Oefenopgaven hoofdstuk 3

Oefenopgave 1a

- Maak een nieuw project met de naam hfdst3
- Maak een class student in de package com.loi met de instance variabele naam.
- Maak de default constructor en getters en setters voor de naam.
- Voeg de class variabele wiskTotaal, aantalCijfers toe
- Neem de volgende import op: import java.util.Scanner;
- Neem de volgende instantie-variabele op: Scanner keybord;

`Zet de volgende code in de default constructor: keybord = new Scanner(System.in);`

- Voeg de volgende methode toe:

```
public void leesCijfer() {
            double cijfer = keybord.nextDouble();
            wiskTotaal += cijfer;
            aantalCijfers++;
}
```

- Maak een methode “double getGemiddelde()” waarin u het gemiddelde uitrekent en via de return teruggeeft.
- Voeg de methode `toString()` toe. Deze geeft de volgende output:
  `Karel heeft voor wiskunde gemiddeld het cijfer: 7.5`
  U moet in die methode gebruikmaken van de methode getGemiddelde();
- Maak een class TestStudent (met main-methode) waarmee u de class Student test.
  - In de main methode maakt u eerst een nieuwe student.
  - U gebruikt de setter om de naam in te vullen.
  - U roept 2x het leescijfer aan.
  - Als laatste zet u met System…. en toString het resultaat op het scherm.

Beantwoord de volgende vraag:
Waar staan de lokale variabelen in dit programma?

Oefenopgave 1b

Maak een class Kat. Een Kat heeft de volgende eigenschappen:
Naam, ras, leeftijd. Een kat kan miauwen.
Maak een class testKat. Maak drie katten aan: Tijger is een Siamees van 4 jaar. Puk is een Bengaal van 10 jaar. Sammie is een Europese Korthaar van 19 jaar. Laat alle drie de katten miauwen. Tijdens een miauw vertelt de kat zijn naam, ras en leeftijd.
Beantwoord de volgende vragen

a. Wat is de overeenkomst tussen Puk en Sammie?
b. Waarin verschillen Puk en Sammie?
c. Tonen Puk en Sammie verschillend gedrag?

Oefenopgave 2

De object lifecycle probeert u in een programma weer te geven.

- Maak een class exam zoals bij paragraaf 3.2.2.
- Voeg de default constructor toe.
- Voeg de methode finalize() als volgt toe: Insert code: kies voor Override Methods, kies: finalize()

Nu proberen we na te gaan wat er gebeurt wanneer er een object gemaakt wordt. Daarom voegen we de volgende code toe: System.out.println(xxxxx)

- In de constructor: `"de constructor wordt aangeroepen"`
- In setName: `"setName() is aangeroepen"`
- In finalize(): `"Het object wordt afgebroken"`

Achter aan de code voor de laatste accolade neemt u de volgende twee statements op:

- `static{ System.out.println("de static wordt uitgevoerd"); }`
- `{ System.out.println("de initializer wordt uitgevoerd"); }`

Het programma moet nog getest worden:

- Maak nu de class ObjectLife1
- Test het programma.

U ziet het volgende in de output:

```
de static initializer wordt uitgevoerd
de initializer wordt uitgevoerd
de constructor wordt aangeroepen
setName() is aangeroepen
de initializer wordt uitgevoerd
de constructor wordt aangeroepen
setName() is aangeroepen
de teller = 0
```

Wat opvalt, is dat de methode System.out.println in de methode finalize niet aangeroepen wordt.

In 3.2 staat: All Java classes can override the method `finalize`, which executes just before an object is garbage collected. In theory, you can use this method to free up resources being used by an object, although doing so, isn't recommended because **its execution is not guaranteed to happen**.

Dat gebeurt hier nu! Misschien komt dat omdat we Systeemfuncties aanroepen. Daarom breidt u de class exam uit met een teller.

- Voeg toe: `static int teller = 0;`
- Voeg `++teller;` toe voor en na super.xxx
- Voeg in ObjectLife1 op de laatste regel in main toe:

```
System.out.println("de teller = " + Exam.teller);
```

Voer de code uit en u ziet dat het niet helpt, de teller wordt niet verhoogd.

Wat kunnen we nu van deze code leren? De volgorde van het aanroepen in een classe/object

1. New Exam() wordt aangeroepen
2. Eerst wordt de static initializer aangeroepen.
3. Daarna de initializer.
4. Het object wordt gemaakt.
5. Daarna wordt de constructor aangeroepen.
6. De setName() wordt in ObjectLife1 aangeroepen.
7. myExam = null → wellicht wordt finalize() aangeroepen.
8. New Exam() wordt aangeroepen.
9. De initializer wordt aangeroepen (niet de static initializer).
10. De setName() wordt aangeroepen.
11. System.out.println("de teller wordt aangeroepen, deze is niet opgehoogd!
12. Main sluit af → wellicht wordt finalize() aangeroepen.

