Programmeren I

21 januari 2015

Algemene Richtlijnen

- Schrijf je naam bovenaan elk antwoordblad en kladblad. Schrijf niet met potlood of in het rood op je antwoordbladen.
- Gebruik voor elke vraag een afzonderlijk blad.
- Nummer alle antwoordbladen, en schrijf op het eerste blad het aantal antwoordbladen.
- Los elke (deel)vraag afzonderlijk op. Vermeld duidelijk het nummer van de (deel)vraag bij elk antwoord. Indien je een (deel)vraag niet beantwoordt, vermeld dan ook duidelijk het nummer van deze (deel)vraag samen met de melding 'geen antwoord'.
- Geef op het einde alles af (antwoordbladen, kladpapier, opgave).
- Schrijf verzorgd! We kunnen het ons niet veroorloven veel tijd te steken in het ontcijferen van het geschrift van alle studenten of het uitzoeken welke stukken van een oplossing niet doorstreept zijn. Als het niet duidelijk is, dan is het fout.

Richtlijnen Programmeren 1

- Voorzie voldoende uitleg bij je oplossing! Dat moeten geen volzinnen zijn.
- Om de oefeningen op te lossen heb je enkel de klassen en methodes nodig die we via het document *api.pdf* op Minerva hebben aangegeven. Je *mag* echter alle klassen uit de pakketten java.lang en java.util gebruiken als je die handiger vindt. Je mag ook alle methodes uit java.io.PrintStream gebruiken. Andere klassen uit de Java bibliotheek of andere bibliotheken mag je niet gebruiken.
- Lees eerst de volledige opgave van een oefening! Zo vermijd je dat je in een later deel van de oefening je oplossing moet herwerken.
- Bij *alle* code die je zelf schrijft is het belangrijk om code van een goede kwaliteit te schrijven. Het is niet voldoende dat de code correct werkt. Het kan zijn dat in sommige vragen bepaalde functionaliteit meerdere keren beschreven wordt. Het is aan jou om dit te detecteren en op gepaste wijze in je oplossing te verwerken.
- Bij vragen waar je zelf code schrijft mag je extra klassen en methodes toevoegen. In sommige gevallen kan dit nodig zijn om een meer eenvoudige en/of kwalitatief betere oplossing te geven.

1 Technische vragen (3 punten) ANTWOORD OP DIT BLAD

Geef voor elk van de onderstaande methode oproepen wat de return-waarde is.

new C().n() = **new** C().q() =

new A().q() = **new** X().r(**new** A()) =

new B().q() = **new** X().s(**new** C()) =

```
public interface I {
    public int m(int a);
}

public class A implements I {
    public int m(int a) {
        return 1;
    }
    public int n() {
        return 2;
    }
    public int p() throws Exception {
        throw new E();
    }
    public int q() {
        try {
            p();
        } catch (RuntimeException e) {
            return 0;
        } catch (Exception e) {
            return 1;
        }
        return −1;
    }
}

public class B extends A {
    public int m(int a) {
        a = 3;
        return 2;
    }
    public int p() {
        throw new IllegalArgumentException();
    }
}
```

```
public class C extends B {
    public int m(int a) {
        return 4 + super.m(a);
    }
    public int n() {
        int a = 2;
        int b = m(a) * 2;
        return a + b;
    }
    public int p() {
        return 7;
    }

}

public class X {
    public int r(I i) {
        return ((A) i).n();
    }
    public int s(I i) {
        return r(i);
    }
    public int s(B b) {
        return 5 * r(b);
    }
}

public class E extends Exception {}
```

2 Theorie (2 punten) ANTWOORD OP DIT BLAD

```
public class A {
   public X f() {...}
}

public abstract class B extends A {
   public abstract Y f();
}
```

```
public class X {
}
public class Y extends X {
}
```

Gegeven is de bovenstaande code voor klassen A, B, X, en Y. Je weet niet welke overige klassen aanwezig zijn; elke klasse in de bovenstaande code kan nog subklassen hebben. Je weet enkel dat alle code compileert zonder waarschuwingen. Kan de laatste lijn van methode m in het onderstaande voorbeeld een ClassCastException gooien? Indien ja, leg uit wat er fout kan lopen, en schrijf een oproep van methode m zodat de laatste lijn van m deze exception gooit. Indien nee, bewijs waarom dit niet kan.

```
public void m(B b) {
   X x = b.f();
   Y y = (Y) x;
}
```

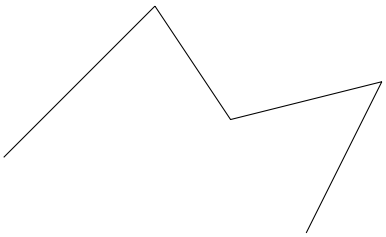
3 Polygonen (3 punten)

```
public class Point {
    private double x;
    private double y;
    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
    public Path to(Point other) {
        return new Path(this, other);
    }
}
```

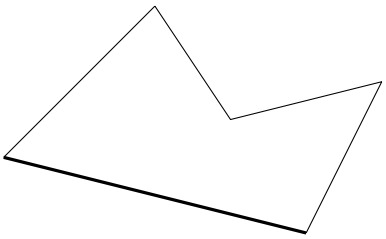
```
import java.util.*;
public class Path {
    private List<Point> points;
    public Path(Point first, Point second) {
        points = new ArrayList<Point>();
        points.add(first);
        points.add(second);
    }
    public Path(List<Point> list) {
        /* ... zie opgave ... */
    }
    public Path to(Point destination) {
        List<Point> points = new ArrayList<Point>(this.points);
        points.add(destination);
        return new Path(points);
    }
    public Polygon close() {
        return new Polygon(this);
    }
}
```

```
public class Polygon {
    private Path path;
    public Polygon(Path path) {
        this.path = path;
    }
}
```

Een Path stelt een opeenvolging van lijnstukken voor waarin elk punt verbonden is met het volgende punt van het pad. Een Polygon wordt gemaakt door het Path te sluiten zodanig dat het laatste punt van het pad verbonden is met het eerste punt van het pad, zoals grafisch weergegeven in de onderstaande figuur.



