

1 Object-georiënteerd Analyseren, Ontwerpen en Implementeren in UML

In de vorige opdrachten heb je kennis gemaakt met verschillende programmeerconcepten in JAVA. De nadruk in deze opdrachten lag vooral op het samenstellen van systemen op basis van eerder ontworpen en geïmplementeerde componenten en wel op zodanige wijze dat het aan elkaar koppelen van onderdelen niet leidde tot rigoureuze aanpassingen van deze componenten zelf. Abstractie via interfaces speelde hierbij een sleutelrol. In deze opdracht ga je de geleerde programmeertechnieken wederom toepassen, maar pas nadat je een op informele wijze beschreven casus hebt *geanalyseerd* en hiervoor een systeem hebt *ontworpen*. Hierbij ga je de analyse-/ontwerptaal **UML** gebruiken. Daarna ga je op basis van dit ontwerp het systeem *implementeren*.

2 Leerdoelen

In deze opgave vragen we je om een casus te analyseren en uit te werken tot een detailontwerp door in ieder geval de bijbehorende *klassediagrammen* te specificeren. Bovendien selecteer je uit je ontworpen systeem enkele onderdelen waarvan je een *sequence diagram* en een *activity diagram* aangeeft.

Na afloop van deze opdracht ben je in staat om:

- een informeel beschreven systeem van een uitwerken tot een detailontwerp;
- een functioneel en technisch ontwerp uit te werken met behulp van UML;
- de verschillende UML-diagrammen kennen en kunnen toepassen voor het specificeren van de onderdelen van het ontwerp.
- Een gemaakt UML-ontwerp om te zetten/uit te breiden naar een werkende applicatie;
- Ontwerpfouten te lokaliseren en te verbeteren;
- Gegevens *persistent* te maken, hetgeen wil zeggen dat je een manier bedenkt om de gegevens om te slaan zodat ze bij de volgende uitvoering van je programma weer hergebruikt kunnen worden.

3 Instructie

Bestudeer de stof die op het college is besproken. Voor een uitgewerkt voorbeeld van een analyse, ontwerp en implementatie van een eenvoudig systeem zie b.v. <http://www.math-cs.gordon.edu/courses/cs211/ATMExample/>

Deelopdracht A: probleemschets

De 'ruwe casus' is dermate uitgebreid dat het vrijwel onmogelijk is om hiervoor binnen redelijke tijd de volledige 'analyse-ontwerp-implementatie' cyclus te doorlopen. Normaal gesproken zou je op basis van een eerste analyse van de casus een tijdsplanning opstellen. In deze opdracht ga je echter anders te werk, namelijk, je gaat allereerst een deelcasus bedenken waarvan je verwacht dat deze binnen 1 week gerealiseerd kan worden. Het is hierbij zaak om je eigen mogelijkheden en die van je practicumpartner goed in te schatten.

1. Reduceer de casus tot een behapbaar probleem. Geef de globale beschrijving van de onderdelen die je verder gaat uitwerken. Schroom niet om flink te schrappen! Naar alle waarschijnlijkheid ga je geen webapplicatie bouwen maar gewoon een JAVA programma. En voor de opslag van gegevens gebruik je geen 'echte database' maar een eenvoudige rij-achtige datastructuur. Misschien laat je ook de beheerder achterwege. Je uiteindelijke systeem dient wel minimaal de winkel en de

klanten te bevatten, waarbij de klanten in staat moeten zijn om producten te kopen en te koop aan te bieden.

2. Geef een volledig klassediagram waar alle gebruikte klassen in beschreven worden.
3. Specificeer tenminste één sequence diagram.
4. Specificeer tenminste één activity diagram.

Deelopdracht B: probleemschets

Je probeert bij de implementatie zo nauwkeurig mogelijk je eigen ontwerpbeslissingen op te volgen. Dit houdt onder andere in dat je gebruik probeert te maken van de mogelijkheid in Visual Paradigm (of, als je daar de voorkeur aan geeft, een ander UML-tool) om uit je UML-ontwerp direct JAVA code te genereren om deze vervolgens te completeren tot een geheel werkende applicatie.

Waar de casusbeschrijving zich niet over uit laat, is het gebruikersinterface. *Grafische Gebruikersinterfaces* (G(raphical U(ser) I(nterface)s) komen volgende week tijdens het college aan de orde. Maar voor deze opdracht verwachten we niet dat je jouw programma van een geavanceerde GUI voorziet. Een simpel op tekst gebaseerd interface (gebruik makende van `System.out/ System.in`) volstaat.

Om gegevens niet aan het einde van iedere programma-uitvoering kwijt te raken, moeten deze op de een of andere manier worden opgeslagen. Het ligt hier misschien wel voor de hand om de gegevens op te slaan in een *database*. Voor het beheer van een database (toevoegen, opvragen en wijzigen van informatie) wordt vaak gebruik gemaakt van een zogenaamd *Database Management System*. Echter, het heeft nogal wat voeten in aarde om vanuit JAVA een verbinding te maken met een database, en door middel van geschikte *queries* gegevens op te halen uit of te veranderen in deze database, zie o.a. *JHTP* hoofdstuk 24. Om niet nodeloos veel tijd in dit onderdeel te hoeven steken, stellen we voor dat je voor de opslag van data gebruik te maakt van zogenaamde *ObjectStreams* (*JHTP* hoofdstuk 15.5). In feite hoeft je van de klassen van objecten die je wil wegschrijven alleen aan te geven dat ze het *Serializable*-interface (uit `java.io`) implementeren. Dit interface is een wat vreemde eend in de bijt: het bevat geen enkele abstracte methode. Dus hoeft je ook verder niets aan je klasse toe te voegen om het te implementeren. Voor het inlezen en wegschrijven van objecten kun je resp. de methodes `readObject` en `writeObject` uit de klassen `ObjectInputStream` en `ObjectOutputStream` gebruiken. Om dit te illustreren vind je op **BB** een aantal klassen die je ook als basis voor je eigen implementatie kunt gebruiken.

4 Producten

4.1 Deelopdracht A

Ook bij deze opdracht ben je vrij je eigen ontwikkelomgeving te kiezen, maar raden we je sterk aan om voor het UML gedeelte de tool *Visual Paradigm* te gebruiken. Om te voorkomen dat de assistenten allerlei software moeten installeren bij het beoordelen van de ingeleverde uitwerkingen vragen we je om van alle producten *screenshots* te maken, deze screenshots in één pdf-document te verwerken en dit vervolgens in te leveren via Blackboard.

4.2 Deelopdracht B

Voor dit onderdeel moet je als producten een JAVA applicatie, zoals gebruikelijk bestaande uit een verzameling klassedefinities, inleveren, die aan het voorgegeven ontwerp voldoet alsook aan de kwaliteitscriteria zoals die gesteld zijn voor deze cursus.

5 Inleveren van je producten

Vóór zondag 19 april, 11:00 uur, via Blackboard. Lever het pdf-document uit onderdeel A en *alle* .java files uit onderdeel B in.