

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Auteurs: | | Bart Janisse | | Patrick van ieperen | |  | | Datum: 29-01-2013 | | **Fontys Life 2013**  Software ontwerp |



Fontys Life 2013

Software ontwerp

|  |  |
| --- | --- |
| Opdracht: | Business case robot simulatie |
| Begeleider: | Erik van der Schriek (docent) |
| Instelling: | Fontys hogeschool Eindhoven |
| Studie: | ICT & Technology |
| Schooljaar: | 2013 - 2014 |
| Datum: |  |
| Student: | Bart Janisse / Patrick van ieperen |
| Studentnummer: | 2213829 / 123456 |
| Correspondentie email: | [b.janisse@fontys.student.nl](mailto:b.janisse@fontys.student.nl) |

Inhoudsopgave

[1 Wijzigingshistorie 2](#_Toc374467938)

[2 Inleiding 3](#_Toc374467939)

[2.1 doel 3](#_Toc374467940)

[2.2 Definities en afkortingen 3](#_Toc374467941)

[3 Requirements 4](#_Toc374467942)

[3.1 Niet functionele requirements 4](#_Toc374467943)

[3.2 Functionele requirements 4](#_Toc374467944)

[4 Grenzen van het systeem 6](#_Toc374467949)

[4.1 Use cases 6](#_Toc374467951)

[4.2 Use case diagram 11](#_Toc374467952)

[5 GUI 12](#_Toc374467953)

[6 Concept klasse diagram 13](#_Toc374467954)

# Wijzigingshistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | datum | Wijziging | Auteur(s) |
| 0.1 | 15-12-2013 | Concept | Bart / Patrick |
| 0.2 | 16-12-2013 | Klasse Obstakel toegevoegd | Patrick |
| 0.2 | 16-12-2013 | Klasse plant toegevoegd | Patrick |
|  |  |  |  |
|  |  |  |  |

# Inleiding

## doel

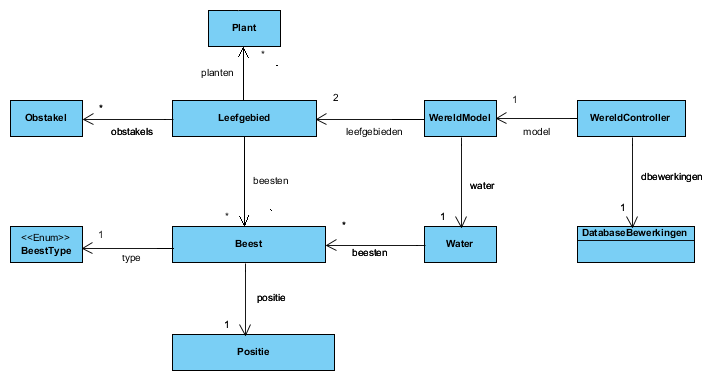
Dit document beschrijft in gedetailleerd het software ontwerp.

