Toepassingsopdracht

Tom Hofkens

2018 – 2019

1 Opgave

1.1 De probleemstelling

De scholengemeenschap Kontich-Hove schrijft al jaren software voor de scholen van de scholengemeenschap. Het puntenprogramma moet dringend herschreven worden (o.a. door de opkomst van co-teaching) en daarvoor is jullie hulp nodig.

Een of meerdere leerkrachten geven les over een bepaald vak aan een klas van leerlingen. Dit gebeurt a.h.v. toetsen. Een toets is een verzameling van punten met een bepaald maximum voor een groep van leerlingen. Een verzameling van toetsen wordt gebundeld in een puntenlijst. Een puntenlijst heeft een volgnummer in het jaar zodat alle puntenlijsten met een bepaald volgnummer gebruikt kunnen worden om een tussentijds maandrapport te geven. Alle toetsen van een periode (bv trimester of semester) vormen samen de punten voor dagelijks werk. Er zijn ook toetsen tijdens de examenperiode die samen de punten voor het examen vormen. De punten DW en EX worden in een bepaalde verhouding opgeteld om een totaal per vak te vormen. Op basis van het gewicht van elk vak (typisch volgens het aantal uren dat het gegeven wordt) kan er ook op elk rapport een algemeen totaal bepaald worden. In het voorbeeldbestand system.txt vind je een uitgewerkt voorbeeld dat ook in de slides aan bod komt. Om leerkrachten vlot te laten werken wordt er een undo functie toegevoegd.

1.2 Analyse

De volgende objecten dienen als bouwstenen voor de oplossing. Stel voor elk een ADT op. Een id is gewoon een uniek getal dat begint bij 1.

- Leraar: krijgt een vijfletterige unieke naamcode (bv HOFKT), een voornaam en een achternaam.
- Leerling: krijgt een zevencijferige unieke stamboeknummer (bv 1800001), een voornaam, een achternaam, een klas en een klasnummer.
- Toets: heeft een unieke naam zonder spaties, een maximum en een verzameling van punten. Zoeksleutel: naam.
- Punt: heeft een id, een stamboeknummer van een leerling, een naam van een toets en een timestamp. Zoeksleutel: id.
- Puntenlijst: heeft een id, een type (M voor maand, E voor examen), een periode (1, 2, ...), een of meerdere naamcodes, een vakcode (bv WIS), een klas, aantal uren en bevat een aantal toetsen.
- Rapport: bevat alle puntenlijsten van iedereen. Zoeksleutel: gebruik hier een samengestelde zoeksleutel, nl. type gevolgd door de periode (dus bv M1 (= alle punten van het eerste maandrapport) of E1 (= alle punten van de eerste examenperiode)). Merk op dat de zoeksleutel hier niet langer uniek is. Dat probleem zal je dus moeten oplossen.

1.3 Ontwerp

1.3.1 ADT Queue

Meerdere leerkrachten kunnen voor eenzelfde leerling punten ingeven voor hetzelfde vak. De punten komen binnen en moeten verwerkt worden volgens het FIFO principe. Gebruik een queue voor die punten. Indien iemand een priority queue implementeert (zie verder), gebruik je als priority de timestamp.

1.3.2 ADT Stack

De undo-functionaliteit wordt geïmplementeerd d.m.v. een stack. In het inputbestand staan er verschillende commando's na het start commando. Enkel die commando's worden gebruikt in de undo. Het volstaat om het commando (met parameters) op de stack te zetten. Merk op dat je dus per gebruiker een afzonderlijke stack moet bijhouden. Bij een undo operatie wordt de top van de stack gehaald en het commando dat op de top stond wordt ongedaan gemaakt.

1.3.3 ADT Tabellen

De toets, de puntenlijst en het rapport worden ontworpen volgens het ADT Tabel.

