Assignment 30 – Shapes

Doel

Het doel van deze opdracht is het oefenen met het Java OOP-concept **abstract class**.

Achtergrond

Verschillende vormen (bijvoorbeeld een cirkel en een rechthoek) hebben gemeenschappelijk gedrag en eigenschappen.

Zo heeft een vorm een naam en kan het oppervlak en de omtrek worden berekend. Nu kan er voor elke vorm een specifieke klasse worden geschreven, maar omdat de vormen gemeenschappelijkheden, leent een **abstract class** hier uitstekend voor.

Overzicht

Deze opdracht heeft 5 classes. Maak deze aan op het moment dat erom gevraagd wordt.

- Class Assignment30 (main)
- Class Shape (abstract)
- Class Rectangle
- Class Circle
- Class Square

1. Voorbereiding

a. Maak een package aan met de naam **assignment30** (let op packages beginnen altijd met een kleine letter)

b. Maak in de package een nieuwe class met de naam **Assignment30** en maak de **main()** methode aan. De verdere implementatie van de **main()** methode wordt verder op in de opdracht besproken.

2. Class Shape (vorm)

We beginnen met het maken van de class **Shape**, omdat dit de super class is van de classes Rectangle, Circle en Square.

a. Maak in de package **assignment30** een nieuwe class met de naam **Shape** en geef als modifier het keyword **abstract** mee (om Java kenbaar te maken dat het een abstract class betreft).

b. De class **Shape** heeft 1 private field **name** van het type **String**.

c. De class **Shape** heeft een constructor met 1 parameter, welke de name zet bij het instantiëren van een nieuw object.

d. De class **Shape** heeft 3 **abstract** methoden:

- **draw()** met return type **void**
- **calculateArea()** met return type **double**
- **calculatePerimeter()** met return type **double**

- Let op deze methodes hebben geen implementatie.

e. Maak voor het field **name** de bijbehorende getter en setter.

3. Class Rectangle (rechthoek)

a. Maak in de package **assignment30** een nieuwe class met de naam **Rectangle**.

b. De class **Rectangle extend** van de class **Shape**

c. De class **Rectangle** heeft twee **private** fields, resp. **width** en **height**. Beide zijn van het type **double**.

d. De class **Rectangle** heeft een constructor met 2 parameters, welke de width en de height zet bij het instantiëren van een nieuw object.

e. Binnen de constructor wordt de naam **"Rectangle"** hardcoded gezet, door het aanroepen van de constructor van de **super** class **Shape**.

f. Tot op dit punt zal IntelliJ aangeven dat er iets niet oké is met de class Rectangle (rood onderstreept), dit is omdat de implementatie van de 3 methoden nog niet gedaan is maar wel moet (we voldoen nog niet aan het 'contract').

g. Implementatie methode **draw()**:

- Deze methode print de hardcoded tekst "Drawing a rectangle…"

h. Implementatie methode **calculateArea()**:

- Deze methode berekent de oppervlakte van de Rectangle volgens de onderstaande formule en geeft de berekende waarde terug.
  - **Oppervlakte = breedte * hoogte**

i. Implementatie methode **calculatePerimeter()**:

- Deze methode berekent de omtrek van de **Rectangle** volgens de onderstaande formule en geeft de berekende waarde terug.
  - **Omtrek = 2.0 * breedte * hoogte**

j. Maak de getters en setters voor de fields width en height.

4. Class Circle (cirkel)

a. Maak in de package **assignment30** een nieuwe class met de naam **Circle**.

b. De class **Circle extend** van de class **Shape**

c. De class **Circle** heeft 1 **private** field **radius**, welke van het type **double** is.

d. De class **Circle** heeft een constructor met 1 parameter, welke de **radius** zet bij het instantiëren van een nieuw object.

e. Binnen de constructor wordt de naam **"Circle"** hardcoded gezet, door het aanroepen van de constructor van de **super** class **Shape**.

f. Tot op dit punt zal IntelliJ aangeven dat er iets niet oké is met de class **Circle** (rood onderstreept), dit is omdat de implementatie van de 3 methoden nog niet gedaan is maar wel moet (we voldoen nog niet aan het 'contract').

g. Implementatie methode **draw()**:

- Deze methode print de hardcoded tekst "Drawing a circle…"

h. Implementatie methode **calculateArea()**:

- Deze methode berekent de oppervlakte van de **Circle** volgens de onderstaande formule en geeft de berekende waarde terug.
  - **Oppervlakte = pi * radius * radius**

- Voor de waarde van pi kun je gebruik maken van de constante PI in de class Math, welke standaard behoord tot de JDK. Je kunt deze gebruiken door **Math.PI**.

i. Implementatie methode **calculatePerimeter()**:

- Deze methode berekent de omtrek van de **Circle** volgens de onderstaande formule en geeft de berekende waarde terug.
- **Omtrek = 2.0 * pi * radius**

j. Maak de getters en setters voor de fields width en height.

