Computertentamen Objectgeorienteerd Programmeren TI1205
6 november 2013 14.00-17.00

Bij dit tentamen mag je gebruik maken van:

- Java in Two Semesters (Charatan & Kans)
- De slides van de cursus, op papier of online beschikbaar via Blackboard
- De javadoc API documentatie van Java die via Blackboard te raadplegen is (Blackboard → TI1205 → Assignments → Java 1.6 API)

Dit tentamen bevat een **basis-opdracht (9 punten)** en **1 uitbreiding (1 punt)** (totale opdracht: 4 bladzijden).

Log in op de computer met volgende gegevens:
Login: ewi_ti1200
Paswoord: Welkom01

TIP: lees de opdracht eerst helemaal door en begin daarna pas met implementeren.

Een aantal **hints/tips**:

- Je programma moet compileren (niet compileren == niet slagen).
- Als je programma af is, zip je je bestanden. Maak een zip bestand van je **src** folder (waar je .java bestanden in staan). Geef dit zip bestand volgende naam: <studentennummer>.zip, dus bijvoorbeeld 12121212.zip. Stop ook het textbestand bestand met de invoer in dit zip bestand. Het is dus niet nodig om het hele Eclipse project in de zip te stoppen!
- Maak je werk in de "default package", maak geen aparte package aan (dit maakt het nakijkwerk eenvoudiger).
- Uploaden gebeurt via Blackboard → TI1205 → Assignments → Tentamen. Onder hetzelfde kopje kan je ook het textbestand downloaden.
- Je mag alle software aanwezig op de computer gebruiken.
- Het netwerk is uitgeschakeld op toegang tot Blackboard na.
- Mobiele telefoons blijven in je rugzak/jaszak en verschijnen *niet* op tafel. Je schakelt ze ook uit.
- Poging tot **fraude wordt bestraft**.

Basisopdracht (9 punten)

Amazon.nl wordt binnenkort gelanceerd. In het eerste jaar dat Amazon.nl op de markt is wil Amazon.nl zich toeleggen op boeken en muziek (zowel CDs als online muziek). Een uitbreiding van het gamma is op korte termijn op til (denk aan mobiele telefoons, computers, tablets, etc), net zoals de andere winkels van Amazon.

Er wordt aan jou gevraagd om de lancering van Amazon.nl voor te bereiden door de ruggengraat van de software voor Amazon.nl te schrijven. Amazon.nl houdt zijn gegevens bij in een file. Die ziet er als volgt uit (en mag zeker niet veranderd worden!):

```
BOEK Harry Mulisch, De Aanslag, 9023466349, Bezige Bij, 246, 2010, 19.95
BOEK Dan Brown, De Da Vinci Code, 9024562287, Luitingh, 432, 2009, 12.49
CD Foo Fighters, Wasting Light, 0886978449320, Sony, 2011, 11.95
MP3 Hooverphonic, Reflection, 0888837802826, Sony, 2013, 15.00, 165
```

De volledige file **amazon.txt** is beschikbaar via Blackboard.

De volgorde van de elementen in de file kan belangrijk zijn. De volgorde is dezelfde als in de beschrijving hieronder.

Een **BOEK** wordt gekenmerkt door:

- De auteur
- De titel
- Het 10-cijferig ISBN nummber
- De uitgever
- Het aantal bladzijden
- Het jaar van uitgave
- De prijs

Een **CD** wordt gekenmerkt door:

- De artiest/uitvoerder
- De titel
- De EAN code
- Het label
- Het jaar van uitgave
- De prijs

Een **MP3** wordt gekenmerkt door:

- De artiest/uitvoerder
- De titel
- De EAN code
- Het label
- Het jaar van uitgave
- De prijs
- De kwaliteit van de MP3 uitgedrukt in de bitrate

Nu vraagt Amazon.nl om een programmaatje te maken dat:

- De file amazon.txt die hierboven beschreven wordt kan **inlezen**
- Het toelaat om nieuwe boeken, CDs of MP3s toe te voegen aan de catalogus
- Optie (zie uitbreiding): Het toelaat om de producten in de catalogus te sorteren op 2 manieren, namelijk op:
    - Alfabetisch op auteur/artiest/uitvoerder (oftewel van A → Z))
    - Prijs (van laag naar hoog)
- Het toelaat om het bestand (met bijvoorbeeld een nieuw boek) **weg te schrijven naar file**.