1.4 Implementatie

1.4.1 ADT Tabel via een circulair gelinkte ketting

Alle toetsen worden bijgehouden in een tabel die geïmplementeerd wordt a.h.v. een circulair gelinkte ketting (ketting implementatie, geen array implementatie).

1.4.2 ADT Tabel via een dubbelgelinkte ketting

De punten worden bijgehouden in een tabel die geïmplementeerd wordt a.h.v. een dubbelgelinkte ketting (ketting implementatie, geen array implementatie). Buiten de ADT bewerkingen voorzie je ook een sort op de zoeksleutel (kies zelf een algoritme uit de cursus). Let op dat je dus geen ADT gesorteerde lijst moet implementeren, maar een ketting die de mogelijkheid heeft zichzelf te sorteren.

1.4.3 ADT Tabel via binaire zoekboom

De puntenlijsten van het rapport worden bijgehouden in een tabel die geïmplementeerd wordt a.h.v. een binaire zoekboom met een samengestelde zoeksleutel (type+periode). Wat betreft traversals moet je enkel de inorder traversal implementeren. De methodes AttachLeftSubtree, DetachLeftSubtree (en analoog voor Right) moet je niet implementeren.

1.4.4 Geavanceerde tabellen

Het moet mogelijk zijn om de huidige tabelimplementaties (van alle bovenstaande tabellen dus) te vervangen door andere tabelimplementaties. Binnen je groep zal je verschillende geavanceerde tabelimplementaties (1 per student) moeten maken (zie opdracht 5). Om die uitwisselbaar te maken met de andere tabelimplementaties zal je wrappers moeten schrijven. Dat zijn klasses die het contract van de ene klasse koppelen aan die van een andere klasse. Het is dus de bedoeling dat je de geavanceerde tabelimplementaties met de specifieke ADT bewerkingen uit de cursus gebruikt.

Een voorbeeld van een wrapper vind je in de cursus bij de implementatie van het ADT Stack m.b.v. het ADT Lijst (bij Implementaties van het ADT Stack). Daar staat in het contract bijvoorbeeld de ADT operatie pop, maar intern wordt de ADT operatie remove van het ADT Lijst gebruikt. Kijk zeker ook in de slides van de toepassingsopdracht voor een voorbeeld.

2 Opdrachten

2.1 Algemeen

- Als je een contract opstelt, doe dat dan voor alle methodes met daarin
  - de beschrijving van de methode,
  - de input parameter(s)
  - de output parameter(s)
  - de pre- en postcondities.
- Hou je aan de contracten van de cursus. Gebruik ook de benamingen uit de cursus.
- Stel je contracten op in plain text bestanden. Zo kan je makkelijk aanpassen en gegevens uitwisselen.
- Voor de implementatie gebruik je **Python 3.x** die geïnstalleerd is onder Ubuntu (op de pc's van het departement). Test dat op voorhand zeker uit. Als het daar niet werkt, dan kan je niet deelnemen aan het examen en krijg je een afwezig.
- Probeer ervoor te zorgen dat je ADT's werken, onafhankelijk van het type waarvoor je ze momenteel implementeert.
- Bij elke implementatie horen tests die aantonen dat je code werkt. Stel een aantal goed gekozen tests op waarbij je ook de pre- en postcondities test. Denk aan de zwakke plekken van je ADT's, maar probeer zoveel mogelijk extremen te testen.
- Het is de bedoeling dat je de ADT's zo algemeen mogelijk houdt en die gaat gebruiken in je systeem. Het is bijvoorbeeld niet de bedoeling om een stack te schrijven specifiek voor deze applicatie. Je implementeert een stack volgens het contract van de cursus en gebruikt die in je systeem.
- Hoe je de ADT's implementeert, kies je zelf, **tenzij het anders vermeld staat in de opdracht**. Je mag bijvoorbeeld zelf beslissen om een array implementatie te gebruiken of een ketting implementatie als er niets gespecifieerd is.
- Bij alle boomimplementaties gaan we bij het verwijderen telkens op zoek naar de inorder successor (dus nooit de predecessor).
- In de cursus staat er duidelijk welke parameters er dienen voor input en welke voor output. In Python gebruik je functieparameters voor input en return values voor outputparameters. Als er meerdere outputparameters zijn, gebruik je een tuple.

