|  |  |
| --- | --- |
|  | 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

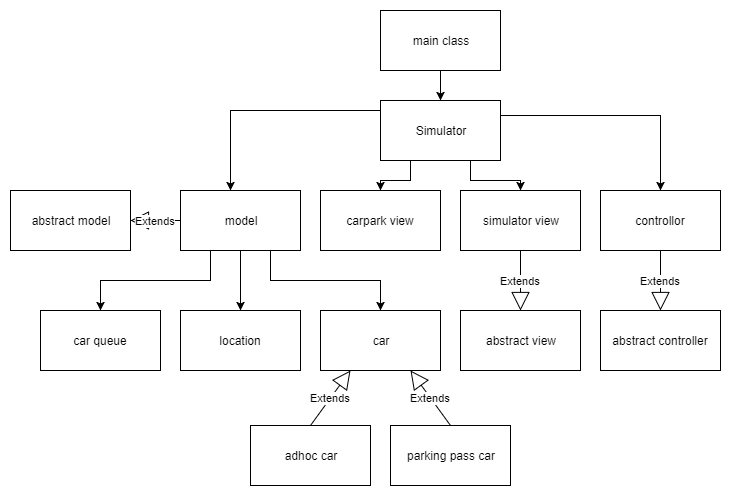
Een verschil van het MVCDynamicModelThreadGeneralized model en het Life model is dat het Life model gebruikt maakt van meerdere controllers. Hierdoor heeft Life een overzichtelijkere opbouw van de code, omdat alle samenhangende code in dezelfde controller zit.

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

## Project omzetten naar MVC

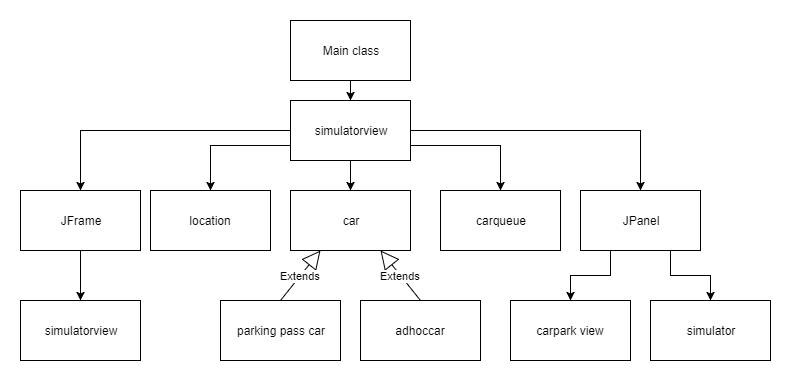
We gaan eerste instantie meerdere packages aanmaken. Namelijk: Controller, main, model, runner en view. Zoals aangegeven door de productowner laten we de controller in eerste instantie leeg. In de package main gaan we een klasse maken die simulator heet en die gaat de hele programmatuur aanmaken. Zoals 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.

