|  |  |
| --- | --- |
|  | Parkeersimulator |
|  |  |
| 2/2/2018 | Groep π |
|  | Joeri de Wagt 373603  Mike Dijkstra 368983  Kerwin Sneijders 377834  Benjamin Meijer 331274  Jelle de Jonge 299499 |

Inhoud

[Probleemdefenitie 2](#_Toc504655477)

[Analyse van de as-is situatie 3](#_Toc504655478)

[MVC gebruiken 4](#_Toc504655479)

[Verschil MVCDynamicModelThread en MVCDynamicModelThreadGeneralized 4](#_Toc504655480)

[Verschil MVCDynamicModelThreadGeneralized en Life 4](#_Toc504655481)

[Project omzetten naar MVC 5](#_Toc504655482)

[Eerste uitbreidingen 6](#_Toc504655483)

[Het model zoals de simulator ons werd toegereikt. 6](#_Toc504655484)

[Het model nadat wij er MVC op toegepast hebben. 6](#_Toc504655485)

[Sequensen diagram voor het aanmaken van locaties 7](#_Toc504655486)

# Probleemdefenitie

Het bedrijf Cityparking Groningen heeft ons een simulator aangeleverd die een parkeergarage nabootst. Hiermee proberen ze de drukte in de parkeergarage in het centrum van Groningen na te bootsen. Alleen voldoet de simulatie nog niet aan hun verwachting. Het geeft nog niet een realistisch beeld van de werkelijkheid. De simulatie geeft een algemeen beeld van de drukte, maar nog niet van bijvoorbeeld de drukte op een donderdagavond; wanneer er koopavond is. Of een zaterdagavond, wanneer het theater geopend is.

In de huidige simulatie wordt al wel rekening gehouden met abonnementhouders, maar worden er nog geen plekken gereserveerd voor deze abonnees. In de simulatie wordt ook nog geen rekening gehouden met reserveringen, terwijl dat in de praktijk wel kan.

Er ontstaan, op drukke momenten, rijen voor de parkeergarage. Als de rij te lang is, dan gaan potentiële klanten op zoek naar een andere parkeerplek. Het bedrijf wil dit voorkomen door de lay-out van de parkeergarage zo te veranderen, dat er geen tot minimale rijvorming is. Hiervoor willen ze de simulatie van de parkeergarage gebruiken.

# Analyse van de as-is situatie

De situatie van de simulatie op het moment laat te wensen over. Er is onvolledige code voor de simulatie. Er is niet eens een klasse wat ervoor zorgt dat de simulatie het doet. Als we die klasse wel toevoegen en de simulatie runnen zien we een paar problemen. Er wordt geen rekening gehouden met verschillende druktes. Er komen een vaste aantal auto’s binnen en er wordt nauwelijks gefluctueerd op bijvoorbeeld een drukke donderdagavond. Als je de simulatie laat uitwerken dan blijven er alleen nog maar gewone auto’s komen en geen abonnees meer. In een echte situatie zullen er nog wel abonnees blijven komen. Ook raakt de garage nooit vol. Als de topdrukte is, dan zal de garage een keer vol raken. Er ontstaat dan een rij voor de garage. Als er mensen aankomen en de rij te lang vinden, zullen ze een andere parkeerplek gaan zoeken. Ook dit is niet in de simulatie opgenomen.

In de klassen Adhoccar en parkingpasscar zit dubbele code. De code is makkelijker te onderhouden als deze code in de superklassen car wordt gezet zodat de code maar één keer aangepast hoeft te worden.

Als je het programma meerdere keren runt, dan krijg je steeds hetzelfde resultaat; de parkeergarage is ongeveer halfvol, met alleen maar gewone auto’s. Dit is geen waarheidsgetrouw beeld van de werkelijkheid.

# MVC gebruiken

## Verschil MVCDynamicModelThread en MVCDynamicModelThreadGeneralized

Het MVCDynamicModelThread model maakt gebruik van één enkele package en MVCDynamicModelThreadGeneralized verdeelt de klassen onder meerdere packages. Het model MVCDynamicModelThreadGeneralized maakt gebruik van abstracte klassen en MVCDynamicModelThread maakt hier geen gebruik van. Het voordeel van het MVCDynamicModelThreadGeneralized model is dat gezamenlijke methodes kunnen onderbrengen in de superklasse en hierdoor gedupliceerde code verwijderen.

