TECHNISCHE UNIVERSITEIT DELFT
Faculteit Elektrotechniek, Wiskunde en Informatica

TU Delft

TI1206 Object Georiënteerd Programmeren (computertentamen)
28 oktober 2015, 13:30-16:30

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 → TI1206 → Assignments → Java 1.7 API)

Dit tentamen bevat een **basis-opdracht (9 punten)** en **1 uitbreiding (1 punt)** (totale opdracht: 6 bladzijden).

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

TIP 1: lees de opdracht eerst helemaal door en begin daarna pas met implementeren.

TIP 2: het zou kunnen dat jUnit 4.0 niet standaard staat in het Eclipse build path staat. Als je via "New" een Unit Test Case aanmaakt en jUnit 4.0 aanklikt, dan geeft Eclipse aan dat jUnit niet in het build path staat en kom je onmiddellijk bij het Java build path menu uit. In het tabje "Libraries" klik je op "Add Library" en dan zal Eclipse zelf voorstellen om jUnit toe te voegen.

Een aantal **hints/tips**:

- Je programma moet compileren (niet compileren == niet slagen).
- Bekijk helemaal achteraan de opdracht de scoreverdeling van dit tentamen.
- Als je programma af is, zip je je bestanden. Gebruik hiervoor het programma 7Zip dat beschikbaar is op de computers. 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!
- TIP: de workspace directory van Eclipse staat op de H: schijf. Gebruik het programma 7Zip en ga naar de H: schijf om je opdracht te zippen (doe dit via 7Zip en niet via de Windows Verkenner / Windows Explorer).
- Maak je werk in de "default package", maak geen aparte package aan (dit maakt het nakijkwerk eenvoudiger, -1 punt bij gebruik van packages).
- Uploaden gebeurt via Blackboard → TI1206 → 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.**

ProRail, de infrastructuurbeheerder van het Nederlandse spoorwegnet, heeft behoefte aan een nieuw softwaresysteem. Ze willen een systeem hebben dat bijhoudt welke treinen zich op een bepaalde dag op het spoor bevinden en dit willen ze gebruiken om inzicht te krijgen in de beschikbaarheid en bezetting van de rijpaden van het spoorwegnet. ProRail heeft jou gecontacteerd om een prototype te maken van deze applicatie. ProRail is zich er goed van bewust dat dit slechts een prototype is en ze zijn op dit moment bezig met extra requirements in kaart te brengen, maar ze hebben ook snel toegang nodig tot het prototype!

Meer bepaald wil ProRail dat jij een applicatie ontwikkelt die een XML-bestand inleest waarin staat welke treinen op het spoornet rijden.

Een voorbeeld van deze file ziet er als volgt uit:

```
<TREINEN>
	<INTERCITY>
		<VERTREK>14:20</VERTREK>
		<AANKOMST>15:00</AANKOMST>
		<STATION>Rotterdam Centraal</STATION>
		<ICSTATION>Schiedam Centrum</ICSTATION>
		<STATION>Delft Zuid</STATION>
		<ICSTATION>Delft</ICSTATION>
		<STATION>Rijswijk</STATION>
		<STATION>Den Haag Moerwijk</STATION>
		<STATION>Den Haag HS</STATION>
	</INTERCITY>
	<SPRINTER>
		<VERTREK>14:30</VERTREK>
		<AANKOMST>15:20</AANKOMST>
		<STATION>Rotterdam Centraal</STATION>
		<ICSTATION>Schiedam Centrum</ICSTATION>
		<STATION>Delft Zuid</STATION>
		<ICSTATION>Delft</ICSTATION>
		<STATION>Rijswijk</STATION>
		<STATION>Den Haag Moerwijk</STATION>
		<STATION>Den Haag HS</STATION>
	</SPRINTER>
	<GOEDERENTREIN>
		<VERTREK>7:30</VERTREK>
		<AANKOMST>9:20</AANKOMST>
		<STATION>Eindhoven</STATION>
		<ICSTATION>Weert</ICSTATION>
		<STATION>Roermond</STATION>
		<ICSTATION>Sittard</ICSTATION>
		<STATION>Maastricht</STATION>
	</GOEDERENTREIN>
</TREINEN>
```

De volledige file **treinen.txt** is beschikbaar via Blackboard (en is niet ingesprongen zoals dit voorbeeld, dit maakt de verwerking makkelijker).

