataopslag heeft weggeschreven en alle gegevens individueel heeft gelabelt, stuurt het model deze door naar de view, die de gegevens weer presenteert aan de gebruiker. Er is één groot voordeel aan het gebruiken en toepassen van dit model in je code, en deze reden is dat je door dit patroon gemakkelijk nieuwe views kan implementeren zonder dat je het model zelf hoeft aan te passen. (Hernandez, 2021) (Model-view-controller-model, 2022)

Figuur 3 (Model-view-controller-model, 2022)

**A computer screen with white text

Description automatically generated****UML Diagrammen**Unified Modeling Language (UML) is een modeling language, het grote doel van UML is om je het ontwerp van je systeem weer te beelden in een diagram. Je hebt 2 “hoofdklassen” aan diagrammen. Deze zijn statische diagrammen en dynamische diagrammen. Er zijn veel verschillende soorten statische diagrammen, waaronder; een klassendiagram, een objectendiagram, een componentendiagram en een gebruikersdiagram. Een klassendiagram bestaat uit drie delen, het bovenste gedeelte, het middelste gedeelte en het onderste gedeelte. Het bovenste gedeelte van het klassendiagram bevat onder andere de naam van de klasse, het middelste gedeelte bevat de attributen van het diagram. Dit zijn meestal encapsulated (private) attributes, en deze kan dus alleen vanaf de klasse zelf geroepen worden. Het onderste gedeelte bevat de methoden (Lees OOP) van het klassendiagram. Je gebruikt UML dus om een voorbeeld te schetsen van hoe je systeem er uit gaat zien.

Figuur 4

# **Systeemomschrijving**

De applicatie kan alleen gespeeld worden op desktop apparaten. We maken hierbij gebruik van C# WPF. Elke client kan via het klikken op het exe bestand om het spel opstarten.

## **WPF**

WPF, ook wel “Windows Presentation Foundation” is een programmeertaal ontworpen voor computer applicaties. Je kan met WPF verschillende dingen uitvoeren. In het programma word de UI geschreven in XAML, en de Application Logic wordt geschreven in C# en het programma word gebruikt bij het “.NET-framework” Denk hierbij aan het designen van 2D- en 3D graphics, het maken en laten uitvoeren van verschillende animaties, tekst invoeren in je programma, de controls (en dus keybinds) van je spel vastleggen en nog veel meer. Voor ons project gebruiken wij de laatste openbare versie van WPF, WPF 4.6.

# **Bibliografie**

Coding, K. O. (2020, 2 27). *bject Oriented Programming - The Four Pillars of OOP*. Retrieved from Youtube: https://www.youtube.com/watch?v=1ONhXmQuWP8

Doherty, E. (2023, 8 23). *What is object-oriented programming? OOP explained in depth*. Retrieved from educative: https://www.educative.io/blog/object-oriented-programming

Hernandez, R. D. (2021, 4 21). *The Model View Controller Pattern – MVC Architecture and Frameworks Explained*. Retrieved from FreeCodeChamp: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/

*Model-view-controller-model*. (2022, 9 22). Retrieved from Wikipedia: https://nl.wikipedia.org/wiki/Model-view-controller-model

Yadav, N. (2023). *4 Pillars of OOPs in Java* . Retrieved from The Java Programmer: https://www.thejavaprogrammer.com/4-pillars-of-oops-in-java/