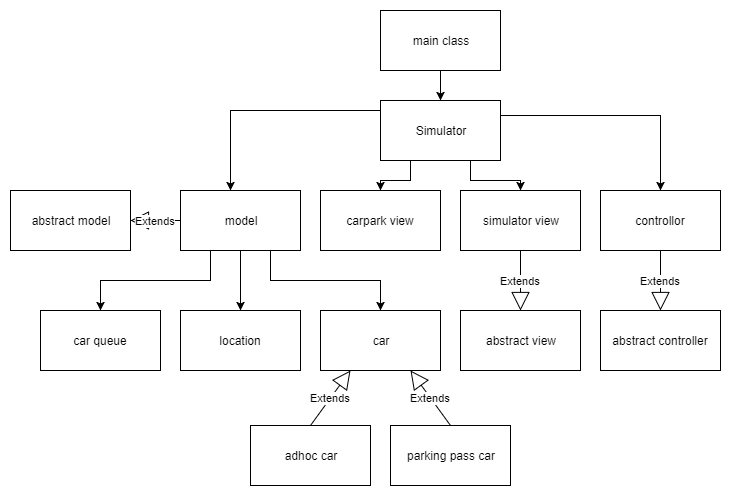


# Eerste uitbreidingen

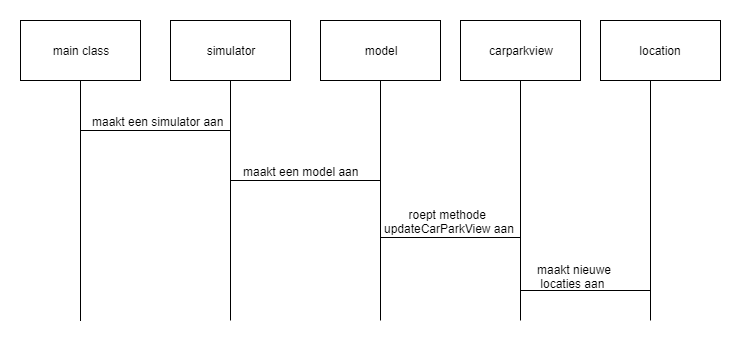
## Het model zoals de simulator ons werd toegereikt.



## 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.

## Black- en whitebox testing

**Whitebox testing.**

Tijdens het proces is er op verschillende manieren gekeken dat wanneer wij de input veranderen of de simulatie nog steeds werkt. In de eindsimulatie kunnen wij de waardes van variabelen die invloed hebben op hoe de simulatie.

Tijdens het testen van deze functionaliteit kwamen wij erachter dat het verhogen van de max lengte normale rij en max lengte abonnement rij geen invloed hadden op de simulatie.  
Daarom is er besloten om deze instellingen uit het menu te halen.

Op het moment dat de prijs per auto negatief gezet wordt, geeft de lijngrafiek de data niet meer goed weer. Hiervoor hebben we ingesteld dat deze waarde niet meer negatief gezet mag worden. Wanneer deze waarde erg hoog wordt gezet blijft de simulatie wel goed werken alleen wordt de lijngrafiek slechter leesbaar. Dit kan echter niet opgelost worden. Er is besloten dat wij wel toelaten het bedrag erg hoog te zetten.

De rest van de waardes breken de simulatie niet.

**Blackbox testing**

Wij zijn naar een testpersoon op zoek gegaan die geen verstand had van code (een leek).

Deze hebben wij ook gevonden en we hebben het programma klaar gezet. Vervolgens heeft deze persoon het programma uitgetest. Dit zou in principe zonder problemen moeten verlopen. Tijdens het testen hebben zich geen problemen voorgedaan. Het programma is niet kapot gelopen door handelingen van deze testpersoon. Het programma werd goed begrepen en de testpersoon kon alle informatie goed aflezen. Er zijn een aantal vragen gesteld wat het programma moest doen of wat het programma liet zien. Na uitleg was duidelijk.

## 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**

Het logo van cityparking toegevoegd aan de GUI. Dit wordt gedaan door een image aan te maken en hier vervolgens een Jlabel van te maken. Deze JLabel voeg je toe aan een JPanel.

**Klok**

De klok van de simulator weergeeft de dag en de tijd in een HH:MM formaat. In de klasse timecontroller wordt de tijd omgezet naar dit formaat. De dag wordt in de hele simulator als int gebruikt om het om te zetten naar een dag in letters wordt een switch statment gebruikt.

**instellingen**

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. De instellingen worden in een apart frame getoond. Om de variabele waarden te tonen en te veranderen worden getters en setters gebruikt.

**Staafdiagram**

De uitbreiding van het staafdiagram laat de huidige staat van de parkeergarage zien. Elke staaf stelt een samenstelling van de aanwezige auto’s voor of de hoeveelheid lege parkeerplekken en van welke soort. De kleuren en de bijbehorende informatie is als volgt:

Rood: De normale aantal auto’s in de garage.

Blauw: De abonnee auto’s in de garage.

Oranje: De auto’s in de garage die gereserveerd hebben.

Wit: De gewone lege parkeerplekken.

Cyaan: De lege parkeerplekken voor de abonnees.

Geel: De lege parkeerplekken voor de auto’s die gereserveerd hebben.