Om je systeem vlot te kunnen testen (zowel in groep als op het examen), gebruik je input- en outputbestanden met een vast formaat.

Voor de ADT implementaties voorzie je een inputbestand dat een script bevat met commando's. Dat bestand moet ingelezen worden en alle commando's moeten uitgevoerd worden. **Zonder deze functionaliteit kan je geen (individuele) punten krijgen**. Op Blackboard vind je een voorbeeldbestand. Als output gebruik je de **dot language** (wikipedia,graphviz) zodat je makkelijk een grafische voorstelling kan maken van je implementatie.

Voor het uiteindelijke systeem voorzie je eveneens een script voorafgegaan door een initialisatiemogelijkheid. Dit bestand mag je eventueel zelf uitbreiden moest dat nodig zijn. Wat er in het voorbeeldbestand staat, moet je zeker voorzien. Als output wordt er een statusrapport gegenereerd in html. Op Blackboard vind je voorbeeldbestanden voor zowel input als output. Ook zonder deze functionaliteit kan je geen (groeps-) punten krijgen.

2.2 Opdracht 1 (individueel)

Elk groepslid stelt contracten op (volgens de cursus!) van drie eenvoudige ADT's (zie hieronder) en maakt twee ketting implementaties:

1. Je kiest uit een stack en een queue.
2. Je kiest uit een circulair gelinkte ketting en een dubbel gelinkte ketting.
3. Een binaire zoekboom.

Ieder groepslid maakt een binaire zoekboom (tijdens de lessen Inleiding Programmeren). In totaal moet je dus drie ketting implementaties hebben en drie contracten. Zorg er wel voor dat binnen je groep de vijf (stack, queue, circulair gelinkte ketting, dubbel gelinkte ketting en binaire zoekboom) aan bod komen, want je zal ze allemaal nodig hebben. Spreek dat dus goed af.

2.3 Opdracht 2 (individueel)

Elk groepslid stelt een ontwerp op voor het hele systeem, d.w.z. een contract voor **elke** klasse en methode (zonder de geavanceerde tabelimplementaties, maar met de tabellen d.w.z. je stelt wel een contract op voor het ADT tabel om de punten, toetsen en puntenlijsten bij te houden, maar nog niet voor de 2-3-boom, 2-3-4-boom, rood-zwart-boom of hashmap). Identificeer alle klassen en de bewerkingen die je nodig hebt volgens de opgave. Dit zijn enerzijds de opgegeven klassen (op p.1) en anderzijds de klassen waarvan je zelf vindt dat die noodzakelijk zijn.

2.4 Opdracht 3 (individueel)

Op Blackboard vind je het feedbackformulier GSFeedbackDesign.docx. Binnen jullie groepje gaat ieder het ontwerp van 1 ander grondig evalueren en van feedback voorzien. Het is voldoende dat je 1 keer het feedbackformulier invult, niet 1 keer per ADT.

2.5 Opdracht 4 (groep)

Stel een nieuw contract op in groep op basis van de opgestelde contracten en de feedback. Je bespreekt met de groep alle ontwerpen en kiest telkens de beste oplossing voor elk ADT. Je komt uiteindelijk tot een nieuwe versie (versie 2). Noteer in het ontwerp bij elke methode en klasse de verantwoordelijkheden: wie het gaat implementeren en wie het gaat testen.

2.6 Opdracht 5 (individueel)

Per groep ga je een aantal (typisch 1 per student, maar meer mag ook) geavanceerde gegevensstructuren ontwerpen en implementeren aan de hand van de cursus. Spreek af binnen je groep wie er welke gegevensstructuur gaat ontwerpen en implementeren. Voor deze opdracht is het voldoende om een contract op te stellen, de implementatie volgt in opdracht 7.

