**Eindrapport**

Meme-Tegeltjeswijsheden

| Naam student | Student nummer |
| --- | --- |
| Lars Caspers | 390533 |
| Maurice Wijker | 387286 |
| Johan Schuringa | 387146 |
| Casper Scholte-Albers | 375329 |

Inhoud

[Hoofdstuk – 1 Probleem analyse 3](#_Toc535485618)

[Probleem definitie 3](#_Toc535485619)

[Hoofdstuk – 2 Analyse as-is situatie 4](#_Toc535485620)

[Analyse van de as-is situatie 4](#_Toc535485621)

[Problemen en oplossingen 4](#_Toc535485622)

[Toevoegingen 4](#_Toc535485623)

Hoofdstuk – 1 Probleem analyse

## 

## Probleem definitie

Het parkeerbeleid dient efficiënter gemaakt te worden, zodat de inkomsten gemaximaliseerd kunnen worden. Daarnaast dient een plek van tevoren gereserveerd te kunnen worden. Om het parkeerbeleid zo efficiënt mogelijk te maken dient er rekening gehouden worden met een aantal variabelen: De bezetting en de verhoudingen tussen gereserveerde plekken voor abonnementhouders, niet gereserveerde plekken en reguliere gereserveerde plekken.

Simulaties kunnen dit soort bedrijfsvraagstukken oplossen. Men kan in een simulatie de bovengenoemde variabelen veranderen. Het resultaat van deze veranderingen wordt veel sneller duidelijk in de simulatie dan in het echt. Ook hoeft er niet getest te worden in de echte wereld, wat vaak geld scheelt.

Hoofdstuk – 2 Analyse as-is situatie

### Problemen en oplossingen

Opvallend aan de huidige situatie is dat de parkeergarage steeds tot ongeveer de helft vol loopt en blijft dan rond dat gebied hangen. Dit is niet in verhouding met een echte situatie omdat er bij een echte parkeergarage momenten zijn dat de garage weinig bezoekers heeft of juist helemaal vol zit. Om een beter idee te geven over hoe de parkeergarage drukken momenten doorkomt zal er een GUI gemaakt kunnen worden waarin de instroom en uitstroom van auto’s bepaald kan worden om zo drukte momenten te kunnen simuleren.

### Toevoegingen

Het zal mogelijk moeten zijn om plekken te reserveren in de parkeergarage (abonnementhouders en niet-abonnementhouders). Dit kan gedaan worden door stukken de garage alleen voor reserveringen beschikbaar te maken. Daarnaast moet de GUI uitgebreid worden om meer informatie te kunnen tonen.

Hoofdstuk – 3 MVC gebruiken

## MVCDynamicModelThread t.o.v MVCDynamicModelThreadGeneralized

De MVCDynamicmodelthreadgeneralized deelt het project in een MVC structuur door zowel de klassen in de MVC structuur te plaatsen als packages. De klassen bestanden (.java) die bij M, V of C horen zitten dan ook in de bijbehorende packages. Daartegenover deelt MVCDynamicModelTread enkel de klassen in een MVC structuur in.

Ook heeft MVCDynamicmodelthreadgeneralized abstracte parentklasses bij ieder onderdeel van MVC. De abstracte parentklassen nemen een gedeelte van de functionaliteit van de normale MVC over. De normale MVC klassen overerven hiervan.

Verder hebben de abstracte controller en view beide een verschillend serialversionUID. Dit zijn beiden constanten. Vermoedelijk zijn het verschillende serialversionUID’s omdat die beide verschillende instanties van het model moeten benaderen.

## MVCDynamicModelThreadGeneralized t.o.v. Life

**Exceptions:**

In het bestand “Life” staat een extra map genaamd “exceptions”. In exceptions staat “LifeException”. Deze klasse overerft van de Java Klasse “Exceptions”. Dit zorgt ervoor dat er exceptions kunnen worden gegenereerd.

**Controller:**

Bij beide bestanden zorgen de controllers er voor dat er knoppen en invoervelden beschikbaar zijn. Hiermee kun je de simulaties bedienen.

**View:**

Beide MVC’s laten in een view het resultaat zien van de ingevoerde parameters en ingedrukte knoppen. Elk resultaat wat getoond wordt, individueel aangemaakt in een eigen view klasse.