Practicum OO — lente 2015 **Opgave 5**

Leerdoelen

Na het maken van deze opgave kun je

- subklassen van een (abstracte) klasse maken;
- werken met inheritance in Java;
- objecten opslaan in een *collection*; de objecten in een collection aflopen met een (enhanced) for-loop.

Quiz

In deze opgave ga je een programma maken dat de gebruiker een aantal quizvragen laat beantwoorden. Het gaat niet om het maken van een heel mooi interface of heel slimme vragen. Van belang is dat er verschillende soorten van vragen zijn en dat ieder van die soorten op een goede manier gerepresenteerd wordt in Java.

Deze quiz kent verschillende soorten vragen die elk een eigen soort informatie tonen en eigen invoer verwachten. Deze vragen worden één voor één aangeboden aan de gebruiker.

1 De vragen

Een vraag bevat tenminste de methoden:

```
public String toString()
public boolean isCorrect(String antwoord)
public String juisteAntwoord()
```

De `toString` methode zal een string opleveren die gebruikt kan worden om de vraag afdrukken. Met `isCorrect` kun je nagaan of een antwoord correct is. De methode `juisteAntwoord` geeft een string met het juiste antwoord als afdrukbare tekst, deze kun je bijvoorbeeld gebruiken als de gebruiker een verkeerd antwoord heeft gegeven. Definieer een (abstracte) klasse `Vraag` om de vragen in op te slaan. Deze klasse bevat minstens de genoemde methoden.

Iedere vraag heeft een vast `gewicht` dat voor iedereen zichtbaar is. Dit is een getal tussen 1 en 5. Indien het gewicht niet of verkeerd wordt opgegeven, heeft de vraag gewicht 3.

2 Open-vragen

Maak een subklasse voor open-vragen. Op een goed gestelde vraag is precies één correct antwoord. Dit antwoord wordt tijdens de constructie van de vraag aangegeven. De constructoren van deze klasse zijn

```
public OpenVraag(String vraag, String antwoord, int gewicht)
public OpenVraag(String vraag, String antwoord)
```

Op Blackboard staan een aantal open-vragen. Natuurlijk mag je daar zelf iets aan toevoegen.

Een lijst van vragen stellen

Omdat we geen grens willen opnemen aan het aantal vragen slaan we ze op in een `List`, in plaats van in een rij met vaste grootte om de vragen in op te slaan. We zullen zeer binnenkort details over lijsten bespreken. Tot die tijd kun je dit gebruiken als in onderstaand voorbeeld.

```
public void speel() {
    List<Vraag> vragen = new LinkedList<Vraag>();
    vragen.add(new OpenVraag("Wat is de complexiteit van binair zoeken?", "O(log N)", 2));
    vragen.add(new OpenVraag(
```

```
                "Hoeveel constructoren je minstens maken bij een klasse in Java?", "0"));
    for (int i = 0; i < vragen.size(), i += 1)
        System.out.print(vragen.get(i));
}
```

Het is eleganter om in plaats van de **for**-loop met een index `i`, een enhanced **for**-loop te gebruiken:

```
for (Vraag v: vragen)
    System.out.print(v);
```

Het programma zal de gebruiker om een antwoord vragen, dit antwoord inlezen en controleren. Bij het controleren van het antwoord wil je niet op hoofdletters controleren. Dit kan door de methode `equalsIgnoreCase` van `string` te gebruiken in plaats van de gewone `equals`.

3 Meerkeuzevragen

Maak vervolgens een klasse voor meerkeuzevragen. Deze vragen hebben een rijtje met antwoorden:

```
public Meerkeuze(String vraag, String [ ] antwoorden, int juisteAntwoord, int gewicht)
public Meerkeuze(String vraag, String [ ] antwoorden, int juisteAntwoord)
```

Het getal `juiste` geeft aan welk van de antwoorden in het rijtje correct is. Bij het afdrukken worden alle mogelijkheden gelabeld met letters `a`, `b`, `c` enz. Het goede antwoord bij een multiplechoicevraag is het label van het juiste antwoord. Voeg een aantal multiplechoicevragen toe aan je programma.

4 Twee-keuzevragen

Een speciale vorm van meerkeuzevragen zijn de twee-keuzevragen met korte antwoorden, bijvoorbeeld `Ja` en `Nee`, of `Goed` en `Fout`. Maak een subklasse voor deze vragen met constructoren:

```
TweeKeuze(String vraag, String antwoord1, String antwoord2, int juiste, int gewicht)
TweeKeuze(String vraag, String antwoord1, String antwoord2, int juiste)
```

Bij deze vragen worden niet alle opties gelabeld met een letter afgedrukt, maar teksten als

```
Ja of Nee: Is er in Java verschil tussen een interface en een abstracte klasse?
Goed of Fout: Iedere klasse definitie in Java bevat een constructor.
```

Het verwachte antwoord op deze vragen is dus een van de keuzen. Een aantal van de gegeven open vragen is eigenlijk zo'n twee-keuzevraag. Gebruik deze klasse in je programma.

5 Vragen herhalen

Om de gebruiker extra te laten oefenen wordem vragen die de gebruiker verkeerd heeft beantwoord, nogmaals gesteld nadat alle vragen voor de eerste keer zijn gesteld. We voegen deze vragen toe aan een lijst van nieuwe vragen. De klasse `Vraag` heeft daarvoor een methode `Vraag duplicate()` die het object zelf oplevert. Voor multiplechoicevragen maakt deze methode een nieuw `Vraag`-object waarin de volgorde van de antwoorden is veranderd, het is voldoende om alle antwoorden een pseudorandom hoeveelheid op te schuiven. Je kunt een pseudorandom hoeveelheid `shift`, met $0 \leq$ `shift` $<$ `bound`, bepalen met:

```
Random random = new Random();
int shift = random.nextInt(bound);
```

Natuurlijk moet ook het nummer van het juiste antwoord hieraan worden aangepast.

De twee-keuzevragen moeten *niet* aangepast worden door `duplicate`.

Na afloop krijgt de gebruiker een overzicht van het aantal behaalde punten met vragen die meteen juist zijn beantwoord, en het aantal punten dat in de tweede ronde is gescoord.

Inleveren

Je programma heeft minstens de vragen uit de voorbeeld bestanden, zo nodig met een beter type. Leveren **vóór zondag 8 Maart, 11:00 uur**, via Blackboard je Java bestanden in.