De auto’s worden berekend door met een lus door de aanwezige auto’s te zoeken. Daar wordt bekeken welke kleur ze hebben. De parkeerplekken worden opgevraagd door met een lus door de parkeerplaatsen te zoeken. Daar wordt eerst gekeken of de parkeerplaats onbezet is en daarna wordt gekeken van welke type de lege parkeerplaats is. Deze gegeven worden verzameld en door de grootte van de aantal parkeerplaatsen gedeeld. Daarna doen we het keer 100% om de waardes om te zetten in procenten. Deze worden gecast naar het type integer om ervoor te zorgen dat het gehele waardes worden.

**Wachtrij view**

In de view van de wachtrijen kun je de verschillende auto’s zien die in bepaalde wachtrijen staan. De normale auto’s in de rij laat zien hoeveel auto’s in de rij staan voor de ingang. De abonnees in de rij laat zien hoeveel abonnees in de rij staan voor de ingang. Dit geldt ook voor reserveringen in de rij, maar dan voor de reserveringen. De informatie die de huidige betalingsrij laat zien zijn de mensen die in de rij staan om te betalen. Dit kunnen alleen normale auto’s en reserveringen zijn, omdat abonnees niet hoeven te betalen bij de kassa. De reserveringen betalen ook 20% meer omdat zij een extra kostenpost hebben vanwege de reservatie. De informatie die uitgang wachtrij laat zien is hoeveel auto’s in de rij staan om de garage te verlaten. De informatie die vertrokken uit de rij laat zien is de hoeveelheid mensen die niet in de rij zijn gaan staan of eerder zijn vertrokken uit de rij. Meer hierover verderop in het rapport.

**Managementinformatie**

In de managementinformatie kun je informatie vinden over de inkomsten van de garage. De dagomzet laat de omzet zien dat realtime wordt geüpdatet. Als de dag voorbij is wordt de dagomzet in de bijbehoren dagomzet gezet. Dus als het bijvoorbeeld maandag is en de tijd gaat over naar de dinsdag, dan wordt de dagomzet in de maandagomzet gezet. Dit gaat de hele tijd zo door en als het weer maandag is wordt de omzet die dan in de maandagomzet staat overschreven. Als laatste laat de managementview de huidige weekomzet zien. Die doet hij door alle omzetten van de week op te tellen. Dus als bijvoorbeeld de dinsdagomzet veranderd, updatet de simulator ook gelijk de huidige weekomzet. De omzetten worden berekend door een variabele die zichzelf ophoogt elke keer als iemand door de betalingsrij is geweest. Aan het einde van de dag wordt deze variabele weer naar 0 gezet.

**Geluid**

Er wordt elk dag een geluid afgespeeld. Het geluid waarvoor we gekozen is van een muntstuk. Dit ten teken dat de dag voorbij is en weer nieuwe omzet gerealiseerd is.

**Realistischere rijen**

De huidige simulatie van de parkeergarage laat een realistischere weergave zien van de werkelijkheid. Onder andere omdat de rijen niet meer oneindig doorgroeien. Dit is gerealiseerd door elke keer dat een auto toegevoegd is de rij te kijken of de rij niet te lang is. In de huidige simulatie kijkt hij of de rij niet langer is dan 30. Als dat zo is dan laat hij de auto in de rij staan. Als de rij wel langer is dan haalt hij de auto weer weg uit de rij. Dus in een echte situatie is dat hetzelfde als dat een auto doorrijdt. We kijken niet of de rij te lang is voordat de auto toegevoegd wordt, omdat we anders niet kunnen bijhouden of een auto is doorgereden of niet. Dit is nodig om de informatie bij te houden over doorgereden auto’s. Verder wordt er aan elke auto een timer toegevoegd. Dit is nodig om ervoor te zorgen als auto’s te lang in de rij staan dat ze vertrekken. De tijd die meegegeven wordt aan elke auto is random. Ze kunnen willekeurig tussen de 5 en 15 minuten weer vertrekken uit de rij. Elke keer als dit gebeurt wordt deze informatie ook weer meegegeven, zodat de informatie over de vertrokken auto’s ook geüpdatet wordt.

**Reserveringen**

De uitbreiding reserveringen zorgt ervoor dat er iedere tick een kans is dat er reserveringen aangemaakt wordt. Wanneer dit gebeurd wordt er een reservering en een reserveringsauto aangemaakt. Ze krijgen beide hetzelfde kenteken mee.

