KdG
Karel de Grote Hogeschool

EX/2016-17/2/9048/3

Programmeren 1 - Java: OO Technieken

Groep	Naam, Voornaam (student)	Studentnummer	Examencode
			201617290483-84

	Periode: 2	AS / Jaar
Datum 11 januari 2017 8:30	Type examen Laptop	Examenduur 120
Vaktitularis(sen) anniek.cornelis@kdg.be hans.vochten@kdg.be herwig.desmet@kdg.be johan.ven@kdg.be kris.behiels@kdg.be	 lars.willemsens@kdg.be mark.goovaerts@kdg.be wim.dekeyser@kdg.be wouter.deketelaere@kdg.be	Punten

Toegelaten hulpmiddelen
Openboek: handboek(en), nota's, laptop, GEEN INTERNET, behalve Blackboard.

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!

A. Ga in Blackboard naar de cursus **KDG_HB.BTI.26114_EXAMEN: Programmeren 1 - Java: OO Technieken examen** en volg daar de procedure. In **stap 2** moet je de opgave downloaden en unzippen.

B. **Hernoem** de folder uit A als volgt: "***Groep_Achternaam_Voornaam***" (bv: INF102A_Hetfield_James).

C. **Open** het project in de folder met IntellIJ.

D. **Zoek het** bestand **"achternaam_voornaam.txt"** en hernoem met jouw naam.

E. Nu ben je klaar om de opgave te lezen. Succes!

KdG
Karel de Grote Hogeschool

EX/2016-17/2/9048/3

Programmeren 1 - Java: OO Technieken

Groep	Naam, Voornaam (student)	Studentnummer	Examencode 201617290483-84

OPGAVE

Grondig lezen en herlezen

In deze opgave ga je een **slim digitaal archief** maken. Een digitaal archief bevat een vooropgesteld aantal digitale **dozen** en **kokers** (wij gebruiken hier respectievelijk 200 en 100 stuks), tezamen noemen we deze **archiefhouders**. Nieuwe **archiefstukken** worden door het archief automatisch in de juiste archiefhouders gestockeerd: **boeken en video's** gaan in dozen en **kaarten** worden bewaard in ronde kokers.

Wanneer de *main* methode van *TestArchief* volgende console-output geeft, is je oplossing correct. Deze methode gaat random archiefstukken toevoegen aan het slimme archief totdat er een fout optreedt.

```
be.kdg.examen.klassement.exceptions.ArchiveringsException: Archief: geen kokers meer beschikbaar
Inhoud archief:
Aantal gebruikte dozen: 95
Aantal gebruikte kokers: 100
boek (2016-12-07 09:50:55,   20p, 0.01m³)
boek (2016-12-07 09:50:55,   25p, 0.01m³)
boek (2016-12-07 09:50:55,   42p, 0.02m³)
boek (2016-12-07 09:50:55,   44p, 0.02m³)
// deel weggelaten
boek (2016-12-07 09:50:55,  949p, 0.35m³)
boek (2016-12-07 09:50:55,  968p, 0.35m³)
boek (2016-12-07 09:50:55,  979p, 0.36m³)
boek (2016-12-07 09:50:55,  993p, 0.36m³)
boek (2016-12-07 09:50:55,  997p, 0.36m³)
boek (2016-12-07 09:50:55, 1000p, 0.37m³)
boek (2016-12-07 09:50:55, 1009p, 0.37m³)
```

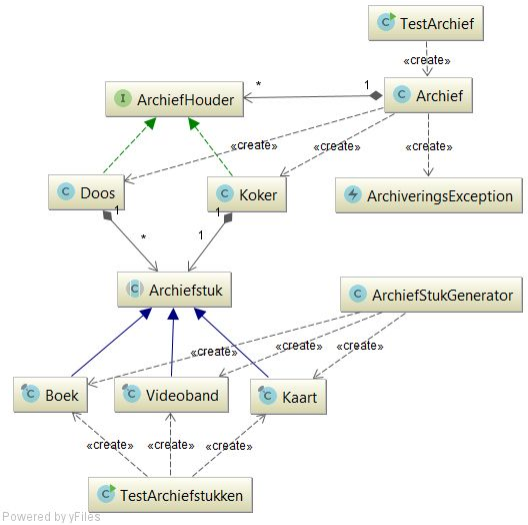
De volgende vragen leiden naar de oplossing van dit probleem. Lees elke vraag grondig, want ze bevatten aanwijzingen. Denk grondig na en gebruik **autocompletion** van IntellIJ zoveel als mogelijk. Probeer verder op elke vraag een antwoord te geven: **elk punt telt**, zelfs als het geheel niet werkt. Sla desnoods een vraag over.

