Programmeren I

29 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().x() = **new** C().y() =

new A().y() = **new** X().a(**new** B()) =

new B().y() = **new** X().b(**new** C()) =

```
public interface I {
    public int x(int a);
}
public class A implements I {
    private int r=7;
    public int x(int a) {
        return 1;
    }
    public int x() {
        return 7+ x(6);
    }
    public int y() {
        try {
            r = z(3);
        } catch (Error e) {
            r++;
        } catch (Exception e) {
            r = r * 2;
        }
        return r;
    }
    public int z(int g) throws Exception {
        r = r + g;
        throw new F();
    }
}
public class B extends A {
    public int x(int a) {
        a = 3;
        return 2;
    }
    public int z(int r) {
        r = r + 3;
        throw new IllegalArgumentException();
    }
}
```

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

}

public class X {
    public int a(I i) {
        return ((A) i).x();
    }
    public int b(B b) {
        return 7 + a(b);
    }
    public int b(I i) {
        return a(i);
    }
}

public class F extends Exception {}
```

2 Theorie (2 punten) ANTWOORD OP DIT BLAD

Als je in een accessor methode een object teruggeeft dat opgeslagen is in een veld, op basis waarvan beslis je dan of je een kopie van dat object teruggeeft of niet? Geef een voorbeeld voor beide gevallen, en motiveer in beide gevallen waarom je al dan niet een kopie teruggeeft.

3 Figuren (3 Punten)

```
public abstract class Shape {
    public abstract boolean contains(Point point);
    public Shape union(Shape other) {return new Union(this,other);}
    public Shape without(Shape other) {return new Difference(this,other);}
    public Shape intersection(Shape other) {return new Intersection(this,other);}
}

public class Point {
    private double x;
    private double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
    public Rectangle to(Point other) {return new Rectangle(this, other);}
    public Circle within(double radius) {return new Circle(this,radius);}
}
public class Rectangle extends Shape {
    private Point bottomLeft;
    private Point topRight;

    public Rectangle(Point bottomLeft, Point topRight) {
        this.bottomLeft = bottomLeft;
        this.topRight = topRight;
    }
}
public class Circle extends Shape {
    private Point center;
    private double radius;

    public Circle(Point center, double radius) {
        this.center = center;
        this.radius = radius;
    }
}
```

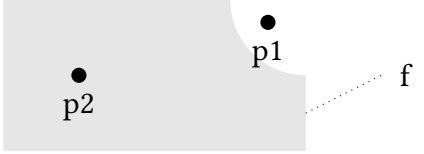
```
public abstract class Composition extends Shape {
    private Shape first;
    private Shape second;
    public Composition(Shape first, Shape second) {
        this.first = first;
        this.second = second;
    }
}
public class Union extends Composition {
    public Union(Shape first, Shape second) {super(first,second);}
}
public class Difference extends Composition {
    public Difference(Shape first, Shape second) {super(first,second);}
}
public class Intersection extends Composition {
    public Intersection(Shape first, Shape second) {super(first,second);}
}
```

Gegeven is de bovenstaande code voor het maken van figuren. Figuren kunnen gecombineerd worden door middel van de operaties: unie, verschil, en doorsnede.

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