De reservering krijgt dan ook een willekeurige tijd tussen 20 minuten en 1 uur en 15 minuten later mee.

De reserveringsauto krijgt dan de tijd van de reservering door. Deze tijd wordt in de reserveringsauto omgezet naar de tijd dat de auto aankomt. Deze tijd krijgt een random afwijking tussen 20 minuten voor de tijd van de reservering en 40 minuten na de tijd van de reservering.

De hoeveelheid reserveringen die tijdens een tick gemaakt moeten worden wordt bepaald door de methode getNumberOfCars.

Nadat tijdens een tick de auto’s die weg zouden gaan zijn weggegaan wordt er gekeken of er over 15 minuten een reservering staat. Wanneer dat zo is veranderd de simulatie een locatie naar een reserveringslocatie.

Tijdens de methode carsArriving wordt er gekeken of er een reserveringsauto is die nu aan zou moeten komen. Wanneer dit het geval is wordt die auto toegevoegd aan de rij.

Iedere keer dat er gekeken wordt naar de eerste auto in de abonneerij kijkt de simulatie ook of de voorste auto een reservering heeft. Wanneer dit het geval is wordt die auto op de locatie gezet met dezelfde kenteken. Die locatie wordt op dat moment ook weer omgezet naar een normale locatie en de timer van de locatie wordt op 0 gezet.

Echter omdat auto’s ook te vroeg of te laat kunnen komen worden ze op dan geweigerd. Hun kenteken wordt niet herkend omdat er geen locatie met hun kenteken is. Wanneer dit het geval is haalt de simulatie die auto weer uit de rij. Dit om te simuleren dat een auto geweigerd wordt en daarom weg rijdt.

Als laatste krijgen de locaties die een reservering worden een timer van 45 ticks mee. Dit omdat de plek 15 minuten van tevoren gereserveerd wordt en 30 minuten wordt vastgehouden. Wanneer de timer 0 bereikt wordt de plek weer in een normale plek omgezet.

**Abonnementen**

De uitbreiding abonnementen zorgt ervoor dat elke locatie de mogelijkheid heeft om ook een abonnementsplek te zijn. Dit zodat er plekken zijn waar alleen pashouders mogen parkeren en de gewone auto’s en de reserveringsauto’s daar niet mogen parkeren.

Iedere tick wordt er een willekeurig aantal abonneeauto’s aangemaakt. Deze auto’s komen ook in de passHolderQueue in plaats van de paymentCarQueue. Hierbij wordt ook rekening gehouden met een maximaal aantal abonnees. Dus wanneer er meer abonnees zouden komen dan er kunnen zijn, wordt dit tegengehouden na het maximaal aantal abonnees.

Op het moment dat er een abonneeauto vooraan een queue staat wordt er gekeken door de simulatie of er een lege abonneeplek is. Wanneer dat het geval is wordt de auto op die plek gezet. Wanneer dit niet het geval is krijgt de auto een normale niet gereserveerde plek toegewezen.

Wanneer deze auto’s weg rijden hoeven ze ook niet meer te betalen.

**Realistischer maken van de simulatie – realistischere opkomst**

Om ervoor te zorgen dat de hoeveelheid auto’s die binnenkomt realistischer is. Word er een factor aangemaakt die per type auto (normale auto, abonneeauto en reserveringsauto) de hoeveelheid binnenkomende auto’s vergroot of verkleint. Dit gebeurt aan de hand van de tijd van de dag.

Dit zorgt er bijvoorbeeld voor dat er tijdens de nacht veel minder auto’s binnenkomen dan tussen 08:00 en 17:00.

Daarbij word er ook rekening gehouden met de koopavond op donderdag avond en de theatershows die op vrijdag avond, zaterdag avond en zondag middag plaats vinden.

**LineGraph**

De lijn grafiek word gebruikt voor het laten zien van de verdiensten per dag. Hier hoeft niks voor ingevoerd te worden en word allemaal automatisch bijgehouden. Wanneer je de snelheid omhoog zet zal de lijn ook sneller op de grafiek getekend worden. De punten zijn aangegeven door de hoeveelheid er op die dag is verdient. De grafiek houd zelf bij als de lijn hoger gaat dan de component, waarna de grafiek word berekend en weer goed word neergeplaatst. Zodat je altijd de lijn van de grafiek zal zien.

