Course jaar : **2017-2018**
Blok : **1.2**

Titel : # Project Parkeergarage

BB-course : **ICTVT 1.2 Design and Build 2017-2018**

Project Parkeergarage

Tijdens dit project zullen jullie in projectgroepen van 3 of 4 personen gaan werken aan een simulatieprogramma. Het doel van deze applicatie is om aan een parkeergarage informatie te verstrekken die gebruikt kan worden om het parkeerbeleid, met name de indeling van de garage met gereserveerde plekken voor abonnementen en reserveringen te optimaliseren. In dit project zullen alle onderwerpen die de revue hebben gepasseerd tijdens dit thema (Object georiënteerd programmeren, configuration management, project management en human computer interaction) aan bod komen.

Case beschrijving “parkeergarage Cityparking Groningen”.

(Dit is een verzonnen case; het bedrijf bestaat niet echt in Groningen).

1. De Organisatie.

De parkeergarage “Cityparking Groningen” bevindt zich in het centrum van Groningen, een grote stad in het noordoosten van Nederland. De garage heeft 500 parkeerplekken verdeeld over drie verdiepingen. Het bedrijf heeft 15 werknemers en is 24 uur per dag, zeven dagen per week open.

Bezoekers kunnen hunnen auto in de garage parkeren door een kaartje te nemen bij het inrijden, en bij vertrek te betalen bij de in de garage geplaatste betaalautomaten. Een andere mogelijkheid is om een abonnement te nemen en te parkeren op een van de voor abonnementhouders gereserveerde plekken. Als extra service wil het berdrijf het mogelijk maken om via de website van de parkeergarage van tevoren een plek te reserveren. Hierbij moet de verwachte aankomsttijd ingevuld worden. Er wordt dan van een kwartier van tevoren tot een half uur na de aangegeven tijd een plek gereserveerd.

De garage heeft twee ingangen en één uitgang. Eén van de twee ingangen wordt gebruikt voor abonnementhouders en zal ook door de klanten die een plek hebben gereserveerd kunnen worden gebruikt.

De garage maakt bij het in en uitrijden van de garage gebruik van nummerbord herkenning. Abonnementhouders hebben dus geen kaartje nodig en kunnen zonder langs de betaalautomaat te gaan uit de garage rijden. Betaling van de reserveringen gebeurt wel bij de betaalautomaten. Dit gebeurt op basis van het nummerbord. De klant voert de gegevens van zijn nummerbord in, betaalt voor de periode die hij in de garage heeft gestaan en kan vervolgens uitrijden.

Er is een maximum aan het aantal abonnementhouders omdat maar een beperkt deel van de garage gereserveerd wordt voor deze klanten. Er is dus ook een wachtlijst voor abonnementen.

Naast de parkeergarage staat het beroemde theater “het concertgebouw”. Dit theater biedt plaats aan 1000 bezoekers en elk weekend zijn er voorstellingen op vrijdag en zaterdagavond en zondagmiddag die altijd uitverkocht zijn. Dit zorgt voor extra drukte in de parkeergarage. Verder zijn er op donderdagavond koopavonden die ook voor extra klanten zorgen.

Op sommige momenten is de rij voor de garage zo lang dat potentiele klanten doorrijden en op zoek gaan naar een andere parkeerplek.

2. De simulator.

“Cityparking Groningen” gebruikt het programma *project-parkeer-garage*. Dit project is een simulatie van het gebruik van de parkeergarage. De organisatie wil de simulatie gebruiken om te onderzoeken wat het effect is van het veranderen van het maximumaantal abonnementhouders en reserveringen op de bezettingsgraad en de omzet van de garage en ook op het aantal gemiste klanten. In de huidige applicatie is de volgende functionaliteit al geïmplementeerd:

- Een simulatie van het aankomen en vertrekken van de klanten.
    - Queues
    - Aankomst van gewone klanten en abonnementhouders.
- Betaling bij vertrek.
    - Reguliere betalingen
    - Gebruik van abonnement
- Een grafische weergave van de bezetting van de garage.

Het is nog niet mogelijk om een deel van de garage te reserveren voor klanten met een abonnement.

Project programma

Periode week 6, project week (sprint) 1

Kick-off

Er wordt een kick-off voor het project gehouden aan het begin van week 6. Hier zal de inrichting van het project besproken worden. Lees dit hele document goed door voordat je aan de slag gaat. Zo voorkom je dat je bijvoorbeeld je rapport moet herschrijven.