Hiernaast zie je nog een **UML diagram** dat de relaties tussen de verschillende elementen van het archief verduidelijkt.

ArchiefStukgenerator, *TestArchiefstukken* en *TestArchief* zijn gewoon hulpklassen die de opgave doen runnen.

Na Vraag 1 kan je met *TestArchiefstukken* testen of je ze correct hebt opgelost. Neem de aangeduide code uit commentaar. De dan verwachte output vind je onderaan in deze file zelf.

Als je opgave helemaal af is, moet je in de *maakRandomArchiefStuk* methode van *ArchiefStukgenerator* een deel van de code uit commentaar halen en één regel in commentaar zetten zoals daar aangegeven. Bekijk deze files nu alvast eens.



KdG
Karel de Grote Hogeschool

EX/2016-17/2/9048/3

Programmeren 1 - Java: OO Technieken

Groep	Naam, Voornaam (student)	Studentnummer	Examencode 201617290483-84

Opmerking 1: de hoeveelheid gebruikte dozen kan in jouw output **verschillen** evenals het aantal blz., ...
Opmerking 2: je moet **geen extra bestanden** aanmaken.
Opmerking 3: **bestaande code mag je NIET wijzigen**, tenzij er in commentaar vermeld staat "*hier aanvullen/wijzigen*" of "*mag je wijzigen*".

ARCHIEFSTUKKEN uitwerken – totaal van 35 punten (35%), subtotaal 35 / 100
Archiefstuk.java, Boek.java, Kaart.java, Videoband.java

Vraag 1a – 15 punten (15%)
Vervolledig de code in bovenstaande files zodat er drie soorten archiefstukken zijn. De eigenschappen van deze drie soorten archiefstukken zijn:

BOEK
- huidig moment als archiveringsdatum
- een random aantal pagina's tussen 10 en 1009
- elk boek heeft de afmetingen van een A4 – papier, nl. **lengte (l)**: 0.29m, en **breedte(b)**: 0.21m
- de **hoogte(h)** van een boek wordt bepaald door het aantal pagina's in het boek, de dikte van de pagina's (0.004m) en de dikte van de kaften (2 x 0.006m)
- mag niet verder uitgebreid worden

VIDEOBAND
- huidig moment als archiveringsdatum
- elke videoband heeft volgende afmetingen **lengte(l)**:0.187, **breedte(b)**: 0.103 en **hoogte(h)**: 0.025
- mag niet verder uitgebreid worden

KAART
- huidig moment als archiveringsdatum
- heeft **geen** afmetingen
- mag niet verder uitgebreid worden

Vraag 1b – 20 punten (20%)
We vervolledigen nu de klassen *Archiefstuk*, *Boek*, *Videoband* en *Kaart* volgens de volgende vereisten:

1. **Elk archiefstuk** implementeert **twee functionaliteiten**: zijn **omvang** kan opgevraagd worden als een decimaal getal. Verder kan hierop **gesorteerd** worden met andere archiefstukken.
   Extra info:
   - De **omvang van een boek** wordt bepaald door zijn afmetingen (zie Vraag 1a), dus $l \cdot b \cdot h$
   - De **omvang van een videoband** wordt bepaald door zijn afmetingen (zie Vraag 1a), dus $l \cdot b \cdot h$
   - De **omvang van een kaart** wordt per definitie gelijkgesteld aan **0**.

   Schrijf **alle** methoden die **twee bovenstaande functionaliteiten** implementeren in de klasse-hiërarchie van de archiefstukken. Doe dit zodanig dat later nieuwe archiefstukken makkelijk toegevoegd kunnen worden. Tip: je hebt *Double.compare* nodig.