De volgorde van de elementen (zowel de top-level elementen (sprinter, intercity, goederentrein) als de low-level elementen (aankomst, vertrek, station, ...)) in de file is willekeurig, daar moet je programma op voorzien zijn, m.a.w., eerst kan er een lijst met stations zijn en pas daarna <VERTREK>, <AANKOMST>, maar net zo goed kan er eerst <VERTREK> staan, daarna stations en dan pas <AANKOMST>. De volgorde van de stations is wel belangrijk en definieert de route van de trein (zie ook hieronder).
De sub-elementen (eigenschappen) per element liggen wel vast (er kan bijvoorbeeld geen <ICSTATION> voorkomen binnen <GOEDERENTREIN>). Elke XML-element (dus <A>...</A>) staat op een nieuwe regel. De eigenschappen per elementen voor de elementen Sprinter, Goederentrein, Intercity, Station en ICStation volgen hieronder:

Een **SPRINTER** wordt gekenmerkt door:
- Het uur van vertrek
- Het uur van aankomst
- Een lijst met stations die deel uitmaken van de route. De trein stopt op elk station op deze route.

Een **Intercity** wordt gekenmerkt door:
- Het uur van vertrek
- Het uur van aankomst
- Een lijst met stations die deel uitmaken van de route. Er is een onderscheid tussen gewone stations en IC-stations in deze lijst. De trein stopt in het begin/eindstation en de tussenliggende IC stations, terwijl de tussenliggende ("normale") stations dienen om de route weer te geven.

Een **Goederentrein** wordt gekenmerkt door:
- Het uur van vertrek
- Het uur van aankomst
- Een lijst met stations die deel uitmaken van de route. Er wordt enkel gestopt op het begin en eindstation; de lijst met stations geeft enkel de route aan.

Een **Station** wordt gekenmerkt door:
- De naam

Een **ICStation** wordt gekenmerkt door:
- De naam

ProRail bekijkt op dit moment hoe internationale treinen zoals Thalys, ICE en Beneluxtrein in dit schema gegoten kunnen worden. Belangrijke overweging hierbij is het idee om een nieuw type station te introduceren dat aangeeft of een internationale trein er kan stoppen. Hou in je klasse design rekening met uitbreidingen in die richting.

Nu vraagt ProRail om een programmaatje te maken dat:

- De file treinen.txt die hierboven beschreven wordt kan **inlezen**
- Het toelaat om nieuwe treinen **toe te voegen** aan het spoorwegnet
- Het toelaat om alle treinen tussen station A en B te **zoeken** (ongeacht tussenliggende stations).
- Het toelaat om alle treinen **weg te schrijven naar file** (alle treinen die zich in de datastructuur in het geheugen bevinden worden weggeschreven naar file).
- Het toelaat om het bestand (met bijvoorbeeld nieuwe configuraties van toestellen) **weg te schrijven naar file.**
- Een **equals()** methode te schrijven voor elke klasse (behalve de klasse die de main bevat)
- 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 - Geef alle treinen in de in-memory database weer op het scherm
   2 - Geef alle treinen van station A naar station B weer
   3 - Voeg een trein toe aan de database
   4 - Wegschrijven naar file
   5 - Stop het programma
```

Optie 1:
Alle treinen worden netjes op het scherm weergegeven in volgend formaat:

```
Intercity van Den Haag HS naar Rotterdam Centraal
   Vertrek: 13:20
   Aankomst: 14:00
   ---Den Haag HS
   ----(Den Haag Moerwijk)
   ----(Rijswijk)
   ---Delft
   ----(Delft Zuid)
   ---Schiedam Centrum
   ---Rotterdam
```

Deze weergave maakt dus een onderscheid tussen stations die op de route liggen (bijvoorbeeld Rijswijk) en station waar de trein stopt (bijvoorbeeld Delft). Hou rekening met volgende:

- een Intercity stopt bij begin en eindstation en bij ICStations
- een Sprinter stopt bij elk station
- een Goederentrein stopt enkel bij begin en eindstation

Optie 2

- Je vraagt de gebruiker om een begin en eindstation op te geven. Gebruik hiervoor System.in en kijk bijvoorbeeld naar het voorbeeldprogramma 7.6 op bladzijde 178 van editie 3 van het boek.
- Geef een overzicht van de trein die voldoen aan deze zoekopdracht (gelijkaardig aan het overzicht van Optie 1)

Optie 3

- Verzamel door vragen te stellen aan de gebruiker informatie over het begin en eindstation van de nieuwe trein, het vertrek en aankomstuur en alle tussenliggende stations. Bij een intercity moet er rekening worden gehouden met ICStations. De volgorde van het invoeren van de stations is de te volgen route.
  (maak opnieuw gebruik van System.in)
- Voorbeeld:

```
Wat is het vertrekstation?
     Amsterdam Centraal
Wat is het uur van vertrek?
     14:55
Wat is het uur van aankomst?
     17:00
Wat is het station op de route (vul S in om te
stoppen met tussenliggende stations)
...
Is het laatst toegevoegde station een InterCity
station? [j(a)/n(ee)]
Wat is het aankomststation?
     Brussel-Centraal
```

Optie 4

De gegevens worden weggeschreven naar file, in **hetzelfde XML formaat** zodat de applicatie de gegevens opnieuw kan inlezen.

Optie 5

De applicatie stopt.

Hierbij enkele **randvoorwaarden** voor deze opdracht:

- Denk na over het nut van het toepassen van **inheritance**.
- Bij het wegschrijven van de lijst moet de vorige file versie van treinen.txt overschreven worden.
- Probeer **de file-naam treinen.txt niet hard te coderen** in je Java programma, maar mee te geven bij het starten van het programma.
- Schrijf unit testen (zie ook achteraan bij de puntenverdeling)!
- Het is verboden om de standaard XML functionaliteit van Java te gebruiken. Je moet de file *zelf* uitlezen, bijvoorbeeld d.m.v. een Scanner.

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. Daar hoort ook javadoc commentaar bij!

Extra uitbreiding (1 punt)

Het op het scherm zetten van alle treinen in de database (Optie 1) kan leiden tot een onoverzichtelijke hoop gegevens. Daarom de suggestie van ProRail om de gegevens te sorteren voor ze op het scherm worden afgedrukt.

Deze uitbreiding van de opdracht heeft dus betrekking op een verbetering van Optie 1 op 2 manieren:

- De resultaten worden gesorteerd op het scherm gezet.
- Het sorteren zelf gebeurt in een aparte thread waardoor het programma responsief blijft.

Sorteren

Sorteren gebeurt door gebruik te maken van de interface **Comparator**. (kijk naar de Javadoc van Comparator! … andere manieren van sorteren geven geen aanleiding tot een bonus)

Zorg ervoor dat je bij het sorteren (1) rekening houdt met het vertrekstation, (2) en in geval van een identiek vertrekstation dan sorteert op het aankomststation en (3) in geval van een identiek aankomst en vertrekstation sorteert op het uur van vertrek.

Threads

Het sorteren doe je in een aparte thread. Echter, terwijl de datastructuur met alle treingegevens erin gesorteerd wordt, moet je er voor zorgen dat de gebruiker in de "main thread" geen gegevens aanpast in deze datastructuur. Maak *geen* kopie van de datastructuur, maar zorg voor thread synchronisatie!

> (!) Trouwens, ook als je sorteren niet implementeert, kan je wel een thread-constructie implementeren die enkel "Sorteren" wegschrijft naar System.out.

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

Overzicht van scores (totaal: 10)

- 2.5 punten voor compileren; niet compileren → score = 1
- 1 punt op goed toepassen van inheritance
- 0.5 punt op correcte implementatie van equals()
- 0.8 punt op inlezen van file
- 0.8 punt op wegschrijven van file
- 0.5 punt op verzorgde code (o.a. commentaar (javadoc))
- 0.7 punt toevoegen nieuwe trein
- 0.5 punt command line interface (inclusief optie 1, alle treinen op het scherm)
- 0.5 punt voor het niet hard-coderen van de file-naam
- 1.2 punt voor de jUnit testen
  - 0.6 voor het goed testen van de klasse Trein
  - 0.6 voor het goed testen van alle andere klassen (behalve de klasse met de main van je programma)
- 1 punt voor de **extra opdracht**
  - 0.5 voor het werken met een Thread
  - 0.5 voor het sorteren

Er is een aftrek van 1 punt als je *niet* werkt binnen de default package!