Sprint 1

Tijdens de uitvoering van het project zal via het Agile/Scrum principe gewerkt moeten worden, waarbij gewerkt zal worden met sprints van één week. De rol van product owner zal vervult worden door de docent die de projectbegeleiding doet. Kies zelf een scrummaster in je groep. Deze eerste week zal gebruikt worden om de projectvoorbereiding uit te voeren, de projectomgeving in te richten en de eerste stappen te maken in de productontwikkeling. Let er bij het plannen van de sprints op dat naast het realiseren van de functionaliteit in sommige weken ook andere opdrachten uitgevoerd moeten worden. Dit heeft natuurlijk impact op de velocity die in die weken behaald kan worden.

De sprints worden afgesloten met een bijeenkomst met de docent begeleider/product owner. Dit is een soort uitgebreide sprint-demo waarbij in ieder geval het product moet worden gedemonstreerd. Zorg er dus voor dat je altijd een werkende versie hebt. Verder bespreek je de retrospective, de burndown chart en het scrumboard.

Projectvoorbereiding

De eerste stap die je nu moet nemen voor de projectinrichting, is het inrichten van een scrumboard, bijvoorbeeld op Trello. Maak hierop de backlog aan en de verschillende kolommen die de fases in de sprint en het project aangeven. Nu kan vervolgens de backlog gevuld worden. Hiervoor moet een overzicht gemaakt worden van de functionaliteit die gedurende het project gerealiseerd moet worden. Bestudeer de lijst die in het onderdeel Requirements wordt gegeven goed en maak in de backlog kaartjes aan voor alle onderdelen die aan het product kunnen worden toegevoegd. Aan al deze kaarten moeten nu volgens de Scrum methode punten (storypoints) worden toegekend.

Nadat het inrichten van de projectomgeving en het voorbereiden van het programma, zoals dat in de paragraven hieronder beschreven wordt, zijn uitgevoerd kan de eerste sprint worden gestart. Voer de stappen uit die nodig zijn om de sprint te initiëren, zoals het bepalen van een sprintdoel en het vullen van de spint-backlog. Het aantal dagen dat je nog hebt in deze sprint is natuurlijk afhankelijk van de hoeveelheid tijd die het jullie heeft gekost om de projectinrichting uit te voeren. Spreek ook met elkaar af waar en wanneer de daily stand-ups gehouden zullen worden.

Projectinrichting

Eclipse en Basis Project

Je moet Eclipse (of Intellij IDEA of een andere IDE) downloaden en installeren, zorg er wel voor dat iedereen in je project dezelfde IDE gebruikt, anders krijg je problemen met het delen van de code via GIT. In dit document gaan we er van uit dat je Eclipse hebt geïnstalleerd. Kijk wat je allemaal kunt doen binnen zo'n IDE. Als je een beetje bekend bent met de IDE kun je het project *project-parking-garage* importeren.

Version Control met Git

Installeer de EGit plugin om git versiebeheer te integreren in Eclipse. Maak een git repository voor jullie code op een centrale server zoals GitHub.com of GitLab.com en voeg de aangeleverde code toe aan je repository. Vanaf nu moet je alle veranderingen die je aanbrengt in de code committen en pushen naar deze repository.

Main methode

Om het programma buiten BlueJ te kunnen gebruiken moet je een “main” methode aan het programma toevoegen. Bestudeer appendix E “Running Java without BlueJ” van het BlueJ boek. Hierin staat beschreven hoe je deze methode kunt maken en hoe je een *.jar file maakt. Implementeer deze main-methode nu in je project.

Debuggen

Gebruik de debugger in Eclipse een aantal keren om door het project heen te lopen. Experimenteer met het plaatsen van breakpoints, stepping into en over functies en het opvragen van de waarden van variabelen.

Probleem Analyse

Je moet alvast beginnen met het eindrapport. Deze week moet je de eerste twee hoofdstukken schrijven:

1. Probleem definitie.

In dit hoofdstuk beschrijf je probleem dat je probeert op te lossen. Probeer het probleem op een generieke en realistische manier te beschrijven. Beschrijf hoe simulaties gebruikt kunnen worden om bedrijfsvraagstukken op te lossen.

2. Analyse van de as-is situatie.

Maak een analyse van de huidige staat van het programma. Wat zijn de problemen met het huidige programma, hoe kunnen die verholpen worden, welke functionaliteit moet er worden toegevoegd? Vragen die je kunt stellen om te ontdekken of de simulator bruikbaar is zijn bijvoorbeeld:

- Als ik het programma meerdere keer draai, krijg ik dan hetzelfde resultaat? Is dat goed of slecht?
- Is er een patroon te herkennen in het resultaat?
- Is dit ook wat ik in de werkelijkheid verwacht?
- Is de simulator eenvoudig uit te breiden/te onderhouden? Wat zijn de beperkingen?
- Wat kan ik op internet vinden over dit soort optimalisatie problemen?