Oefenopgave 3a

Bij oefenopgave 1 hebt u een class Student gemaakt. Daarin staat in de code verschillende System-opdrachten, die u liever in de testfile zou zetten. Men wil graag algemene classes schrijven die u zowel in een windows-omgeving als op een console kunt uitvoeren.
Sloop deze constructies eruit. Verander leesCijfer in getCijfer en pas TestStudent aan.
U mag natuurlijk ook alles opnieuw maken.
In TestStudent maakt u nu een methode leesCijfer die een double van het toetsenbord leest en via return een double teruggeeft. U zult een constructor moeten schrijven.
In main moet u nu opnemen:

```
TestStudent test = new TestStudent();
```

De aanroep `student.leesCijfer();` wordt `student.setCijfer(test.leesCijfer());`

Vraag

U hebt waarschijnlijk in de class een static variabele opgenomen voor wiskTotaal en een aantalCijfers. Is dat terecht of moeten deze twee variabelen gewoon instance variabelen zijn?

Oefenopgave 3b

Voeg nu de variabelen klasWiskTotaal en klasAantalCijfers toe.
Pas de betreffende methode aan.
Maak een methode getKlasGemiddelde().
Roep deze methode getKlasGemiddelde() in de testfile aan.

Oefenopgave 4

Deze paragraaf gaat over overloading en dat passen we nu toe.

- Maak een class met de naam Convertor.
- Maak een static methode convertValue() waaraan u een double parameter meegeeft en een int terugkrijgt. U kunt door middel van een cast de double omzetten naar een int.
- Maak een static methode convertValue() waaraan u een int parameter meegeeft en een double terugkrijgt. U kunt door middel van een cast de int omzetten naar een double.
- Maak een static methode convertValue() waaraan u twee parameters meegeeft, een double waarin een getal staat en een int die aangeeft op hoeveel cijfers achter de komma u wilt afronden. U krijgt de afgeronde double terug.
  Stel dat u 1,98765 op drie cijfers wilt afronden, dan moet u terugkrijgen 1,988.
  Hoe doe u dat? Vermenigvuldig met 1000. Vervolgens 0,5 erbij optellen en het getal omzetten naar een int (een int kapt af, dus rondt niet af) en als laatste het getal weer vermenigvuldigen met duizend.
  Hoe komt u aan die duizend? Math.pow(10.0, aantalDecimalen).
- Maak een testclass.

Oefenopgave 5

Als uitgangspunt nemen we oefenopgave 4, de class Student.
Voeg de volgende constructoren toe:

- De default constructor (wanneer die nog niet aanwezig is).

Voeg de code toe: wiskTotaal = 0; aantalCijfers = 0;

- Een constructor waarbij u een naam meegeeft (genereren). U moet de default constructor aanroepen.

- Een constructor waarbij u een naam en een cijfer meegeeft (deze kunt u niet genereren). Voeg code toe: De naam komt in de variabele naam. Tevens roept u de methode set cijfer aan. U moet de default constructor aanroepen.
- Een constructor waarbij u een cijfer en een naam meegeeft. In de code roept u de constructor met naam en cijfer aan.
- U past de code van de class TestStudent aan.

Oefenopgave 6

In paragraaf 3.6.3 ziet u een methode daysoffWork(int… days)
Breid nu opgave 3.5 uit met de methode void setCijfer() waarbij u een willekeurig aantal cijfers kunt meegeven. U gebruikt een for net zoals bij daysoffWork(), alleen in plaats van daysOff += days[i]; roept u de methode setCijfer aan.
Test de opgave door aan de classTestStudent de methode setCijfer met 3 en een keer met 4 cijfers aan te roepen. U geeft de cijfers direct mee.
Nu probeert u ook op dezelfde manier een constructor te maken. U geeft de naam mee en meerdere cijfers.

Vraag
Hoe vaak wordt in de class Student een overloaded constructor gebruikt en hoe vaak een overloaded methode?

Vraag
Zou u zelf de class Student zo ontworpen hebben?

Oefenopgave 7

Hoe wordt encapsulation toegepast in de class Student?
Hoe wordt informaton hiding toegepast in de class Student?

Oefenopgave 8

Wat is het verschil tussen call by reference en call by value?

Maak een methode swap (student,student) in de class TestStudent. (zie boek)
Probeer dat uit en u ziet hetzelfde resultaat als in het boek. De namen van de student zijn niet veranderd.
Wanneer u de code van swap direct in main zet, wat gebeurt er dan?