



EX/2015-16/5/155963

Groep	Naam, Voornaam (student)	Studentennummer	Examencode 2015165155963-08
-------	--------------------------	-----------------	--------------------------------

	Periode: 5	AS / Jaar
Datum Donderdag 18 augustus	Type examen Laptop	Examenduur 120 13.30-15.30
Toegelaten hulpmiddelen Open boek = papieren documentatie en laptop (geen internet behalve Blackboard)	Vaktitularis(sen) kris.behiels@kdg.be mark.goovaerts@kdg.be	Punten

LEES AANDACHTIG DEZE INSTRUCTIES VOORALEER JE AAN HET EXAMEN BEGINT

Algemene richtlijnen

Dit is een **open boek**-examen, maar het is niet toegelaten om te communiceren met anderen via internet, een ad hoc-netwerk of op een andere wijze. Bovendien wordt elke activiteit op de laptop gelogd en moet het logbestand op het einde worden ingeleverd. Volg de procedure voor de laptopexamens. **Elke inbreuk tegen deze procedure wordt behandeld als een vorm van fraude!**

Ga in Blackboard naar de cursus "**EXAMEN: Programmeren 2 Java examen**". Volg nauwgezet de procedure die daar beschreven wordt. In stap 2 moet je de opgave downloaden en unzippen. Open het project in IntelliJ, mogelijk moet je de versie van de JDK aanpassen.

Belangrijk

- Als je **configuratieproblemen** hebt, roep er dan een Java-docent bij. (Dat kost je GEEN punten)
- Naast **IntelliJ** maak je ook gebruik van **junit**, deze jars zitten reeds mee gezippt in de projectfile.
- Sla regelmatig je volledige project op. Op het einde van het examen heb je **geen excuus** indien er een bestand "verloren" is gegaan!
- Het is **NIET** toegelaten om namen van klassen, packages of projecten te wijzigen.
- Werk rustig, maar hou toch de tijd in de gaten. Per onderdeel wordt de **richttijd** en het **puntengewicht** (in %) aangegeven.
- ➡ • Tik in de commentaarzone van de klasse **ReflectionRunner** van **1_Annotations** bovenaan je **naam** en **klasgroep** in. Doe het nu meteen a.u.b. In de andere klassen is dat niet nodig.

Algemeen concept :

Het IntelliJ project bestaat uit 4 modules die onafhankelijk van elkaar gecompileerd en gerund kunnen worden:

- Deel 1 gaat over annotations (15%)
- Deel 2 gaat over concurrency en streams (20%)
- Deel 3 gaat over XML en streams (30%)
- Deel 4 gaat over patterns (35%)

Kies zelf met welk deel je begint.



EX/2015-16/5/155963

Groep	Naam, Voornaam (student)	Studentennummer	Examencode 2015165155963-08
-------	--------------------------	-----------------	--------------------------------

DEEL 1: (15%)	Annotations	richttijd: 20min
---------------	-------------	------------------

We willen **reflection** toepassen en at runtime methoden benaderen die voorzien zijn van de custom annotation **@Refactored**

1.1 Maak in de package **annotated** een annotation met de naam **@Refactored**; ze is bedoeld om at runtime gebruikt te worden en boven een methode geplaatst te worden. De annotation heeft 2 parameters, een **String** met de naam **value** en een **int** met de naam **count**.

In de klasse **Programmer** hoef je niets te wijzigen, de rode aanduidingen verdwijnen vanzelf als je de annotation gemaakt hebt.

1.2 Vul de klasse **RunDeel1** aan met een **for** lus zodanig dat je in de map voor elk van de verschillende programmeurs (= **value** parameter van de annotation) in de map als value het totaal van alle **count** parameterwaarden bekomt.