Path path = ...



Polygon polygon = path.close()

1. Schrijf de implementatie van de constructor Path(List<Point>). De punten van het Path moeten de punten zijn die in de gegeven lijst zitten. Schrijf de constructor zo dat de methode voor het berekenen van de omtrek van een Polygon (zie punt 3) geen exception kan gooien.

2. Teken het objectdiagram voor de waarde die berekend wordt door de **laatste** expressie van de onderstaande code. Als naar een object verwezen wordt door een variabele van het onderstaande code fragment, gebruik dan de naam van die variabele als naam van het object.

```
Point p1 = new Point(1, 2);
Point p2 = new Point(4, 7);
Point p3 = new Point(3, 6);
Point p4 = new Point(5, 0);
p1.to(p2).to(p3).to(p4).close();
```

3. Schrijf een methode die de omtrek van een polygon berekent. De afstand tussen twee punten (x_1, y_1) en (x_2, y_2) is gelijk aan $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Als je methodes of velden toevoegt aan een bestaande klasse, dan hoef je de bestaande code niet over te schrijven. Geef voor *alle* velden en methodes die je definieert duidelijk aan in welke klasse ze toevoegd worden!

4 Transport (6 punten)

1. Leg uit in termen van afhankelijkheden en cohesie waarom deze code slecht is.
2. Verbeter de code. Jouw code moet een *rekenAf* methode hebben met dezelfde functionaliteit als de huidige *rekenAf*. Jouw versie moet niet dezelfde parameters hebben.
3. Leg uit in termen van afhankelijkheden en cohesie waarom jouw versie beter is.

Veronderstel dat het veld *kilometerStand* voor elke nummerplaat het totaal aantal kilometers bevat dat die bestelwagen of vrachtwagen al gereden heeft. Veronderstel dat *IllegalVehicleTypeException* een subklasse is van *RuntimeException*.

```
import java.util.*;
public class VerhuurBedrijf {
    public static int BESTELWAGEN = 1;
    public static int VRACHTWAGEN = 2;
    public static Map<String, Double> kilometerStand = new HashMap<String, Double>();

    public static double rekenAf(int soort, String nummerplaat, int dagen, double kilometers) {
        if(nummerplaat == null) {throw new IllegalArgumentException();}
        double result;
        double totaleKilometers = kilometerStand.get(nummerplaat);
        if (soort == BESTELWAGEN) {
            if (dagen > 7) {
                result = 15 * dagen;
            } else {
                result = 25 * dagen;
            }
            result = result * Math.max((1 − totaleKilometers / 500000), 0.5);
            result = result + 0.6 * kilometers;
        } else if (soort == VRACHTWAGEN) {
            result = 500 * dagen;
            result = result * Math.max((1 − totaleKilometers / 1000000), 0.5);
            if(kilometers < 100) {
                result = result + 1.2 * kilometers;
            } else {
                result = result + kilometers;
            }
        } else {
            throw new IllegalVehicleTypeException();
        }
        kilometerStand.put(nummerplaat, totaleKilometers + kilometers);
        return result;
    }
}
```

5 Beeldverwerking (6 punten)

Een afbeelding bestaat uit een rechthoekig rooster van pixels. Elke pixel heeft een kleur die voorgesteld kan worden door drie kleurcomponenten: rood, groen, en blauw. Elke kleurcomponent heeft een waarde van 0 tot en met 255. De X-Y coördinaat van de pixel linksonder is (0,0). De breedte en de hoogte van een afbeelding zijn strikt positieve gehele getallen. Schrijf de nodige code om de onderstaande functionaliteit te voorzien. Zorg ervoor dat je code de juiste exceptions gooit indien men probeert om een zinloze operatie uit te voeren.

1. Je moet een afbeelding kunnen aanmaken met een gegeven breedte en hoogte. Initieel zijn alle kleurcomponenten van alle pixels gelijk aan 0.

2. Voorzie een methode die voor elke pixel de waarde van de groene kleurcomponent van de pixel vervangt door het gemiddelde van de groene kleurcomponenten van de aangrenzende pixels zoals die waren *voor* de methode werd opgeroepen.

3. Voorzie een methode die een derde partij toelaat om eender welke bewerking toe te passen op alle pixels van een afbeelding, en die voldoet aan de volgende voorwaarden:
   a) Jouw code mag niet weten hoe de nieuwe waarde van een pixel berekend wordt.
   b) De code die de nieuwe waarde voor een pixel berekent mag niet afhangen van de vorm van de afbeelding. Nu is een afbeelding rechthoekig, maar de bewerking moet ook blijven werken indien we een ronde afbeelding gebruiken, of een afbeelding waar gaten in zitten.
   c) De code die de nieuwe waarde voor een pixel berekent mag enkel gebruik maken van de *oorspronkelijke waardes* van die pixel en de aangrenzende pixels.
   d) De bewerking moet op eender welke afbeelding toegepast kunnen worden.

   Schrijf als voorbeeld een stuk code om de blauwe kleurcomponent van elke pixel te vervangen door het maximum van de blauwe kleurcomponenten van alle aangrenzende pixels, maar enkel indien de X-coordinaat een even getal is en de Y-coordinaat een oneven getal. Schrijf ook de concrete methode oproep om dit uit te voeren.