Je kan kiezen uit:

- een 2-3 boom,
- een 2-3-4 boom (maar niet a.h.v. rood-zwart bomen),
- een rood-zwart boom,
- een hashmap die instelbaar via een parameter kan switchen tussen een hashmap met linear probing, quadratic probing en een hashmap met separate chaining (zorg ervoor dat de gebruikte implementatie voor seperate chaining makkelijk kan aangepast worden, begin bijvoorbeeld met de dubbel gelinkte ketting, maar zorg dat die makkelijk kan aangepast worden naar een andere implementatie).

De rood-zwart boom is de moeilijkste. Als er niemand in de groep dit ziet zitten, kan je opteren om de heap te implementeren (ketting implementatie). Dit is uiteraard geen tabelimplementatie maar kan de queue uit je systeem vervangen. Dit heeft wel gevolgen voor de evaluatie (zie 5.3). Ook de 2-3-4-boom is niet eenvoudig. Indien er niemand in de groep het ziet zitten om de 2-3-4-boom te implementeren, kan je opteren om die samen te doen (een student implementeert de insert en een andere de delete). Ook dit heeft gevolgen voor de evaluatie.

2.7 Opdracht 6 (individueel)

Elk groepslid implementeert een reeks testen gebaseerd op het ontwerp van 1 collega (op basis van opdracht 5). Alle ontwerpen moeten aan bod komen, dus spreek goed af wie wat doet.

2.8 Opdracht 7 (individueel)

Deze opdracht bouwt voort op opdracht 5. Implementeer (individueel) aan de hand van je ontwerp een eerste versie van je geavanceerde gegevensstructuur. Gebruik de testen uit opdracht 6 om je implementatie te controleren.

2.9 Opdracht 8 (groep)

Implementeer het hele systeem op basis van de tweede versie van het ontwerp. Let op: het resultaat van deze opdracht is een werkende eindversie waarbij het hele contract geïmplementeerd moet zijn. Voorzie ook voldoende tests.

3 Planning

Als je iets moet insturen is dat telkens ten laatste op zondag **vóór 22.00u**. Insturen doe je via Blackboard. De laatste deadline is heel belangrijk. Code die te laat ingediend wordt, zal verwijderd worden van Blackboard en kan je niet gebruiken op het examen.

week	onderwerp	
2	Implementatie sorteeralgoritmes.	contact-moment
3	Oefeningen reeks 1.	contact-moment
4	Oefeningen reeks 2. Uitleg opgave toepassingsopdracht Theorie OOAD	contact-moment
5	Oefeningen reeks 3.	contact-moment
6	Oefeningen reeks 4.*Maak een groep op Blackboard (via communicatie > groepen of via "Inschrijving in groepen") met de* ***namen van alle groepsleden in de naam van de groep*** *tegen vrijdag.*	contact-moment
7	Midterm examen	
8	Opdracht 1 en 2 insturen op Blackboard tegen dinsdag. Tijdens dit contactmoment worden belangrijke afspraken gemaakt (wie welke stukken van het ontwerp implementeert, wie er welke geavanceerde gegevensstructuur implementeert en wie het zal testen zie opdrachten 5, 6 en 7). **Aanwezigheid is verplicht en er zullen absenties worden genomen.** Opdracht 3 en 4 insturen op Blackboard na/tijdens de les van woensdag, ten laatste vrijdag.	verplicht contact-moment
9	Opdracht 5 insturen op Blackboard tegen zondag.	
10	Opdracht 6 insturen op Blackboard tegen zondag.	
11		
12	Opdracht 7 insturen op Blackboard tegen zondag.	
13	Opdracht 8 insturen op Blackboard tegen zondag.	
voor het examen	*Voorbereiding: bekijk de algoritmes uit de cursus van de verschillende geavanceerde gegevensstructuren uit opdracht 5 nog eens zodat je daar geen tijd mee verliest.* Je komt samen met je groep om de code van mekaar te leren kennen. Hier worden de testen uitgeprobeerd en stelt elk groepslid de eigen code voor waarbij je extra aandacht besteedt aan de manier waarop je de algoritmes geïmplementeerd hebt. Dit is een belangrijke voorbereiding voor het mondelinge examen. Let op: je kan geen aanpassingen meer doen aan je code! Het examen gebeurt met de code ingediend in week 13.	je kiest zelf wanneer je samenkomt met je groep
examen		contact-moment op afspraak