- Om gebruikersinteractie mogelijk te maken is een tekstuele **command line interface** voldoende – te implementeren met System.out.*. Deze moet er als volgt uitzien:

```
Maak uw keuze:
   1 – Voeg een nieuwe boek toe
   2 – Voeg een nieuwe CD toe
   3 – Voeg een nieuwe MP3 toe
   4 – Print gesorteerd op auteur/uitvoerder/artiest (A->Z)
   5 – Print gesorteerd op prijs (laag naar hoog)
   6 – Schrijf naar file
   7 – Sluit programma
```

Voor opties 1, 2 en 3 kan je dan d.m.v. een aantal vragen aan de gebruiker vragen om de nodige gegevens in te vullen (gebruik hiervoor System.in en kijk bijvoorbeeld naar voorbeeldprogramma 7.6 op bladzijde 178 van editie 3 van het boek).

Hierbij enkele randvoorwaarden voor deze opdracht:

- Denk na over het nut van het toepassen van **inheritance**.
- Het sorteren van de rekeningen is een uitbreiding (zie verderop). Als je beslist om de uitbreiding niet te implementeren, moet je minstens een methode sort() voorzien, die simpelweg afdrukt “Sorteren”. Als je dan toch bezig bent: sorteren mag de werking van het programma niet teveel hinderen, m.a.w., het programma moet responsief blijven terwijl er gesorteerd wordt. Uiteraard kan het sorteren enige tijd duren en daarom moet je het sorteren door een aparte **thread** laten uitvoeren. Zorg er dus voor dat als je wil sorteren:
    - Je een nieuwe thread opstart
    - Je toegang behoudt tot de originele datastructuur (bijvoorbeeld ArrayList) waarin alle devices zitten (dus geen kopie maken)

  Het sorteren zelf is een uitbreiding, zie hieronder.
- Pas echter op, als je met een aparte thread werkt die alle rekeningen sorteert, moet je aan **thread-synchronisatie** denken, anders kunnen er vreemde dingen gebeuren, bijvoorbeeld:
    - De gebruiker geeft 2 keer kort achter elkaar de opdracht om te sorteren (en de eerste sorteer-opdracht is nog bezig)
    - De gebruiker geeft opdracht om te sorteren en tijdens het sorteren worden er al nieuwe voertuigen toegevoegd.

  (!) Trouwens, ook als je sorteren niet implementeert, kan je wel een thread-constructie implementeren die enkel “Sorteren” wegschrijft naar System.out.
- Bij het wegschrijven van de lijst moet de vorige file versie van *apple.txt* overschreven worden.
- Probeer **de file-naam *amazon.txt* niet hard te coderen** in je Java programma, maar mee te geven via (1) de command line bij opstart van het programma of (2) door een vraag aan de gebruiker te stellen.
- Kwaliteit van het programma is uiterst belangrijk! Schrijf daarom **voor de klasse Boek jUnit test methoden**. Zorg voor zowel positieve als negatieve testen.

Nog enkele tips:

- Het programma moet **compileren**.
- Voor een goed cijfer moet het programma ook **werken**, zonder excepties.
- Verzorg de **stijl van het programma**, m.a.w. indenteer je code, kies logische namen, etc.
- Let goed op de vetgedrukte woorden in de beschrijving hierboven, elk stellen ze een belangrijke requirement voor van het programma dat je moet implementeren.

Extra uitbreiding (1 punt)

TIP: zorg er sowieso voor dat je de basisopdracht af hebt voor je aan de extra opdracht begint!

Het sorteren van *alle item* op *2 verschillende manieren* kan je oplossen door klassen te maken die de interface **Comparator** implementeren**.** Alle items *samen* moeten gesorteerd kunnen worden op 2 verschillende manieren:

- Enerzijds op auteur/artiest/uitvoerder
- Anderszijds op prijs

TIP: bekijk de javadoc informatie van **Comparator** (de javadoc pagina's kan je raadplegen via Blackboard!). Je zal 2 klassen moeten definieren die een Comparator implementeren, 1 voor elk van de sorteerwijzen.

Als je het sorteren implementeert, zorg er dan ook voor dat opties 4 en 5 van het tekstuele menu (zie vorige bladzijde) een overzicht van items afdrukt op het scherm (gesorteerd).

Overzicht van scores

- 3 punten voor compileren, niet compileren → score = 1
- 1 punt op goed toepassen van inheritance
- 0.8 punt op inlezen van file
- 0.8 punt op wegschrijven van file
- 0.4 punt op verzorgde code (o.a. commentaar (javadoc))
- 0.5 punt toevoegen boek, CD, MP3
- 0.5 punt command line interface
- 0.5 punt voor het niet hard-coderen van de file-naam
- 0.5 punten voor de thread
- 0.5 punten voor thread-synchronisatie
- 0.5 punten voor de jUnit testen
- 1 punt voor de extra opdracht