Toepassingsopdracht Gegevensabstractie en -structuren 2014-2015

1 Opgave

1.1 De probleemstelling

De Kinepolis groep is op zoek naar een nieuw reservatiesysteem. Gebruikers kunnen via jullie programma ticketten reserveren voor een bepaalde vertoning. Filmvertoningen beginnen pas op vastgestelde tijdsslots (14.30u, 17.00u, 20.00u, 22.30u), kunnen variëren van dag tot dag en gaan door in een zaal van het complex. Om hun service te verhogen naar hun gebruikers toe, zal een film pas beginnen als alle personen die een ticket gereserveerd hebben, aanwezig zijn.

1.2 Analyse

De volgende objecten dienen als bouwstenen voor de oplossing. Stel voor elk een ADT op.

- Zaal: bevat een nummer en het aantal plaatsen. Zoeksleutel: nummer.
- Vertoning: bevat een id, een zaalnummer, een slot , een datum, een filmid, aantal vrije plaatsen. Zoeksleutel: id.
- Film: bevat een id, een titel en een rating (een float). Zoeksleutel: id.
- Reservatie: bevat een id, een userid, een timestamp (tijd en datum), een vertoningid en een aantal plaatsen die gereserveerd worden. Blijft altijd bewaard. Zoeksleutel: id.
- Gebruiker: bevat een id, een voornaam, een achternaam en een e-mailadres. Zoeksleutel: id.

Voor volgende objecten hoef je geen ADT op te stellen.

- Slot: tijdens een slot kan een film getoond worden. Mogelijke waarden zijn: 14.30u, 17.00u, 20.00u, 22.30u.
- Ticket: bevat een gebruikerid.

1.3 Ontwerp

1.3.1 ADT Queue

De reservaties komen binnen en moeten verwerkt worden volgens het FIFO principe. Schrijf een queue voor die reservaties.

1.3.2 ADT Stack

Hou per vertoning een stack bij. Als een reservatie voor *n* plaatsen verwerkt wordt, plaatst het systeem *n* tickets in de stack van de voorstelling. Telkens een gebruiker de zaal binnenkomt, wordt er een ticket van de stack gepopt. Enkel indien de stack leeg is, begint de film. Als de stack vol zit, kunnen er geen reservaties meer gebeuren. Reservaties worden nooit half afgewerkt. Als er nog 2 plaatsen vrij zijn en er komt een reservatie voor 4 ticketten, dan wordt de reservatie geweigerd.

1.3.3 ADT Tabellen

De vertoningen en films worden ontworpen volgens het ADT Tabel. Om de films te doorlopen moet er een inorder traversal gebruikt worden.

1.4 Implementatie

1.4.1 ADT Tabel via gelinkte ketting

De vertoningen worden bijgehouden in een tabel die geïmplementeerd wordt a.h.v. een gelinkte lijst (linkbased implementatie). Buiten de ADT bewerkingen voorzie je ook een sort (kies zelf een algoritme uit de cursus). Let op dat je dus geen gesorteerde lijst moet implementeren, maar een ketting die de mogelijkheid heeft zichzelf te sorteren.
Je sorteert daarbij op datum, slot en zaal.

1.4.2 ADT Tabel via binaire zoekboom

De films worden bijgehouden in een tabel die geïmplementeerd wordt a.h.v. een binaire zoekboom met zoeksleutel de rating. Je moet enkel de inorder traversal implementeren. De methodes AttachLeftSubtree, DetachLeftSubtree (en analoog voor Right) moet je niet implementeren.

1.4.3 Geavanceerde tabellen

Het moet mogelijk zijn om de huidige tabelimplementaties te vervangen door andere tabelimplementaties. Binnen je groep zal je 4 geavanceerde tabelimplementaties 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 (slide 5 van 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.

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 parameters
  - 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**. 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 linkbased implementatie.