2. Zorg er nu voor dat elk *Boek*-object op gepaste wijze **uitgeprint** kan worden op het scherm. Kijk daarvoor naar de console-output op pagina 2.

KdG
Karel de Grote Hogeschool

EX/2016-17/2/9048/3

Programmeren 1 - Java: OO Technieken

Groep	Naam, Voornaam (student)	Studentnummer	Examencode 201617290483-84

3. Aan elk archiefstuk kan met de methode `hoortIn` gevraagd worden in welke ***type*** `ArchiefHouder` het thuis hoort. Een boek of een videoband wordt gearchiveerd in een doos, een kaart in een koker.

4. Kijk ook goed na welke eigenschappen van bovenstaande klassen **niet meer aangepast mogen** worden en dwing dit af.

Je kan nu met `TestArchiefstukken` testen of je deze vraag correct hebt opgelost. Pas aan waar nodig.
ARCHIEFHOUDER uitwerken – totaal van 20 punten (20%), subtotaal 55 / 100
`ArchiefHouder.java`

Vraag 2a – 4 punten (4%)
Vervolledig de interface `ArchiefHouder` verder zodat **implementaties** ervan voldoen aan volgende **eigenschappen/functionaliteiten:**

- je kan hun maximaal volume opvragen. Dit is een decimale waarde.
- je kan hun huidige vullingsgraad opvragen (= percentage van het maximaal volume)
- je kan controleren of ze vol zijn
- je kan vragen of om een archiefstuk te ontvangen om het in zichzelf te stockeren

Tip: interfaces hebben geen attributen maar beschikken enkel over methoden.

DOZEN en KOKERS uitwerken
`Doos.java en Koker.java`

Vraag 2b – 12 punten (12%)
Het archief gebruikt alleen dozen en kokers (`Doos` en `Koker`) als archiefhouders en deze moeten dus voldoen aan de voorwaarden uit **Vraag 2a**:

- ze kennen hun eigen maximaal volume als een decimale waarde
- ze kennen hun huidige vullingsgraad
- ze weten of ze al dan niet vol zijn
- ze kunnen het ontvangen archiefstuk op correcte wijze in zichzelf stockeren

Werk deze functionaliteit uit in de gepaste files en hou rekening met de volgende extra vereisten:

DOZEN

a. Dozen hebben allen dezelfde afmeting **(lengte**: 0.305m, **breedte**: 0.215m, **hoogte**: 0.110m).
b. Dozen stockeren boeken en videobanden in een lijst. Er kunnen dus meerdere boeken en videobanden in.
c. De omvang van de boeken en videobanden bepaalt de vullingsgraad van de doos.
d. Dozen zijn vol wanneer zij een vullingsgraad hebben van meer dan 95% (maximum 100%). Hiervoor moet je natuurlijk eerst **de totale omvang van de boeken en videobanden** in de doos berekenen.

KdG
Karel de Grote Hogeschool

Programmeren 1 - Java: OO Technieken

Groep	Naam, Voornaam (student)	Studentnummer	Examencode 201617290483-84

KOKERS

a. Kokers hebben steeds dezelfde afmeting (**hoogte** (h) : 0.450m, **straal**(r) : 0.25m). De capaciteit van een koker kan je bereken met de **formule $2 \cdot \pi \cdot r^2 \cdot h$**
b. In een koker kan slechts één kaart gestockeerd worden.
c. De vullingsgraad is steeds 1. Hier is een zeer elegante oplossing voor in de interface `ArchiefHouder`.

Vraag 2c – 4 punten (4%)
Een doos heeft een extra methode: `public List<Boek> getBoeken()`.
Werk deze methode uit zodat deze een lijst teruggeeft met daarin enkel de boeken in die doos.

ARCHIEF uitwerken – totaal van 30 punten (30%), subtotaal 85 / 100
`Archief.java`
Opgelet: dit is het moeilijkste deel van de opgave.