## Verschil MVCDynamicModelThreadGeneralized en Life

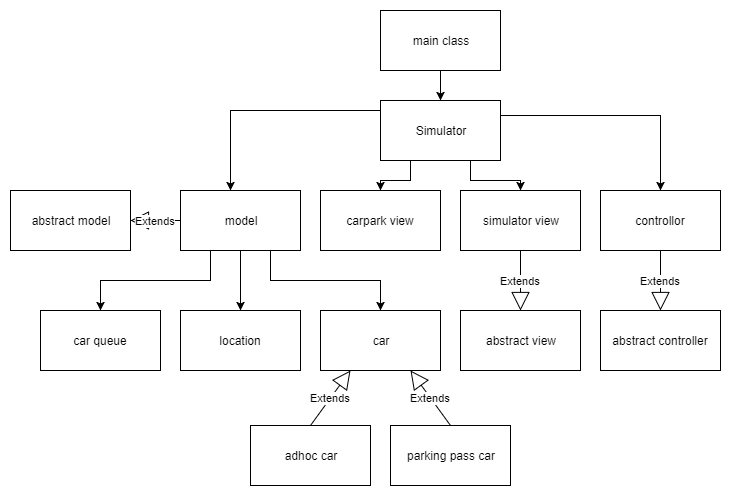
Een verschil van het MVCDynamicModelThreadGeneralized model en het Life model is dat het Life model gebruikt maakt van meerdere controllers. Hierdoor heeft Life een overzichtelijkere opbouw van de code, omdat alle samenhangende code in dezelfde controller zit.

Life heeft ook een exceptions klasse en deze heeft MVCDynamicModelThreadGeneralized niet. Hierdoor kan Life foutmeldingen genereren en MVCDynamicModelThreadGeneralized niet.

## Project omzetten naar MVC

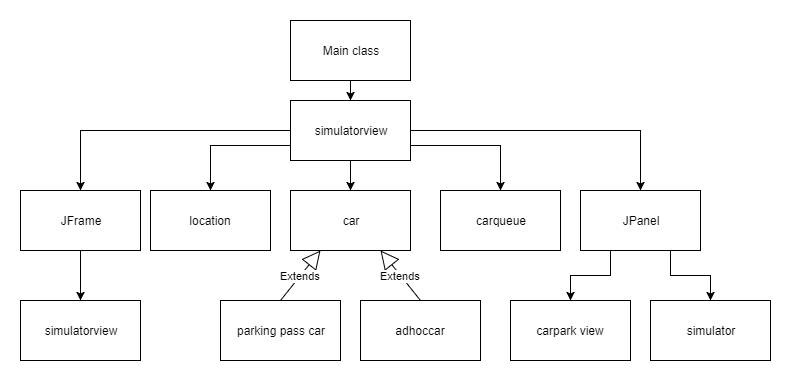
We gaan eerste instantie meerdere packages aanmaken. Namelijk: Controller, main, model, runner en view. Zoals aangegeven door de productowner laten we de controller in eerste instantie leeg. In de package main gaan we een klasse maken die simulator heet en die gaat de hele programmatuur aanmaken. Zoals het model, de verschillende views en de controllers. In de package model zetten we alle klassen die objecten aanmaken, zoals car, location, adHocCar, ParkingPassCar en een nieuwe klasse model. In model staat het grootste gedeelte van alle methodes. Zoals tick, run enzovoort. We gaan een nieuwe package genaamd runner aanmaken. We gaan hier de main klasse inzetten die maakt een nieuwe simulator en start dus het project. We gaan een package genaamd view aanmaken. Hier zetten we de CarParkView in die nu als innerclass bij simulatorView staat. Het wordt dus een eigen klasse in plaats van een innerclass.

Hierbij vindt u het goedgekeurde MVC model, tevens weergegeven in hoofdstuk 4.