5. Class Square (vierkant)

Maak in de package **assignment30** een nieuwe class met de naam **Square**. Laat deze extenden van de **abstract** class **Shape**. Bedenk zelf de verdere implementatie van de class **Square**, denk aan de field(s), constructor, override methoden en getters & setters.

De formules voor het berekenen van de oppervlakte en omtrek van een vierkant zijn:

- **Oppervlakte = zijde x zijde**
- **Omtrek = 4 x zijde**

6. Class Assignment30

Voordat we de **main()** methode gaan implementeren, gaan we eerst een methode schrijven die info uitprint over de shape.

6.1. Methode printShapeInfo()

Voor de leesbaarheid is het wenselijk om deze methode onder de main() methode te plaatsen.

a. Maak een **static** methode **printShapeInfo()**, welke drie parameters heeft, resp. **name**, **area** en **perimeter**. Waarbij **name** van het type **String** is, **area** en perimeter van het type **double**.

b. Print de naam, de oppervlakte en de omtrek van de shape naar de console:

- Maak gebruik van System.out.println()
- Maak gebruik van String.format, waarbij de **name %s** is, de **area %.2f** en de **omtrek %.3f**.

6.2. Methode main()

a. Instantieer van de class **Rectangle**, een nieuw object met de naam **rectangle** en geef een **breedte** van **20.5** en een **hoogte** van **30.1** mee aan de constructor.

b. Roep de methode **draw()** aan van het object **rectangle**.

c. Print de naam, de oppervlakte en de omtrek van het object **rectangle**:

- Maak gebruik van methode printShapeInfo()
- Roep voor de naam, het berekende oppervlakte en omtrek de overeenkomstige methoden aan van het object.

d. Instantieer van de class **Circle**, een nieuw object met de naam **circle** en geef een **radius** van **52.12** mee aan de constructor.

e. Roep de methode **draw()** aan van het object **circle**.

f. Print de naam, de oppervlakte en de omtrek van het object **circle**:

- Maak gebruik van de methode printShapeInfo()
- Roep voor de naam, het berekende oppervlakte en omtrek de overeenkomstige methoden aan van het object.

g. Instantieer van de class **Square**, een nieuw object met de naam **square** en geef een **zijde** van **20.0** mee aan de constructor.

h. Roep de methode **draw()** aan van het object **square**.

i. Print de naam, de oppervlakte en de omtrek van het object **square**:

- Maak gebruik van de methode printShapeInfo().
- Roep voor de naam, het berekende oppervlakte en omtrek de overeenkomstige methoden aan van het object.

j. Pas de grootte aan van de 3 objecten **rectangle, circle** en **square**.

k. Print opnieuw van elk object de naam, de oppervlakte en de omtrek met **printShapeInfo()**.

7. Methode isPositive

De vormen accepteren, volgens bovenstaande opdracht, negatieve waarden voor de fields width, height, radius en side. Dit is natuurlijk niet wenselijk.

- Controleer de waarde voordat deze gezet wordt naar het desbetreffende field.
- Indien deze negatief is, zet dan het desbetreffende field op 0 (nul).
- Implementeer een methode **isPositive**, welke de mee gegeven waarde controleert en **true** teruggeeft indien deze positief is en **false** indien deze negatief is. Gemakshalve gaan we ervan uit dat 0 (nul) positief is.
- Zie jij het voordeel van abstract classes en inheritance?
- Breid de main() methode uit door de waarden van de shapes aan te passen naar negatieve waarden.

8. Output

Onderstaand de output van het programma na volledige implementatie:

```
Drawing a rectangle...
The Rectangle has an area of 617,05 and a perimeter of 101,200
Drawing a circle...
The Circle has an area of 8534,12 and a perimeter of 327,480
Drawing a square...
The Square has an area of 400,00 and a perimeter of 80,000
The Rectangle has an area of 57,67 and a perimeter of 32,320
The Circle has an area of 4,75 and a perimeter of 7,728
The Square has an area of 80,64 and a perimeter of 35,920
The Rectangle has an area of 0,00 and a perimeter of 0,000
The Circle has an area of 0,00 and a perimeter of 0,000
The Square has an area of 0,00 and a perimeter of 0,000
```

9. Opmerking

Na het maken van de opdracht zul je wellicht opmerken dat bepaalde getters niet gebruikt worden. Deze mogen verwijderd worden, maar kunnen handig blijken wanneer een bepaald field opgevraagd moet worden.