Schrijf de eerste twee hoofdstukken en stuur ze naar de docent om ze te laten controleren.

Periode week 7, project week (sprint) 2

Sprint 2

Voer weer de activiteiten uit om de sprint voor te bereiden. Bepaal op basis van de velocity die je in de vorige sprint hebt behaald hoeveel punten je deze week verwacht te verwerken. Let er wel op dat deze week ook nog twee andere activiteiten voor het project moeten worden uitgevoerd.

Refactor naar Model-View-Controller

Bestudeer de structuur van de volgende drie Eclipse projecten die in mvcextended.zip staan:

- MVCDynamicModelThread.
- MVCDynamicModelThreadGeneralized: hetzelfde voorbeeld alleen in dit geval zijn de MVC-aspecten van het programma daar waar mogelijk in abstracte superklassen gezet.
- Life: Een simulatie programma dat ongeveer dezelfde MVC-achtige abstracte superklassen gebruikt. In dit project vind je ook twee verschillende controllers om te laten zien hoe flexibel MVC kan zijn.

Alle drie de projecten maken gebruik van Threading om er voor te zorgen dat het programma blijft reageren zelfs als de simulatie draait. Threading zal uitgebreid behandeld worden in het tweede jaar, het is dus niet noodzakelijk dat je op dit moment compleet begrijpt hoe het programma werkt. Probeer Theading echter wel toe te passen in je eigen simulator.

Vergelijk de implementatie van MVC in het MVCDynamicModelThread project met die in het MVCDynamicModelThreadGeneralized project en vergelijk dit laatste project ook met het Life project. Wat zijn de overeenkomsten en wat zijn de verschillen?

Beschrijf je bevindingen in het rapport in “Hoofdstuk 3 – MVC gebruiken”. Beschrijf ook hoe je in het project *parkeer-garage* dezelfde MVC structuur kunt krijgen als in het *Life* project. Maak een ontwerp dat de nieuwe structuur van het programma weergeeft. Laat dit nieuwe ontwerp door je docent controleren voordat je de veranderingen implementeert.

N.B. Het belangrijkste doel van MVC is om de verschillende delen van de applicatie te ontkoppelen. Een goede manier om te controleren of je MVC implementatie goed is, is door te kijken of je een controller kunt toevoegen zonder dat het model veranderd moet worden.

Klasse Diagram

Maak twee UML klasse diagrammen, de eerste geeft de klassen weer voor de aanpassing naar het MVC patroon en de tweede die de situatie na de aanpassingen weer geeft. Maak ook een sequentie diagram waarin één stap van de simulatie wordt beschreven. Voeg deze diagrammen toe aan je rapport in hoofdstuk 4 “Eerste uitbreidingen”. Beschrijf hierin ook de overige aanpassingen die je deze week gedaan hebt.

Periode week 8 / 9, project week (sprint) 3

Sprint 3

Voer weer de activiteiten uit om de sprint voor te bereiden. Bepaal op basis van de velocity die je in de vorige sprint hebt behaald hoeveel punten je deze week verwacht te verwerken. Deze week moet er naast het ontwikkelen van de applicatie ook een presentatie worden voorbereid worden en moet het verslag afgerond worden. Houdt hier weer rekeing mee.

Afronden verslag

Beschrijf alle veranderingen die je hebt aangebracht in het rapport. Voeg ook een hoofdstuk toe waarin je jullie conclusies en aanbevelingen beschrijft. Dit completeert het “zakelijke” deel van het rapport. Maak het rapport af door je persoonlijke reflectie toe te voegen. Doe dit in een apart hoofdstuk.

Requirements

In dit hoofdstuk volgt een lijst met wensen met betrekking tot het ontwikkelen van de functionaliteit zoals de product owner die verwoordt heeft. De onderdelen die hier genoemd worden zullen vertaald moeten worden naar tickets voor de backlog. Zorg er hierbij voor dat elk ticket een duidelijk afgebakend stuk functionaliteit beschrijft. Elk onderdeel dat hieronder beschreven wordt kan dus één of meerdere tickets opleveren.

Een simpele GUI maken

Ontwikkel een simpele GUI met knoppen die het mogelijk maken om het programma 1 of 100 stappen te laten uitvoeren. Hiermee kun je de applicatie testen. Besteed hier nog niet te veel tijd aan. De komende weken zal je de GUI nog verder moeten uitbreiden en aanpassen. Om een idee te krijgen van hoe je een GUI aan je applicatie kunt toevoegen kun je de vijf Eclipse projecten die in de mvcextended.zip file bekijken. Deze is te vinden op BlackBoard. In deze projecten kun je zien hoe de logica en de GUI van elkaar gescheiden worden in de applicatie. In de tweede week van het project gaan we nader kijken naar de MVC structuur die gebruikt wordt in deze projecten.