2.2 Opdracht 1

Elk groepslid maakt twee linkbased implementaties:

1. Je kiest uit een stack en een queue.
2. Je kiest uit een binaire zoekboom en een dubbelgelinkte ketting.

Zorg er wel voor dat binnen je groep de vier (stack, queue, binaire zoekboom en dubbelgelinkte ketting) aan bod komen, want je zal ze allemaal nodig hebben.

2.3 Opdracht 2

Elk groepslid stelt een ontwerp op voor het hele systeem, d.w.z. een contract voor elke klasse en methode. Identificeer alle klasses en de bewerkingen die je nodig hebt volgens de opgave.

2.4 Opdracht 3

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

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 testen.

2.6 Opdracht 5

Per groep ga je een aantal (1 per student) 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).

2.7 Opdracht 6

Elk groepslid implementeert een reeks testen gebaseerd op het ontwerp van 1 collega. Alle ontwerpen moeten aan bod komen.

2.8 Opdracht 7

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

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. Zorg voor output nadat er iets gebeurd is zodat je het systeem makkelijk kan testen. Voorzie ook voldoende tests.

2.10 Opdracht 9

Test de code van je groepsgenoten. Schrijf een reflectie (individueel of in groep) waarbij je aangeeft welke wijzigingen je nog zou kunnen/willen doen aan je systeem.

2.11 Opdracht 10

Implementatie van het systeem versie 2. Dit is opnieuw een eindversie, maar met minder bugs, extra functionaliteit, ideeën die je opgedaan hebt, ...

3 Planning

Als je iets moet insturen is dat telkens ten laatste op **<u>vrijdag vóór 23u</u>** van de week waarin de opdracht staat. Insturen doe je via BB.

week	onderwerp	
5	Oefeningen reeks 1. Uitleg opgave toepassingsopdracht.	contactmoment
6	Oefeningen reeks 2. *Voorbereiding: Lees de opgave van de toepassingsopdracht en het stuk over gelinkte kettingen in de cursus. Maak een groep op BB (via communicatie > groepen).* • vragen over opgave • les over wijzers in Python • uitleg wrappers OOAD principes	contactmoment
7	Examen.	
8	• Opdracht 1 insturen op BB. • Opdracht 2 insturen op BB. • Opdracht 3 insturen op BB.	
9	• 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). **<u>Aanwezigheid is verplicht en er zullen absenties worden genomen.</u>** • Opdracht 4 insturen op BB.	verplicht contactmoment
10		
11	Opdracht 5 insturen op BB.	
12	Opdracht 6 insturen op BB. Opdracht 7 insturen op BB.	
13	*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 tests 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. • Opdracht 8 insturen op BB. • Op het einde van deze bijeenkomst dien je opdracht 9 in op BB.	
examen	Opdracht 10 insturen op BB. Je stuurt alles in ten laatste de dag voor je verdediging.	contactmoment 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 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/....

5 Evaluatie

Voor de doelstellingen verwijzen we naar de cursusinformatie op BB.

De toepassingsopdracht staat op 10 punten. Om vlot te werken met je eigen software is een goede (G)UI handig, maar het is geen einddoel. Er staat in totaal 1 punt op de (G)UI. 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 2 minuten.

5.1 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 en goede afspraken.

5.2 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?).

5.3 Portfolio indienen via BB

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

- implementatie van 1 stack of queue en 1 dubbel gelinkte ketting of 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)
- reflectie per groepslid of 1 per groep (opdracht 9)
- implementatie van het systeem versie 2 (opdracht 10)

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

5.4 Groepsverdeling

Maak een groep van 4 studenten. Maak binnen de cursus een groep aan op BB (via communicatie > groepen) en zorg dat al je groepsleden daar in zitten. Beperk de toegang voor anderen door het maximum aantal gelijk te stellen aan het aantal groepsleden.

5.5 Vragen

Voor vragen over Gegevensabstractie en -structuren kan je steeds terecht bij de assistent (Tom Hofkens - tom.hofkens@uantwerpen.be). Voor typische Python vragen, moet je bij de assistenten van Inleiding Programmeren zijn.