Uiteindelijk dien je de volgende afdruk te bekomen:

```
Refactorings in klasse be.kdg.annotations.Programmer:  
{Erik=3, Jos=5, Staf=2}
```

DEEL 2: (20%)	Concurrency en streams	richttijd: 20 min
---------------	------------------------	-------------------

In plaats van met **Runnable** te werken willen gebruik maken van **Callable** die ons het voordeel biedt om gebruik te maken van **ThreadPools**.

2.1 Laat de klasse **PersoonCallable** de interface **Callable** implementeren en werk de daarbij horende methode uit. Laat de geïmplementeerde methode het gemiddelde loon van alle personen in de **List** teruggeven. Maak hierbij verplicht gebruik van een stream; dus geen lus.

2.2 Maak in de klasse **RunDeel2** gebruik van een **stream** om een **List** van objecten te maken die alleen mannen bevat en die in volgorde van leeftijd gesorteerd is (van oud naar jong). Druk vervolgens de inhoud van de **List** af zodat elke man op een aparte regel komt. Gebruik hiervoor **geen lus**; wel een specifieke methode. Doe exact hetzelfde maar nu voor alle vrouwen.

2.3 Maak gebruik van een fixed threadpool om beide **PersoonCallable** threads parallel te laten uitvoeren. Ze berekenen (zoals in 2.1 aangegeven) respectievelijk het gemiddelde loon van de mannen en de vrouwen. Druk vervolgens de resultaten af (**zie hiervoor de verwachte uitvoer op de volgende pagina**).



EX/2015-16/5/155963

Groep	Naam, Voornaam (student)	Studentennummer	Examencode
			2015165155963-08

Mannen:

Fred Somers	MAN 1965	€3140,84
Patrick Devos	MAN 1967	€2140,21
Lucas Van de Ven	MAN 1972	€3472,44
Eric Buelens	MAN 1984	€2510,24

Vrouwen

Els Callens	VROUW 1974	€1478,91
Griet Op de Beeck	VROUW 1976	€2368,11
Floor Custers	VROUW 1989	€2541,92
Dorien Vanderpoorten	VROUW 1994	€1487,35

Gemiddeld loon mannen: €2815,93

Gemiddeld loon vrouwen: €1969,07

DEEL 3: (30%)	XML en streams	richttijd: 30 min
----------------------	-----------------------	--------------------------

In de package `model` zit de klasse `Foto` die de attributen `nummer (int)`, `soort (enum Soort - ACTION of PORTRAIT)`, `diafragma (double)`, `sluitertijd (String)` en `iso (int)` bevat. De klasse `Portfolio` bevat een `ArrayList` van objecten van deze klasse `Foto`.

3.1 Schrijf in de klasse `ParsingTools` (package `parsing`) de methode `maakXML` verder uit. Je moet ervoor zorgen dat de volgorde van de elementen (attributen) in de xml als volgt is: `nummer - soort - iso - diafragma - sluitertijd`. Maak hierbij verplicht gebruik van de klasse `Marshaller`. Controleer het resulterende XML bestand met de naam `portfolio.xml`.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<portfolio>
  <foto>
    <nummer>53</nummer>
    <soort>ACTION</soort>
    <iso>200</iso>
    <diafragma>2.8</diafragma>
    <sluitertijd>1/1000</sluitertijd>
  </foto>
  <foto>
    <nummer>41</nummer>
    <soort>PORTRAIT</soort>
    <iso>100</iso>
    <diafragma>4.0</diafragma>
    <sluitertijd>1/200</sluitertijd>
  </foto>
  <foto>
    <nummer>26</nummer>
    <soort>ACTION</soort>
    <iso>400</iso>
    <diafragma>11.0</diafragma>
    <sluitertijd>1/500</sluitertijd>
  </foto>
  // enzovoort (let op de volgorde van de elementen)
```



EX/2015-16/5/155963

Groep	Naam, Voornaam (student)	Studentennummer	Examencode
			2015165155963-08

3.2 Schrijf in de klasse **ParsingTools** de methode **leesXML** verder uit. Het resultaat moet uiteindelijk in de **List** met de naam **myList** in de **main** van **RunDeel3** terechtkomen.

3.3 Druk in de **main** van **RunDeel3** met zo weinig mogelijk code de inhoud van **myList** af, maak hierbij gebruik van de **toString** methode van de klasse **Foto**.

3.4 Druk in de **main** van **RunDeel3** alleen de **ACTION** foto's af in volgorde van oplopend nummer en zorg ervoor dat er geen dubbele nummers meer voorkomen. Controleer de uiteindelijke uitvoer die je hierna vindt:

Alle foto's uit XML-file:

```
00053 A 2,8 1/1000 200
00041 P 4,0 1/200 100
00026 A 11,0 1/500 400
00053 A 2,8 1/1000 200
00127 P 1,8 1/1000 100
00547 A 8,0 1/1000 200
00013 P 3,5 1/1000 200
00016 A 5,6 1/60 1600
```

Alle action foto's volgens nummer (zonder dubbels):

```
00016 A 5,6 1/60 1600
00026 A 11,0 1/500 400
00053 A 2,8 1/1000 200
00547 A 8,0 1/1000 200
```

DEEL 4: (35%)	Patterns en Testen	richttijd: 30 min
----------------------	---------------------------	--------------------------

De situatie is als volgt: Er bestaat al een klasse **Film** die je niet meer mag wijzigen en we gaan ervan uit dat er ook reeds een klasse **Verzameling** waarop het **Singleton Pattern** is toegepast is en die je ook niet meer mag wijzigen. Deze laatste klasse moet je echter nog wel schrijven! (4.1)

4.1 Maak in de package **patterns** de klasse **Verzameling**, ze moet aan de volgende voorwaarden voldoen:

- Pas hier het **singleton pattern** toe (maak geen gebruik van *lazy initialization*)
- De klasse bevat slechts één attribuut, met name een **arrayList** met de naam **filmList**
- Voorzie een **static** methode **getInstance** om de enige verzameling terug te geven.
- Voorzie een methode **add** om een **Film**-object dat nog niet in de **arraylist** voorkomt toe te voegen
- Voorzie een methode **aantalFilms** die het aantal elementen in de **arraylist** teruggeeft
- Voorzie een methode **print** om de volledige inhoud van de **arraylist** via de **toString** methode van de klasse **Film** onder elkaar af te drukken.



EX/2015-16/5/155963

Groep	Naam, Voornaam (student)	Studentennummer	Examencode
			2015165155963-08

4.2 Maak nu de klasse **Verzameling** *observeerbaar* zonder nog iets aan de klasse zelf te wijzigen! Maak in dezelfde package een nieuwe klasse met de naam **ObservableVerzameling** die dezelfde functionaliteit heeft als de originele klasse met de naam **Verzameling**.

4.3 De nog te schrijven (in dezelfde package) klasse **VerzamelingObserver** moet ervoor zorgen dat er telkens wanneer er in de klasse **ObservableVerzameling** een **film** object toegevoegd wordt de volledige inhoud van de verzameling via de **print** methode afgedrukt wordt. Als alles goed is bekom je na het uitvoeren van **RunDee14** het volgende (**je moet nog wel iets toevoegen en de kommentaartekens verwijderen**) :

```
2015 "Star Wars: The Force Awakens"  
2015 "The Martian"  
1998 "Psycho"  
1960 "Psycho"
```

4.4 Vul in de package **test** de klasse **TestVerzameling** aan. Schrijf een test om na te gaan of het *singleton pattern* wel degelijk correct is toegepast. Schrijf ook een test om na te gaan of de **add** methode toch geen dezelfde films toevoegt. Verplicht gebruik van de annotatie **@Before** en **@After**.

Zorg ervoor dat je een groene balk krijgt na uitvoeren van de testen.

Veel Succes