Abonnement plekken

In de simulator wordt nog geen rekening gehouden met het feit dat een deel van de plekken specifiek gereserveerd is voor de abonnementhouders. Zorg er voor dat dit wel gebeurt. Verder is de instroom van abonnement houders niet afhankelijk van het aantal abonnementen dat is uitgegeven. Zorg ervoor dat dit ook in de simulator verwerkt wordt.

Extra Views toevoegen

Maak een extra view voor de applicatie, hierin moet informatie in tekstvorm worden gepresenteerd. In deze view moet belangrijke management informatie worden weergegeven. Je moet in deze view de opbrengst van de dag weergeven en de verwachtte opbrengst van de klanten die nog in de garage aanwezig zijn maar nog niet hebben betaald. Maak gebruik van de AbstractView structuur die je in het *Life* project dat in de MVC projecten zit kunt vinden.

Implementeer nog drie andere views, je kunt bijvoorbeeld de manier waarop de huidige informatie wordt weergegeven veranderen (i.e. histogrammen of lijn grafieken). Ook moet je de informatie over de queues weergeven. Zorg er voor dat je in alle views duidelijk kunt zien welk soort auto's weeggeven worden.

Verbeteren van de simulatie

Als je een view hebt gemaakt die de lengte van de rijen weergeeft zal je waarschijnlijk hebben gezien dat er geen rijen ontstaan. Dit is niet zoals het in werkelijkheid is. Pas het simulatie proces aan zodat het aantal auto's dat arriveert per minuut meer lijkt op de werkelijkheid. Houd hierbij rekening met de gegevens die in de casus omschrijving genoemd worden. Pas tevens de simulatie zodanig aan dat er rekening wordt gehouden met het feit dat potentiele klanten mogelijk doorrijden als de rij te lang is. Dit gegeven moet ook in de statistieken worden opgenomen. Maak ook een view waarop de instellingen van de simulator kunnen worden beheerd.

Een klasse voor klanten met reserveringen toevoegen

Voeg een klasse toe voor klanten die hebben gereserveerd. De implementatie van de betaal methode zal verschillend zijn voor de verschillende klanten. Klanten die gereserveerd hebben betalen een extra bedrag voor de reservering. Verder moet de simulatie aangepast worden zodat er plaatsen in de garage gereserveerd worden voor de reserveringen. Bedenkt hoe dit het beste gerealiseerd kan worden. Dus op welk moment moet je plekken gaan reserveren om te

garanderen dat de klanten met reserveringen een plek kunnen vinden. Houdt hierbij rekening met het feit dat niet alle reserveringen komen opdagen en mensen met een reservering te vroeg of te laat kunnen komen. Probeer door middel van de van de informatie die je uit de simulator kunt halen een advies te formuleren voor de organisatie. Probeer bijvoorbeeld te controleren of het geplande reserveringsproces, waarbij vanaf een kwartier voor aanvang een plek gereserveerd wordt altijd voldoende gereserveerde plaatsen oplevert. Hierbij is het dus van belang dat de bezetting van de parkeergarage in de simulator een realistisch beeld geeft van de werkelijkheid.

De garage heeft een modern systeem waarbij de klanten naar hun plek gestuurd kunnen worden en plekken gereserveerd kunnen worden. Reserveringen kunnen dus overal door de garage geplaatst worden, behalve in het deel dat voor de abonnementhouders gereserveerd is.

Bedenk hoe je in de view kunt aangeven of een plek gereserveerd is, een lege plek voor een abonnementhouder of een bezette plek is, en implementeer dit.

Om de simulator een goed beeld van de werkelijkheid te laten geven is het van belang om goed na te denken over het gedrag van de klanten. Dus wanneer komen de verschillende groepen klanten, reserveringen zullen bijvoorbeeld vaak samengaan met een bezoek aan het theater. Wanneer maken de abonnementhouders gebruik van de garage, en wanneer komen de meeste gewone klanten. Maak als het nodig is een aanpassing aan de instroom van klanten in de simulator.

Verbetering van de GUI

Probeer op een bruikbare manier plaatjes en geluiden toe te voegen, je kunt bijvoorbeeld waarschuwingsberichten toevoegen als de rijen te lang zijn of als de bezettingsgraad een bepaald niveau bereikt.