## Definities en afkortingen

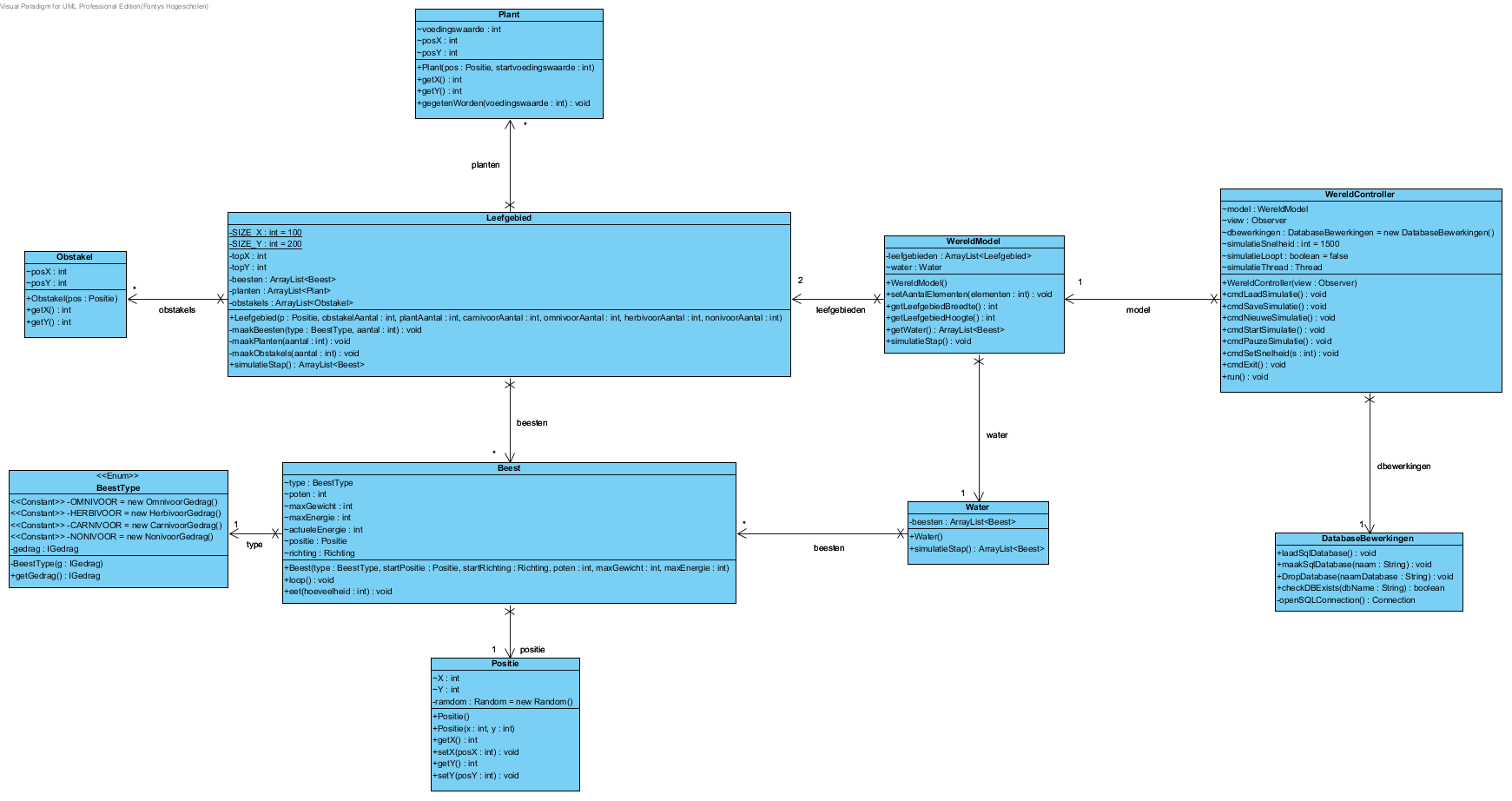
|  |  |
| --- | --- |
| Afkorting | Betekenis |
|  |  |
|  |  |
|  |  |
|  |  |

# Algemene architectuur

Onderstaande afbeelding geeft de globale architectuur weer. Hierin is de onderlinge samenhang van de klassen te zien. Later zal per klasse een gedetailleerde beschrijving worden gegeven.



# Gedetailleerde architectuur



# Klassen

## Beest

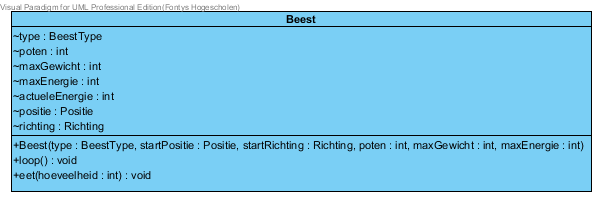
Een beest is in staat om zijn eigen positie te bepalen. Hiervoor is gekozen zodat het leefgebied dit niet van alle beesten hoeft bij te houden.

De vraag die nu rijst is: Hoe weet het beest waar hij naar toe mag? Er kan immers een obstakel in de weg staan.

Uiteindelijk kiezen we er toch voor dat het leefgebied de positie gaat bepalen. Deze weet immers ook waar de obstakels staan enz. Het beest kan wel zijn positie onthouden. Hij krijgt dus van het leefgebied op naar welke positie hij moet.

Omdat het leefgebied ook de obstakels kent zal deze ook de richting van het beest bepalen. Hij kijkt of het nieuwe coördinaat van het beest bezet met een obstakel is en als dit zo is dan zal het leefgebied de richting van het beest veranderen.

### Diagram



### Functie

Deze klasse is een representatie van een beest.

### Constructor(s)

public Beest(BeestType type, Positie startPositie, Richting startRichting, int

poten, int maxGewicht, int maxEnergie)

{

this.type = type;

this.poten = poten;

this.maxGewicht = maxGewicht;

this.maxEnergie = maxEnergie;

this.positie = startPositie;

this.richting = startRichting;

}

parameters

|  |  |  |
| --- | --- | --- |
| **type** | **naam** | **omschrijving** |
| BeestType | type | Type dat een beest kan zijn: Omnivoor, Herbivoor, Carnivoor, Nonivoor |
| Positie | startPositie | Positie waar het beest begint binnen het leefgebied |
| Richting | startRichting | De looprichting waarmee het beest start |
| int | poten | Het aantal poten wat het beest heeft |
| int | maxGewicht | Het maximale gewicht wat het beest kan krijgen |
| int | maxEnergie | De maximale hoeveelheid energie die een beest kan krijgen |