Vraag 3a – 4 punten (4%)
Het archief houdt voor beide **soorten** archiefhouders (`DoosType` en `KokerType uit de ArchiefHouder-interface`) een lijst bij van archiefstukken. Zorg ervoor dat deze beide lijsten opvraagbaar zijn via hun type m.b.v. een Map waarin per type een lijst van archiefstukken wordt bijgehouden. Dus voor elke type is er een lijst in de Map.

Vraag 3b – 10 punten (10%)
Wanneer een archief **geconstrueerd** wordt, gebeurt dit op basis van twee parameters: `aantalDozen` en `aantalKokers`, zie de `Archief`-klasse. Vul deze code aan m.b.v. de rest van de vragen.

Schrijf hiervoor een extra hulpmethode genaamd `initialiseerArchief(int aantalDozen, int aantalKokers)` in de klasse `Archief`. Deze methode wordt opgeroepen bij de **aanmaak** van een nieuw archief en maakt de correcte hoeveelheid lege dozen en lege kokers aan in het archief.

Vraag 3c – 10 punten (10%)
Het archief kan een nieuw archiefstuk automatisch stockeren in de juiste houder m.b.v. **twee methoden**:

1. `bepaalHouder`:
   - als het archiefstuk in een koker hoort, geef je gewoon de laatste vrije koker terug. Onthou de laatste vrijekoker m.b.v. een getal: `kokerIndex`.
   - als het archiefstuk in een doos hoort, controleer je of de laatste gebruikte doos al vol is of niet. Als het archiefstuk de doos niet vol maakt, geef je deze doos terug. Anders neem je de volgende lege doos. Gebruik hiervoor ook een index: `doosIndex`.
   - Wanneer er geen dozen of kokers meer beschikbaar zijn in het archief ontstaat er een foutmelding (een `ArchiveringsException`). Gebruik deze in de methode `bepaalHouder` op de gepaste plaats. De foutmelding is anders voor dozen dan voor kokers: "Archief: geen kokers meer beschikbaar" vs. "Archief: geen dozen meer beschikbaar"

2. `stockeer`: roept `bepaalHouder` aan om de gepaste houder (`Doos` of `Koker`) op te vragen en stockeert het archiefstuk in die bepaalde houder (=slechts 2 regeltjes)

KdG
Karel de Grote Hogeschool

EX/2016-17/2/9048/3

Programmeren 1 - Java: OO Technieken

Groep	Naam, Voornaam (student)	Studentnummer	Examencode
			201617290483-84

Vraag 3d – 6 punten (6%)
Het archief kan alle boeken in de dozen opvragen en sorteren op boekomvang.
Werk hiervoor de methode `public List<Boek> getGesorteerdeBoeken()` verder uit.

Tip 1: overloop alle dozen in het archief en gebruik de methode `public List<Boek> getBoeken()` uit **Vraag 2c**.

Tip 2: gebruik `addAll`

OUTPUT uitwerken – totaal van 15 punten (15%), subtotaal 100 / 100
`Archief.java`
Hier wordt het terug een beetje makkelijker. Punten sprokkelen dus.

Vraag 4a – 5 punten (5%)
Voorzie een inner klasse `Inventaris` in de klasse `Archief`. Instanties hiervan bevatten twee gegevens:

1. het aantal gebruikte dozen (`gebruikteDozen`)
2. het aantal gebruikte kokers (`gebruikteKokers`)

Voorzie nu in de klasse `Archief` de methode `private Inventaris getInventaris()` die een `Invertaris`-object aanmaakt met de correct berekende waarden.

Tip: voor de gebruikte hoeveelheden kan je beroep doen op `doosIndex` en `kokerIndex` uit **Vraag 3c.**

Vraag 4b – 10 punten (10%)
Zorg ervoor dat een `Archief`-Object op het scherm kan geprint worden. Bekijk wederom bovenstaande console-output op pagina 2. Je maakt **verplicht gebruik** van een `StringBuilder` object. Hou daarbij rekening met het feit dat een `Archief`-Object in de `TestArchief` klasse wordt uitgeprint met het commando `System.out.println(archief);`

Tip: Je hebt volgende methode nodig `List<Boek> boeken getGesorteerdeBoeken()` en `Inventaris inventaris = getInventaris().`

Einde!