Enkel wanneer er contactmoment staat ben je verplicht aanwezig en is de assistent aanwezig. De andere momenten kan je zelfstandig of in groep werken in het lokaal (vraag desnoods de sleutel op het secretariaat). Het lokaal is gereserveerd voor jullie.

4 Python tips

4.1 Arrays in Python

Python werkt met lists i.p.v. arrays. Om een array van n elementen te simuleren met een list, initialiseer je de list met n keer het element None waarna die zich in de rest van het programma statisch gedraagt (hij mag vanaf dan niet meer groeien bijvoorbeeld). Gebruik dus geen append of andere specifieke list methodes.

4.2 Samenwerken in Python

Samenwerken in Python aan dezelfde klasse kan makkelijk als volgt:
in Node.py:

```
import Node1
import Node2

class Node:
    __init__ = Node1.__init__

    __str__ = Node2.__str__
```

in Node1.py:

```
def __init__(self, item=None, next=None):
    self.item = item
    self.next = next
```

in Node2.py:

```
def __str__(self):
    output = str(self.item)
    if self.next != None:
        output += ", " + str(self.next)
    return output
```

Zo kan je makkelijk samenwerken in een gedeelde map via Dropbox/Google Drive/Box/.... Je mag uiteraard ook samenwerken met GitHub/Bitbucket/..., maar dat is absoluut geen vereiste. Dat komt aan bod in de verdere opleiding.

5 Evaluatie

Voor de doelstellingen verwijzen we naar de cursusinformatie op Blackboard.

De toepassingsopdracht staat op 10 punten. Tijdens het mondelinge examen kom je met de hele groep langs voor een verdediging. Er is geen tijd voorzien voor een presentatie, maar je toont wel een korte demo van max. 2 minuten waarbij je het inputbestand dat op Blackboardstaat inleest en de nodige output genereert. Er staan geen punten op het opstellen van een (G)UI.

5.1 GUI

Een grafische user interface valt buiten de scope van deze cursus en zal geen punten opleveren. Een GUI kan echter een grote meerwaarde zijn om later bij grotere projecten samen te werken met een klant die over het algemeen weinig kennis heeft over een informaticasysteem. Om snel te weten te komen of wat er geïmplementeerd wordt ook voldoet aan de verwachtingen van de klant, is een goede GUI onontbeerlijk. We laten jullie dan ook vrij om die alsnog toe te voegen (bijvoorbeeld in de periode tijdens de examens). Groepen met een goede GUI komen in aanmerking

om als ambassadeurs van de UAntwerpen uitgenodigd te worden om bijvoorbeeld in de vroegere secundaire school een voorstelling te geven of op openlesdagen.

5.2 Groepswerk op 5 punten

5 punten bestaan uit een groepscijfer geïndividualiseerd aan de hand van peerevaluatie. Hier wordt gekeken naar het ontwerp van je software. Een goed ontwerp is aanpasbaar aan nieuwe wensen van de gebruiker. Tijdens de verdediging zal je (elk lid van de groep) heel **snel aanpassingen moeten doen** in de code dus 'program to change' is het motto. Probeer daarom te werken volgens de OOAD principes die in het eerste hoofdstuk van het handboek staan. Aangezien dit een groepscijfer is, heb je er dus alle baat bij dat alle groepsleden over alles kunnen meepraten. De klemtoon ligt hier dus op goede samenwerking, goede afspraken en goede contracten.