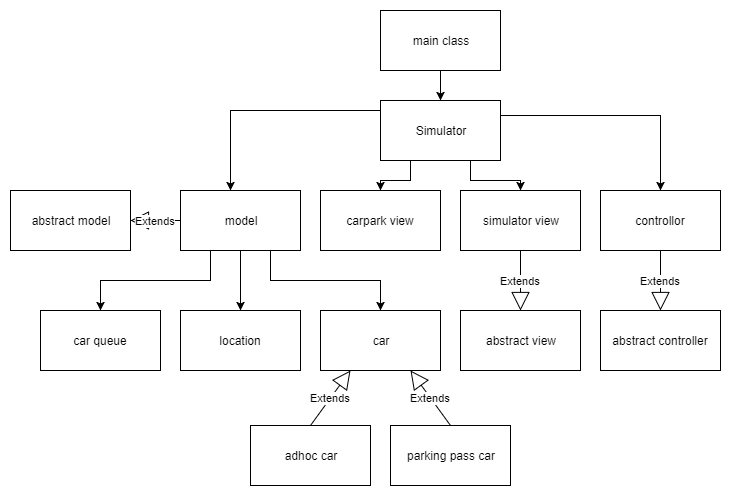


# Eerste uitbreidingen

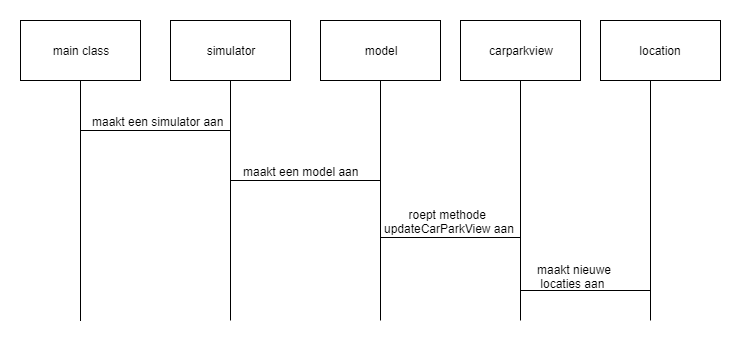
## Het model zoals de simulator ons werd toegereikt.



## Het model nadat wij er MVC op toegepast hebben.



## Sequensen diagram voor het aanmaken van locaties



Na het implementeren van de MVC hebben wij een locationManager gemaakt die locaties als vaste array aanmaakt, in plaats van de locaties bij iedere tick opnieuw aanmaken. Daarbij hebben wij locaties een nieuwe int meegegeven die type heet. Deze wordt gebruikt om onderscheid te maken tussen een gewone plek en een abonneeplek. Het type van een locatie kan met de methode changetype gewijzigd worden.

Daarnaast hebben wij in de controller methodes gemaakt zodat de simulatie met knoppen gestart en gestopt kan worden. Ook kunnen we nu 1 of 100 losse ticks uit voeren.

Er is ook een start gemaakt voor de pieView, Menubar, managementView en de settings.

## Functionaliteiten

Bargraph - Jelle

Piegraph- Kerwin

Linegraph -Joeri

Queue view – Jelle

Queue barView - Joeri

Settings – Mike

Abonnement plekken - done

Reserveringen – done

Managment informatie - Jelle

Klok - Mike

GUI - Joeri

Legenda - Joeri

Slider - Joeri

Knoppen - Kerwin

Plaatje – Mike

Geluid - Jelle

Realistischer maken simulator

* Realistischere rijen – Jelle
* Realistischere opkomst – ben

Plaatje – Mike

Klok - Mike

Settings

Bij instellingen kun je het aantal normale en abonnement auto’s per uur doordeweeks en in het weekend veranderen. Ook kun je de prijs van een parkeerticket instellen. Ook kun je instellen hoeveel abonnement plaatsen er zijn en vanaf welke plek de abonnement plaatsen moeten komen. Verder kun je ook aanpassen hoe lang de rij is voordat auto’s doorrijden omdat de rij te lang is. Ook dit kun je weer doen voor zowel de normale auto rij en de abonnement auto rij. Dit