Bonus uitbreidingen

- In tegenstelling tot de vossen en konijnen simulatie die in het BlueJ boek beschreven wordt is deze simulatie zeer geschikt voor event-driven simulatie. Je kunt proberen om dit in de simulator te implementeren of beschrijven welke aanpassingen in de simulator uitgevoerd moeten worden om deze event-driven te maken. Gebruik hiervoor eventueel pseudocode, UML klasse diagrammen en sequentie diagrammen.

- Auto’s kunnen slecht geparkeerd worden, en twee plaatsen in beslag nemen. Implementeer een klasse van foutparkeerders. Wat kun je hieraan doen? Extra laten betalen?

- Verander de richting van één van de ingangen om er voor te zorgen dat er minder rijen zijn.

- De simulator kan alleen voor de huidige parkeergarage gebruikt worden, pas de simulator aan zodat de inrichting van de parkeergarage veranderd kan worden.

- Je eigen verbeteringen?

Inleveren en beoordeling

Rapport

Inhoud

Het rapport moet ten minste de volgende onderdelen bevatten:

- Naam van het project, namen van de teamleden, inhoudsopgave en bladnummering.
- Een beschrijving van de fouten en problemen die nog in de applicatie aanwezig zijn.
- Een beschrijving van het test-proces (black-box, white-box en/of regressie testen)
- UML: Uitgebreide klasse diagrammen (inclusief methoden, instantie variabelen, scope, stereotypes en static) en de belangrijkste sequentie diagrammen.
- Een beschrijving van de belangrijkste uitbreidingen.
- Een beschrijving van de manier waarop de functionaliteit is geïmplementeerd, wat zijn de belangrijkste eigenschappen van de implementatie, gebruik termen zoals cohesie, encapsulation, koppeling en ontwerp op basis van verantwoordelijkheden.
- Leg uit welke verbeteringen je hebt bereikt door de code te refactoren.
- Een lijst met de bijdrage van de verschillende teamleden en een persoonlijke reflectie op het project.

Lay-out

Schrijf het rapport zodanig dat het als eindrapport opgestuurd kan worden naar een (fictieve) product owner (controleer de lay-out, structuur en vanuit welk perspectief het is geschreven). De beste manier om dit te doen om je in te leven in de volgende situatie: samen met je projectgroep vormen jullie een klein ICT-bedrijf ***Naam*** die ingehuurd is door de product owner om een simulatie programma te maken voor de parkeergarage. De hoofdstukken met de persoonlijke reflectie horen in de bijlages, omdat je deze natuurlijk niet naar de product owner gaat opsturen.

Het eindrapport moet uiterlijk om 15:00 de dag voor de presentatie worden ingeleverd.

Het project zal worden afgesloten met een presentatie waarin een korte demonstratie van het product zal worden gegeven.

Code

Overleg met je docent over de manier waarop je de code moet inleveren (dropbox, git toegang, papier). De code zal beoordeeld worden op de volgende punten:

- Correctheid.
- Juiste taal-constructies.
- Style (commentaar, indentation, naamgeving van de variabelen.
- Technisch niveau van de functionaliteit.

- Originaliteit en creativiteit.
- Correct gebruik van het MVC-patroon.
- Logische package structuur
- Is de Javadoc op de juiste plaats toegevoegd. (Genereer de HTML bladzijdes die de Javadoc weergeven)

Presentatie

De presentaties zijn gepland in week 9 van het blok. De presentaties zullen gegeven worden voor de hele (practicum)klas en twee docenten. De presentaties moeten ongeveer 30 minuten duren, inclusief 10 minuten voor vragen/discussie. De presentatie en het rapport worden door het hele team gemaakt, maar elk teamlid moet bekend zijn met de inhoud van het project.

Beoordeling

- De beoordeling begint bij een 1.
- Je kunt maximaal 3 punten behalen op het rapport en de presentatie.
- Als alle functionaliteit die in dit document beschreven staat gerealiseerd wordt kun je maximaal 4 punten extra scoren.
- Door extra uitbreidingen en functionaliteit te implementeren kunnen nog eens 2 punten gescoord worden.
- Elk team moet een peer review uitvoeren die een individuele bonus/malus score oplevert.
- De docent kan een individuele bonus/malus toekennen op basis van persoonlijke observaties of een afsluitend gesprek.
- Het eindresultaat moet minimaal een 5.5 zijn.

Herkansing

Als het behaalde resultaat voor het project en de presentatie ten minste een 4.5 maar minder dan een 5.5 is, zal de project groep de kans krijgen om hun product te verbeteren. Dit moet voor het einde van de eerste week van het volgende blok afgerond worden.