Tentamen OOP2 Deeltijd

Donderdag 18 juni 2015

Naam	
Studentnr.	
Klas	
Inlevertijd (door surveillant in te vullen	

Instructies

1. Je moet voor deze opgave code toevoegen aan een bestaand startproject. Download de zip van het startproject van Moodle.
2. Upload na afloop je hele Eclipse project naar Moodle.
3. LET OP: noem het bestand <Studentnaam>.zip.
4. Je moet je naam op drie plaatsen invullen:
    a. In de naam van de *zip-file* die je gaat uploaden (zie punt 3 hierboven)
    b. In de naam van het *Eclipse project* (zie Stap 0)
    c. In het *programma* in de klasse Stap0 (zie Stap 0).

Opmerkingen bij dit tentamen

1. Deze toets omvat drie pagina's en is dubbelzijdig afgedrukt. Vergeet de achterkant niet!
2. Uitsluitend toegestaan is het gebruik van het boek, je eigen aantekeningen en al het materiaal op je eigen laptop (inclusief uitwerkingen van practicumopgaven).
3. Na afloop van het tentamen dient dit opgaveblad bij de surveillant te worden ingeleverd. Uitwerkingen in Moodle waarbij geen bijbehorend opgaveblad is ingeleverd worden niet nagekeken!

Richtlijnen bij coderen (zie ook ITOPIA code conventies [ICC])

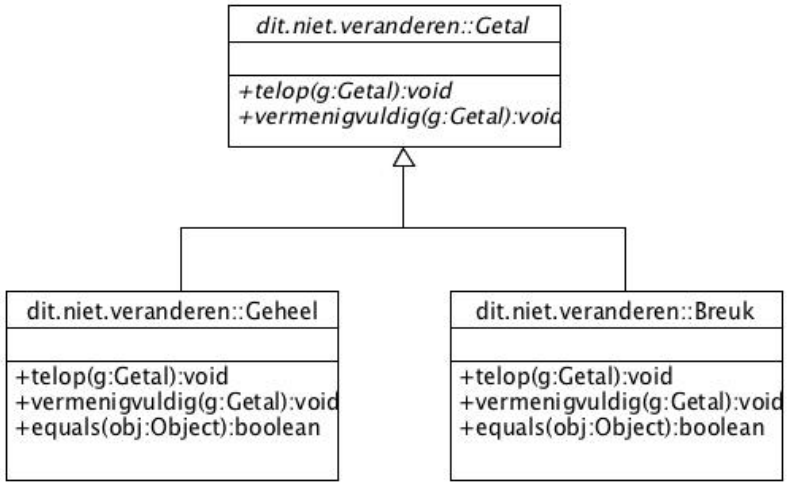
1. Zorg dat je naam en het doel van het programma bovenin staan (ICC #1).
2. Gebruik de juiste inspringing (indentation) bij de lay-out (ICC #2) .
3. Let op juist gebruik hoofdletters en kleine letters (ICC #3).
4. Gebruik goede namen (ICC #4).
5. Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).

Inleiding

In deze toets gaan we onder andere een bestaand systeem testen. Dat systeem is een deel van een rekenmachine voor het basisonderwijs. De machine kan werken met gehele getallen zoals 1, 3, 250… en met breuken zoals 1/3, 25/12 en 314/100. Als het klaar is moet het systeem alle rekenkundige bewerkingen kunnen uitvoeren maar op dit moment zijn alleen optellen en vermenigvuldigen geïmplementeerd.

Alle klassen van de rekenmachine zijn aan het Eclipse project toegevoegd in het package `dit.niet.veranderen`. De klassen in dat package mag je wel bekijken maar je mag ze niet wijzigen! Voor het maken van deze toets is het niet nodig dat je de details van de implementatie begrijpt maar je moet de klassen wel kunnen gebruiken.