5.3 Individueel werk op 5 punten

De overige 5 punten zijn individueel. Tijdens de eindpresentatie krijg je vragen over de **implementatie van een geavanceerde gegevensstructuur die je zelf geïmplementeerd** hebt en **over een ADT dat een van je medestudenten geïmplementeerd** heeft (je weet op voorhand niet welk ADT dat gaat zijn). Zorg dat je weet hoe de code van je medestudenten werkt. Voorzie dan ook voldoende tijd (in week 13) om samen de code te doorlopen. Typische vragen die je hier kan verwachten zijn: hoe werkt het algoritme volgens de cursus? En hoe is het hier geïmplementeerd? Je kan ook theorievragen krijgen toegepast op jullie project (bijvoorbeeld welke efficiëntie heeft het sorteeralgoritme dat jullie gebruikt hebben? Waarom is dat van die orde?).

Bij het individueel werk hoort uiteraard ook een geavanceerde tabelimplementatie. Sommige implementaties zijn moeilijker dan andere. In het bepalen van het individueel cijfer worden moeilijkere implementaties (2-3-4 en rood-zwart) minder streng beoordeeld dan de eenvoudigere (2-3-boom en hashmap). De rood-zwart-boom is de moeilijkste. Enkel als je die implementeert, kan je het maximum van de punten halen. In de volgende tabel kan je zien welke punten je kan halen bij welk ADT. Je mag ook meerdere ADT's implementeren om meer punten te halen. Elke lijn van onderstaande tabel mag geïmplementeerd worden door een of meerdere studenten en alle combinaties zijn mogelijk. Je kan niet meer dan 5/5 halen.

ADT	**/5**
heap	2.5
hashmap	2.5
heap en hashmap	3
2-3-boom	3
2-3-4-boom insert	2
2-3-4-boom delete	2.5
rood-zwart-boom insert	3
rood-zwart-boom delete	3.5

5.4 Portfolio indienen via Blackboard

Op Blackboard zijn er opdrachten binnen de Toepassingsopdracht. Daar zet je de volgende bestanden die samen je groepsportfolio vormen tegen de deadline (zie planning):

- contracten en implementatie van een stack of queue, een dubbel gelinkte of circulaire ketting en een binaire zoekboom van elk groepslid (opdracht 1),
- contract van het systeem versie 1 van elk groepslid (opdracht 2),
- evaluatiefiche van contract van elk groepslid (opdracht 3),

- contract van het systeem versie 2 (1 voor de hele groep) (opdracht 4),
- contract van geavanceerde gegevensstructuur van elk groepslid (opdracht 5)
- tests van geavanceerde gegevensstructuur van elk groepslid (opdracht 6)
- implementatie van geavanceerde gegevensstructuur van elk groepslid (opdracht 7)
- implementatie van het systeem versie 1 (opdracht 8)

De volledigheid van je portfolio kan je eindresultaat van het vak in positieve of negatieve zin beïnvloeden.

5.5 Groepsverdeling

Maak een groep van 4 studenten. Maak binnen de cursus een groep aan op Blackboard(via communicatie > groepen) en kies als naam een opeenvolging van de namen van alle groepsleden. Zorg dat al je groepsleden daar in zitten en beperk de toegang voor anderen door het maximum aantal gelijk te stellen aan het aantal groepsleden. Als je op zoek bent naar een groep, kan je je inschrijven in de groep "Op zoek naar een groep". Je kan zelf zien wie er nog allemaal in die groep zit en op die manier afspreken.

5.6 Vragen

Voor vragen over Gegevensabstractie en -structuren kan je steeds terecht bij de assistent (Tom Hofkens - tom.hofkens@uantwerpen.be).