```
Point c1 = new Point(1,3);
Point c2 = new Point(5,7);
Point c3 = new Point(3,2);
Point c4 = new Point(8,10);
c1.to(c2).union(c3.to(c4).intersection(c2.within(3))).without(c4.within(5));
```

2. Schrijf een methode om na te gaan of een gegeven punt in een figuur ligt. 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 toegevoegd worden!



p1 zit niet in f, p2 zit wel in f

4 Pakjes (6 punten)

De kerstman heeft software nodig voor het plannen van de bezorging van pakjes. Meer bepaald moet hij vooraf kunnen aangeven welk speelgoed in welke dozen en zakken verpakt zal worden. Voor speelgoed, dozen, en zakken gelden de volgende regels.

- Elk stuk speelgoed heeft een vast volume en een vast gewicht. Beiden zijn strikt positieve reële getallen.
- Een doos heeft een vast volume en een eigen gewicht. Het totale gewicht van een doos is gelijk aan de som van het gewicht van de doos zelf en dat van de inhoud van de doos. Een doos kan alle soorten objecten bevatten, inclusief andere dozen en zakken. Het volume van de inhoud van een doos kan echter niet groter zijn dan het volume van de doos zelf.
- Een zak heeft een maximum volume en een eigen gewicht. Het volume van de zak is gelijk aan het volume van de inhoud. Een zak kan alle soorten objecten bevatten, inclusief andere zakken en dozen. Het volume van de inhoud van een zak kan echter niet groter zijn dan het maximum volume van de zak. Het totale gewicht van een zak is gelijk aan de som van het gewicht van de zak zelf en dat van de inhoud van de zak.

Schrijf code die de onderstaande functionaliteit voorziet. Zorg ervoor dat je code de juiste exceptions gooit indien men probeert om ongeldige of zinloze operaties uit te voeren.

- Voorzie code om speelgoed, dozen, en zakken te maken en om dozen en zakken te vullen, rekening houden met de bovenstaande regels.
- Voorzie code om het volume van een stuk speelgoed, een doos, of een zak op te vragen.
- Voorzie code om de massa van een stuk speelgoed, een doos, of een zak op te vragen.
- Voorzie code die een buitenstaander toelaat om te tellen hoeveel voorwerpen in een zak of doos (inclusief deze zak of doos) voldoen aan een voorwaarde die wordt bepaald door deze buitenstaander. Hierbij moet je rekening houden met de volgende beperkingen:
  1. Jouw code mag niet weten wat de voorwaarde is.
  2. De code die controleert of een voorwerp aan de voorwaarde voldoet mag niet afhangen van het feit dat we zoeken in dozen en zakken. Ze moet zonder aanpassing bruikbaar zijn indien we een voorwerp zouden zoeken in bijvoorbeeld een warenhuis.
  3. De bewerking moet op eender welke doos of zak toegepast kunnen worden.

5 Spreadsheet (6 punten)

Een spreadsheet bestaat uit een rechthoekig rooster van cellen. Het aantal rijen en kolommen van een spreadsheet is constant en groter dan nul. Een cel in een spreadsheet kan leeg zijn, of een formule bevatten. Schrijf code die de onderstaande functionaliteit voorziet. Zorg ervoor dat je code de juiste exceptions gooit indien men probeert om ongeldige of zinloze operaties uit te voeren.

1. Je moet een spreadsheet kunnen aanmaken. Initieel zijn alle cellen leeg.
2. Je moet een cel kunnen vervangen door een nieuwe cel.
3. Je moet de waarde van een cel kunnen opvragen.
   a) Voor een lege cel is de waarde gelijk aan 0. Voor een cel met een formule wordt de waarde bepaald door de formule.
   b) Een formule kan een constante zijn, een sommatie, of een gemiddelde.
   c) Voor een som ligt vast van welke cellen de som berekend moet worden: een rechthoek waarvan je de rijen en kolommen van twee overstaande hoeken kent. Je weet niet welke van beiden hoeken links of rechts of boven of onder ligt.
   d) Voor een gemiddelde ligt vast over welke cellen het gemiddelde berekend moet worden: een rechthoek waarvan je de rijen en kolommen van twee overstaande hoeken kent. Je weet niet welke van beiden hoeken links of rechts of boven of onder ligt.
4. Formules in een spreadsheet kunnen zo geschreven worden dat bij het berekenen van de waarde een oneindige lus ontstaat. De onderstaande figuur illustreert dit met een voorbeeld. Je kan dit probleem voorkomen door bij het uitrekenen van een formule bij te houden welke formules je op dat moment *nog aan het uitrekenen bent.* Als je dan detecteert dat je tijdens het uitrekenen van een cel die cel zelf opnieuw gaat uitrekenen, dan kan je een fout genereren. Schrijf een tweede versie van je code voor het berekenen van de waarde van een cel zodanig dat ze niet in een oneindige lus geraakt (**of verwerk het in je eerste versie**).

$\sum(0,1)\ldots(0,4)$	1	2	3	$\sum(0,0)\ldots(0,3)$
(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)

   Merk op dat bij het uitrekenen van een cel, andere cellen meerdere malen mogen worden uitgerekend. Zolang de berekening van de waarde van een cel niet steunt op de waarde van die cel zelf, is er geen probleem. Het onderstaande voorbeeld illustreert dit.

$\sum(0,1)\ldots(0,4)$	$\sum(0,2)\ldots(0,4)$	1	2	3
(0, 0)	(0, 1)	(0, 2)	(0, 3)	(0, 4)