**Queue barview**

De queue barview is een visuele weergave van de van de wachtrijen die aanwezig zijn. Auto’s die aan komen bij de parkeergarage en er niet gelijk in kunnen rijden komen in de wachtrij te staan. Nou zijn er een aantal verschillende auto’s en daarom is het ook netjes om 3 verschillende wachtrijen te hebben. Er zijn 3 wachtrijen voor auto’s zoals: Reserveringen, Abonnementen en Normale auto’s. Wanneer het programma word gestart zullen deze wachtrijen leeg zijn en dus wit. Eenmaal wanneer de wachtrijen gevuld worden komen er kleuren te staan die bij de soort auto behoren. Wanneer de wachtrij volledig vol is zal er een timer gaan lopen, want niet iedere bestuurder wil wachten totdat de hele wachtrij weg is. Ook is er een limit gesteld aan de wachtrij zodat er maar een aantal mensen in kunnen wachten. Dit leek ons realistischer, want een wachtrij van meer dan kilometers lang is zal niemand in gaan staan.

**GUI**

De GUI (Graphical User Interface) die gemaakt is een frame waar alle componenten in gezet zijn en een positie is aangeven. Alle componenten bijvoorbeeld: de taartgrafiek, linegraph en de knoppen zijn allemaal losse componenten en kan je handig op deze manier componenten aanpassen of verwijderen. Er zijn ook instellingen voor het programma om de simulator realistischer te maken. Zoals het aantal auto’s per dag binnen komt en hoe snel ze weer vertrekken. Dit instellingscherm word in een apart scherm naar voren gehaald, want de instellingsscherm hoeft niet altijd voor je beeld te staan.

**Legenda**

De legenda geeft weer welke auto’s en parkeerplekken er zijn. Er zijn 6 verschillende types.

* De lege parkeerplaatsen
* Normale auto’s
* Abonnement plekken leeg
* Abonnement auto’s
* Gereserveerde plekken leeg
* Gereserveerde plekken bezet

Het is gewoon een stukje extra informatie zodat je weet welke soorten auto’s en parkeerplekken er zijn. Iedere soort plek heeft een kleur achter de naam staan zodat je kan zien waar de auto tot behoort.

**Slider**

Met de slider kan je de snelheid van het programma aanpassen. Hierdoor zal alles in versnelde tempo gebeuren en kan je eventueel als vooruit spoel functie gebruiken. De slider staat normaal op 0 en hoe verder je de slider naar rechts toe sleept de sneller gaat het programma alles maken tot je uiteindelijk op 100 staat en niet sneller kan.

**Taart-diagram**

De uitbreiding taart-diagram zorgt voor een simpel en duidelijk overzicht van alle parkeerplaatsen in de simulatie. De taart-diagram is verdeeld in zessen.   
De kleuren en de bijbehorende informatie is als volgt:   
Wit: De gewone lege parkeerplekken.  
Rood: De normale aantal auto’s in de garage.   
Cyaan: De lege parkeerplekken voor de abonnees.   
Blauw: De abonnee auto’s in de garage.  
Geel: De lege parkeerplekken voor de auto’s die gereserveerd hebben.   
Oranje: De auto’s in de garage die gereserveerd hebben.

Deze worden netjes naast elkaar weergegeven met de juiste informatie. Alle gegevens worden opgehaald met getters gemaakt in de model. Met deze informatie word dan berekend hoeveel graden van de taart-diagram gelijk staan aan het aantal (niet) bezette plekken in de garage. Dit gebeurt voor alle drie de types.

**Knoppen**

De uitbreiding knoppen zorgt voor een mogelijkheid om de applicatie te kunnen besturen. Er zijn vijf knoppen toegevoegd.   
De knoppen en de bijbehorende informatie is als volgt:

+1 minuut. Deze knop zorgt ervoor dat de tijd met één minuut vooruit gaat. Alle andere componenten van de applicatie reageren hierop.

+1 uur. Deze knop zorgt ervoor dat de tijd met één uur vooruit gaat. Alle andere componenten van de applicatie reageren hierop.