De ontwerpers van de rekenmachine hebben besloten om geen gebruik te maken van de bestaande Java klassen voor getallen zoals Integer en Double maar hun eigen implementatie van getallen te gebruiken. Daarom zijn er drie klassen gemaakt volgens dit UML diagram:



De abstracte klasse *`Getal`* is de superklasse van `Geheel` en `Breuk`. De klasse `Geheel` implementeert de gehele getallen, `Breuk` implementeert de breuken. Beide concrete klassen implementeren alle abstracte methoden uit *`Getal`* en ook de methode equals die je kan gebruiken in Stap 1.

Op dit moment kan de rekenmachine alleen maar bewerkingen op soortgelijke getallen uitvoeren, dus gehele getallen kunnen worden opgeteld bij gehele getallen maar niet bij breuken. Breuken kunnen worden vermenigvuldigd met andere breuken maar niet met gehele getallen. Pogingen om "gemixte" berekeningen uit te voeren resulteren in een RekenfoutException.

Stap 0

Download het project en importeer het in Eclipse. Het project heet nu Toets OOP2 Deeltijd 2015-06-18. Hernoem dat naar <Groep>_<Studentnaam>_<Studentnr_OOP2 Toets DT, bijv. ID101_JanKlaassen_500123456_ OOP2 Toets DT.

Run de klasse `Stap0` een keer. In het begin wordt de string "`Dit is de OOP2 toets van <Naam> - <StudentNummer>`" afgedrukt. Wijzig die boodschap zo dat je eigen naam en studentnummer worden afgedrukt.

Bekijk ook het voorbeeld van het gebruik van breuken in de klasse `Stap0`.

Stap 1, Unit tests (30 punten)

Genereer met Eclipse unit tests voor de klasse `Breuk`. Maak voldoende tests voor de methoden `telop` en `vermenigvuldig` waarbij je de volgende testdata mag gebruiken:

a	b	a+b	a*b
1/2	1/2	1/1	1/4
1/2	1/6	2/3	1/12
3/4	3/8	9/8	9/32
5/12	0/3	5/12	0/36
1/1	3/7	10/7	3/7

LET OP! De methode `telop` is opzettelijk niet goed geïmplementeerd. Laat dat zien doordat je tenminste één test maakt die faalt.

Stap 2, Excepties (20 punten)

In de klasse `Stap2` wordt een poging gedaan een geheel getal te vermenigvuldigen met een breuk. Omdat dergelijke gemixte bewerkingen nog niet zijn toegestaan wordt er een exceptie gegooid die in de methode `opgave2` wordt opgevangen. De afhandeling is echter heel algemeen en weinig zeggend. Pas de catch clause zo aan dat er exceptie van het goede type wordt gevangen en dat er een passende foutmelding wordt afgedrukt.

Stap 3, Binaire IO (40 punten)

Het project bevat een file met de naam getallen.dat die instanties van (de subklassen van) *`Getal`* bevat, zowel breuken als gehele getallen. In de klasse `Stap3` wordt die file gelezen en afgedrukt. Daarom moet je twee methoden afmaken.

1. De methode `leesBinaireData(String filenaam)` : Lees alle getallen één voor één uit de file. Voeg gehele getallen toe aan een lijst die je uiteindelijk weer als resultaat oplevert. Breuken die in de invoer voorkomen moeten worden genegeerd (weggegooid). De lijst die je maakt bevat dus uitsluitend gehele getallen en ook alle gehele getallen uit de invoer.
2. De methode `writeOutput(List<Getal> getallen)` : Schrijft alle getallen uit de lijst getallen *in tekstvorm* naar de output. Zet alle getallen op dezelfde regel, gescheidendoor komma's.

De uitvoer moet er als volgt uitzien:

```
3, 1, 4, 1, 5, 9, 2, 7
```

Beoordeling

Stap 1:	30
Stap 2:	20
Stap 3:	40
Itopia code conventions	10
Totaal	**100**

Einde van de toets, succes!