### Methoden

Loop verzorgt een simualtie stap waarbij het beest afhankelijk van zijn richting een gridstap maakt.

public void loop()

{

Positie vorige = new Positie();

vorige.X = positie.X;

vorige.Y = positie.Y;

positie.X += richting.getVerplaatsingX();

positie.Y += richting.getVerplaatsingY();

}

Eet kan worden gebruikt om het beest te laten eten. Afhankelijk van zijn gedrag zel de hoeveelheid worden verwerkt.

public void eet(int hoeveelheid)

{

IGedrag gedrag = this.type.getGedrag();

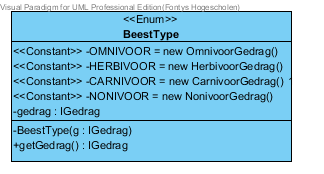
actueleEnergie = gedrag.eet(hoeveelheid, actueleEnergie);

}

## BeestType

Enumeratie voor het type beest

### Diagram



### Functie

In de enumeratie BeestType ligt het gedrag vast. Als een beest bijvoorbeeld als type OMNIVOOR wordt gecreëerd, dan krijgt deze automatisch het Omnivoorgedrag. Het gedrag kan worden opgevraagd.

### Constructor(s)

De constructor voor de enumeratie is private en kan niet worden aangeroepen.

### Methoden

Geeft het gedrag terug wat bij het type beest hoort.

public IGedrag getGedrag()

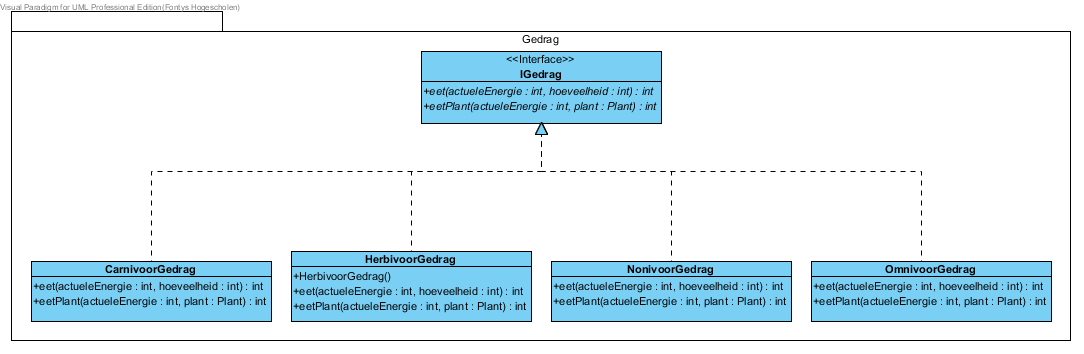
{

return this.gedrag;

}

## Gedrag

Het gedrag is vast gelegd in de interface IGedrag. Dit gedrag wordt voor ieder beesttype in een eigen klassen geïmplementeerd. Per implementatie kan het gedrag zoals eten en voorplanten specifiek worden gemaakt. Het gedrag wordt vastgelegd in de enum BeestType



## OminovoorGedrag

## HerbivoorGedrag

## CarnivoorGedrag

## NonivoorGedrag

## Leefgebied

## Obstakel

Een obstakel krijgt bij de creatie een positie mee via de constructor. Deze positie blijft gedurende hele simulatie het zelfde. Doordat het leefgebied deze positie ook weet, is het niet nodig om een getter voor de positie te

maken. Omdat een obstakel alleen maar een positie gebruikt, is het dan nog wel nodig dat we een klasse obstakel aanmaken. Het leefgebied weet immers waar de obstakels staan, en weet ook waarheen de beesten verplaatsen.

## Plant

Een plant krijgt bij zijn creatie een start positie mee en een start voedingswaarde. De positie van de plant blijft gedurende de simulatie hetzelfde.

Het groeien van een plant gaat als volgt te werk:

* Plant wordt gecreëerd met een voedingswaarde. Mee gekregen via de constructor.
* Elke simulatie stap groet de plant 1 voedingswaarde
* Na het eten worde de voedingswaarde van de plant gereduceerd. Hoeveelheid is afhankelijk van het gedrag van een beest. Dit kan door gaan tot voedingswaarde 0.
* Indien de waarde 0 is kan de plant weer gewoon gaan groeien. Is deze “reïncarnatie” meer dan 10 maal uitgevoerd dan kan de plant niet meer groeien voor 100 simulatie stappen. Daarna gaan de plant weer gewoon groeien.

## Water

## Wereld