+1 dag. Deze knop zorgt ervoor dat de tijd met één dag vooruit gaat. Alle andere componenten van de applicatie reageren hierop.

Start. Deze knop zorgt ervoor dat de tijd automatisch met de gegeven snelheid (Zie slider) vooruit begint te lopen.

Stop. Deze knop zorgt ervoor dat de automatisch lopende tijd stopt als deze eerder gestart was met de Start knop.

Met deze knoppen kan de simulatie-tijd gemanipuleerd worden.

Met action-listeners word gekeken welke knop wanneer word ingedrukt. Wanneer dit gebeurt worden er methodes en berekeningen uitgevoerd om de simulatie op de juiste manier te manipuleren.

## Persoonlijke reflectie en peer-review

**Persoonlijke reflectie Mike:**

Ik vond het een erg leerzaam project. Echter waren veel dingen niet goed uitgelegd van te voren. Gelukkig Er was bijvoorbeeld van te voren niet goed uitgelegd wat een MVC model nou precies inhoud en hoe je dat moet maken. Er waren ook bepaalde dingen zoals Javafx die niet zijn uitgelegd maar erg handig waren geweest als we dat van te voren hadden geweten. Verder vond ik dat de samenwerking uitstekend is verlopen. We hadden duidelijke afspraken waar iedereen zich ook aan heeft gehouden. Verder hebben we elke dag van 10 tot ongeveer 4 uur op school gewerkt wat erg prettig was.

Hieronder vindt u de beoordeling van mijn teamleden met uitleg.

Ik heb Joeri een 8,5 gegeven omdat hij veel kennis heeft overgedragen en hij heeft ook een grote bijdrage aan het project geleverd.

Ik heb Ben een 8 gegeven omdat hij een lastig deel van de code voor zijn rekening heeft genomen.

Jelle heb ik 8 gegeven omdat hij een goede bijdrage aan het project heeft geleverd ook was hij niet te beroert om je te helpen als je ergens niet uitkwam.

Kerwin heb ik een 8 gegeven omdat hij veel mensen heeft geholpen.

**Persoonlijke reflectie Jelle:**

Persoonlijke kijk ik met gemengde gevoelens terug op dit project. Aan de ene kant positief, omdat ik veel heb geleerd over Java. Dit komt omdat je zelf lekker kan prutsen met code en je je eigen fouten moet herstellen. Maar aan de andere kant vond ik de vooraf geleerde stof te weinig. Dit komt vooral door de werkwijze waarop ons Java is geleerd. Met PHP hadden we wekelijks opdrachten waarin we moesten uitpluizen hoe iets werkte. Met Java hadden we ook wel opdrachten, maar hier kregen we de uitwerkingen een pagina eerder net uitgelegd. Hier zat geen uitdaging in.

Verder ben ik wel tevreden over onze werkwijze als projectgroep. Elke dag van 10 uur op school aan het werk tot een uur of 4. Op deze manier kun je gelijk met problemen terecht bij je projectleden die je kunnen helpen. Als we allemaal thuis hadden gewerkt dan waren we vaker vast komen te zitten.

Ik heb Mike een 8 gegeven omdat hij elke dag aanwezig was en ijverig aan het werk is geweest. Ook was hij degene die iedereen weer even tot de orde riep en ervoor zorgde dat het project tijdig af is.

Ik heb Kerwin een 8 gegeven omdat hij met elk probleem die je hebt je graag wil helpen. Hij doet dit elke keer en zeurt geen enkele keer. Zelfs niet als het voor problemen geeft bij zijn eigen werk.

Ik heb Joeri een 8 gegeven omdat hij de meest ervarene is met Java. Hij kan je probleemloos helpen met problemen en doet dit ook.

Ik heb Ben een 8 gegeven omdat hij de meeste doorzettingsvermogen heeft van iedereen. Hij zorgt ervoor dat zijn eigen code goed werkt en duikt goed in de stof als hij tegen problemen aanloopt. Hij is erg zelfstandig.

