HOGESCHOOL

Java Fundamentals - Labo 10

Richtlijnen

- Onderstaande oefeningen worden in de Netbeans IDE uitgevoerd. Denk aan de debug mogelijkheden !
- Maak 1 project aan (met als naam Labo 10), voor elke nieuwe opgave maak je een nieuwe package aan binnen dit project.
- Wanneer je een iteratie moet schrijven kies jezelf welke lus het meest geschikt is, tenzij expliciet gevraagd.
- Vanaf nu zal je je programma opbouwen aan de hand van meerdere klassen. Je zal typisch 1 klasse schrijven waarin je een nieuw type definieert (zoals de klasse Voertuig). De tweede klasse bevat het gebruikersprogramma met de main methode.

Objecten en Methoden in Java

1. **Rechthoek** Schrijf een nieuwe klasse om een type voor rechthoeken aan te maken. De datavelden van deze klasse zijn :
   - een x coördinaat van de linker bovenhoek
   - een y coördinaat van de linker bovenhoek
   - een breedte
   - een lengte

   Voeg volgende methoden toe aan de klasse :
   - een methode `isVierkant` om na te gaan of de rechthoek ook een vierkant is. Laat deze methode als resultaat een boolean teruggeven.

- een methode `berekenOmtrek` die de omtrek van de rechthoek berekent en deze als resultaat teruggeeft.
- een methode `berekenOpp` die de oppervlakte van de rechthoek berekent en deze als resultaat teruggeeft.
- een methode `isIn` die twee parameters als input neemt, namelijk een x en y coordinaat en nagaat of deze coordinaten zich in de rechthoek bevinden (beschouw de randen van de rechthoek ook als ok). Geef een boolean terug als return waarde van deze methode.

Schrijf een gebruikersprogramma waarbij je een rechthoek aanmaakt en waarden toekent voor zijn data. Print nu enkele gegevens van je rechthoek uit : of het een vierkant is, wat de omtrek en de oppervlakte zijn. Vraag vervolgens de gebruiker de coördinaten van een punt in te geven totdat dit punt zich in je rechthoek bevindt.

2. **Rechthoek2** Voeg nu ook een constructor methode toe aan je rechthoek klasse zodat je rechtreeks objecten kan aanmaken door alle gegevens mee te geven als parameter van de constructor. Ga na of je standaard constructor (deze zonder parameters) ook nog werkt.

3. **Dobbelsteen** Herschrijf je programma `Dobbelen` uit Labo 8 als volgt : maak eerst een nieuwe klasse `Dobbelsteen` aan. Deze houdt volgende datavelden bij:

   - het minimum aantal ogen dat je kan werpen (d.i. 1 voor een klassieke dobbelsteen)
   - het maximaal aantal ogen dat je kan werpen (d.i. 6 voor een klassieke dobbelsteen)

   Voorzie ook volgende methoden in je klasse `Dobbelsteen` :

   - een constructormethode die de dobbelsteen initialiseert via 2 parameters, het minimum aantal ogen en het maximaal aantal ogen.
   - een methode `gooi` die het werpen met de dobbelsteen simuleert. Deze methode heeft geen parameters maar genereert telkens wanneer die opgeroepen wordt een nieuwe waarde voor de dobbelsteen en geeft deze waarde terug als resultaat. Deze waarde wordt toegekend aan je dataveld dat het aantal ogen van de laatste worp bijhoudt en wordt als resultaat van de methode teruggegeven.

In je programma `TestDobbelen` maak je nu een object aan van het type `Dobbelsteen` via je constructor methode. Deze dobbelsteen is speciaal namelijk het minimaal aantal ogen is 3 en het maximaal aantal ogen is 10! Tel opnieuw het aantal worpen dat nodig zijn om een vooraf ingegeven aantal ogen te werpen. Wanneer het gewenst aantal ogen niet tot de dobbelsteen behoort (groter dan maximaal aantal ogen of kleiner dan minimaal aantal ogen) dan geef je een error. Blijf input vragen tot een correct aantal ogen gevraagd wordt. Maak gebruik van de methode gooi() van de klasse Dobbelsteen. De output ziet er nog steeds hetzelfde uit namelijk :

```
Hoeveel ogen wil je werpen? [3,10]
15
Hoeveel ogen wil je werpen? [3,10]
10
Nieuwe worp = 12
Nieuwe worp = 3
Nieuwe worp = 9
Nieuwe worp = 6
Nieuwe worp = 10
In 5 pogingen werd een 10 gedobbeld
```

4. **GeheimeCode2** Herschrijf je programma GeheimeCode uit labo9 waarbij je de gebruiker een code van 5 cijfers laat raden. Wanneer de gebruiker een gok doet, antwoordt het programma met volgende gegevens : het aantal cijfers in de gok die op de juiste positie staan alsook de som van die cijfers. Bijvoorbeeld wanneer de geheime code 53840 is, en de gok van de gebruiker is 83241 dan antwoordt het programma met 2 en 7, namelijk cijfers 3 en 4 staan op de juiste plaats (aantal is dus 2) en hun som is 7. Laat de gebruiker een vast aantal keer een gok wagen.

   Je maakt nu een aparte klasse `GeheimeCode`, een nieuw type om met geheime codes te kunnen werken en een aparte testklasse waarin je je nieuwe klasse test (hierin zit de main methode dus). Als dataveld bevat de klasse `GeheimeCode` een array met hierin de geheime code bestaande uit 5 integers. Voorzie ook volgende methoden (die je zal oproepen in je testklasse):

   - een constructormethode waarbij je onmiddellijk een code kan meegeven als parameter.
   - een methode die telt hoeveel cijfers van de gok op de juiste positie staan. Geef de gok mee als argument en als resultaat geef je een aantal terug.
   - een methode om de som te berekenen van de cijfers die op de juiste

positie staan. Geef ook hier de gok mee als parameter van de methode en als resultaat geef je nu de som terug.

5. **TestKleinstePositiefVerschil** Schrijf een testprogramma (**DemoMeetGegevens**) dat een array van 20 meetwaarden, voorgesteld door positieve gehele getallen, inleest. Het programma moet controleren of deze waarden allemaal onderling verschillend zijn. Als dat het geval is, moet bepaald worden welke twee meetwaarden het dichtst bij elkaar liggen en het positieve verschil hiertussen moet uitgeschreven worden. Bvb. voor de rij (4, 2, 6, 1) is de output 1.

   Het testprogramma zal gebruik maken van een extra klasse **MeetGegevens**. Deze klasse bevat een array van meetgegevens (positieve gehele getallen) als dataveld. Deze klasse bevat daarnaast ook volgende methodes:

   - Een constructormethode waarbij je de meetgegevens kan meegeven als parameter (via een int array).
   - Een methode **zijnAlleElementenOnderlingVerschillend** die een boolean teruggeeft.
   - Een methode **bepaalKleinstePositiefVerschil** die een int waarde teruggeeft.