**Persoonlijke reflectie Kerwin:**

Als ik terug kijk op het project vind ik dat we zeer positief kunnen terug kijken op de laatste week. De laatste week hebben we erg veel kunnen doen. De applicatie heeft enorme vooruitgang geboekt. Daarbij komt wel dat de week ervoor erg stroef ging. We hadden niet genoeg kennis en/of informatie om snel en makkelijk door te kunnen werken. In de weken die we hadden heb ik veel van Java geleerd. Het waren zeker interessante weken, ondanks dat ik 2 belangrijke dagen ziek thuis was.

Ondanks dat de eerste week erg stroef verliep ben ik zeer tevreden over de communicatie en werkwijze in onze project groep. We zijn alle dagen van ongeveer 10 uur ’s ochtends tot 5 uur ’s avonds op school geweest om zoveel mogelijk te bereiken. We hadden afgesproken dat niemand vanuit huis zou werken, maar gewoon als groepje bij elkaar. Hierdoor kan je makkelijk en snel problemen met elkaar verhelpen en andermans opinies vragen. Dit verliep erg soepel.

Becijfering:

**Mike,** Ik geef Mike een **8** omdat hij heel veel tijd heeft gestoken in het oplossen van de problemen in het project als het gaat om de views, daarbij heeft hij ervoor gezorgd dat alle documentatie en andere delen van het project goed en op tijd af waren (inclusief dit document).

**Jelle,** Ik geef Jelle een **8** omdat hij elke dag hard heeft gewerkt aan het project. Ook was hij elke dag op tijd!

**Joeri,** Ik geef Joeri een **9** omdat hij ons uit de problemen heeft geholpen als het gaat om MVC. Hij heeft de problemen die wij niet konden oplossen opgelost en daardoor gezorgd dat iedereen weer verder kon.

**Ben,** Ik geef Ben een 8 omdat hij veel tijd heeft gestoken in het ombouwen en oplossen van problemen van een groot deel van het project.

**Persoonlijke reflectie Joeri:**

Ik zal nu vertellen over hoe ik het project hebt gezien en hoe ik het vond gaan. Over het algemeen vond ik dat we ons als team goed door heen hebben geslagen. Soms vond het project soms een beetje moeizaam verlopen. Dit vanwege de weinig kennis die mensen op hebben gedaan in dit leertraject. Het MVC model was in 1 lesje uitgelegd en verder niet. Misschien had dit net zoals PHP beter kleine opdrachten kunnen wezen en niet in bluej of beide. Waarschijnlijk was er dan ook een groter programma uit gekomen. Verder ben ik wel trots op hoe het team dit doorstaan heeft. Er is enorm goed samengewerkt aan problemen en wanneer het nodig was elkaar goed geholpen. Iedere dag was het streven naar 10 uur om aanwezig te wezen op school. Op school zijn wij samen aan het werk gegaan tot een uur of 4 a 5.

Ik heb Kerwin een 8 gegeven. Dit omdat hij enorm veel probeerde te helpen waar hij kon. Ook deed Kerwin goed zijn werk en taken die hij op zich genomen had.

Ik heb Ben een 8 gegeven. Ben had in het begin een klein beetje moeite met het java gedeelte. Dit omdat hij niet precies wist hoe het werkte vanwege de uitleg, maar eenmaal toen hij los was en het begreep ging hij razendsnel aan de slag. Dat lukte ook goed ook! Hij heeft een belangrijk deel van de applicatie gemaakt.

Ik heb Jelle een 8 gegeven. Jelle deed goed zijn werk dat aan hem gevraagd werd. Hij hielp ook goed mee met de dagelijkse taken die moesten gebeuren en deed alle taken die aan hem opgedragen was. Ik heb

Mike een 8 gegeven. Mike was heel erg serieus met het project en je zag aan hem dat hij er echt iets van wou leren. Dit is ook goed gelukt, want hij heeft een belangrijk deel van de applicatie geïmplementeerd.

## Conclusie en aanbevelingen

Alle gevraagde functionaliteit is toegevoegd. Bovendien zijn er enkele prestatieproblemen die in het originele project aanwezig waren opgelost. Wij raden echter wel aan om het project om te schrijven naar javafx in plaats van swing. Dit omdat javafx beter werkt en makkelijker is om te onderhouden. Dit zorgt er ook voor dat de hoeveelheid code wordt verminderd.