Faculteit der Exacte Wetenschappen — tentamen Inleiding Programmeren
Vrije Universiteit — 26 maart 2009, tijdsduur: 2:45 uur

Opgave 1.

a) Gegeven zijn de onderstaande classes

```
class A {
    int i;

    A (int i) {
        this.i = i;
    }
}

class B {
    A a;

    B (int x) {
        a = new A(x);
    }

    void put (int x) {
       a.i = x;
    }
}
```

en een programma dat de classes A en B gebruikt waarin de volgende declaraties/statements staan

```
A a = new A(5);
B b = new B(7);

void print (A a, B b) {
    out.printf("%d, %d\n", a.i, b.a.i);
}

void m1 (A p, int q) {
    p = new A(q);
}

void m2 (B p, int q) {
    p.a = new A(q);
}

void m3 (B p, int q) {
    p = new B(q);
}

void m4 (A a, int q) {
    a.i = q;
}
```

Wat is nu de uitvoer van een aanroep van de hierna gegeven methode doeIets() die in hetzelfde programma staat?

```
void doeIets () {
    print(a, b);
    m1(a, 11);
    print(a, b);
    m2(b, 13);
    print(a, b);
    m3(b, 17);
    print(a, b);
    m4(a, 19);
    print(a, b);
}
```

b) Gegeven is de onderstaande heading van een methode arctan()

```
double arctan (double x, int aantalTermen)
```

Deze methode moet m.b.v. "aantalTermen" van de onderstaande reeks benadert worden

```
arctan(x) = x - x^3/3 + x^5/5 - x^7/7 + x^9/9 - ......
```

Hierbij is x^n de notatie voor "x tot de macht n". Het aantal uit te rekenen termen is gelijk aan de waarde van de parameter aantalTermen. Programmeer deze methode zonder gebruik te maken van methodes uit de class Math. Ga uit van: aantalTermen >= 1.

c) Declareer een variabele "matrix" die een tweedimensionaal array van integers met 5 rijen en 9 kolommen bevat.
Doe dit gestructureerd, d.w.z. gebruik indien nodig constanten.

Schrijf een methode "dalend" die voor een willekeurige matrix van int's kan controleren of de reeks van waarden van de rijsommen dalend is.
De rijsom van rij i is de som van alle waarden in rij i.
De rijsommen zijn dalend als de rijsom van rij 1 groter is dan de rijsom van rij 2 en deze weer groter is dan de rijsom van rij 3, enz. t/m dat de rijsom van de voorlaaste rij groter is dan de rijsom van de laatste rij.

voorbeeld: De matrix 8 9 10 11 met rijsommen 38 is dalend.

4	5	6	7	22
1	2	3	4	10
0	0	0	0	0

d) Gegeven is het programma in de onderstaande class:

```
import java.io.PrintStream;
class Opgave1d {
    PrintStream out;
    int f, g;

    Opgave1d() {
        out = new PrintStream(System.out); f = 4; g = 7;
    }

    int methode1(int g) {
        int f = this.f + g;
        g *= 2;
        out.printf("%d\n%d\n", f, g);
        return this.f;
    }

    void methode2 (int f, int g) {
        this.f += f;
        g = this.g +1;
        methode1(this.f);
        out.printf("%d\n%d\n", f, g);
    }

    void start() {
        f = methode1(5);
        methode2(g, f);
        out.printf("%d\n%d\n", f, g);
    }

    public static void main(String argv[]) {
        new Opgave1d().start();
    }
}
```

Wat drukt dit programma af?

Opgave 2.

a) Gegeven zijn de onderstaande classes welke gebruikt kunnen worden voor het opslaan van gegevens van parkeergarages.

```
class Auto {

    String merk,
           nummerbord;
    int bouwjaar;

    // De constructors en methodes doen er voor deze opgave niet toe.
}

class Parkeergarage {

    static final int MAX_AANTAL_AUTOS = 875;

    Auto[] autoArray;
    int aantalAutos;

    ...

}
```

Schrijf een default constructor en een methode voegToe() voor de class Parkeergarage. De default constructor moet het Parkeergarage-object initialiseren op een lege parkeergarage met 0 auto's waaraan maximaal 875 auto's kunnen worden toegevoegd. De methode voegToe moet de mogelijkheid bieden om 1 Auto-object aan een Parkeergarage-object toe te voegen.

b) Voorzie de class Parkeergarage van een methode

```
Parkeergarage oudeAutos ()
```

welke in een nieuw Parkeergarage-object alle oude auto's retourneert die in de parkeergarage geparkeerd staan.
Voor deze opgave is gedefinieerd dat een auto oud is als het bouwjaar van deze auto voor 2000 ligt.

Indien je een deelprobleem herkent, programmeer dit dan in een aparte methode in de juiste class.

c) Gegeven is dat de class Parkeergarage een methode

```
int merk (String s)
```

bevat. Deze methode retourneert hoeveel van de in de parkeergarage geparkeerde auto's als merk de meegegeven string hebben.

Programmeer, gebruik makend van de methodes oudeAutos() en merk() en zonder gebruik te maken van een herhalingsopdracht, een methode

```
int klassiekers ()
```

die het aantal in de parkeergarahe geparkeerde klassieke auto's retoruneert.
Voor deze opgave is gedefinieerd dat een auto de eigenschap klassiek heeft, als het een oude auto van het merk "Renault" is.

d) Schrijf voor de class Parkeergarage een methode

```
void sleepWeg (String kenteken)
```

waarmee een auto kan worden verwijderd uit de parkeergarage.

Opgave 3.

a) Schrijf een recursieve functie aantalCijfers() die voor een willekeurig niet-negatief geheel getal n, de som van de cijfers van n kan bepalen.

b) Schrijf een recursieve functie som() die voor een willekeurig niet-negatief geheel getal n, de som van de getallen 0 t/m n kan bepalen.

c) Gegeven is dat de class String een methode substring() als volgt bevat:

```
String substring (int beginindex, int eindindex)
```

welke de substring van de gehele String retourneert vanaf de gegeven beginindex tot de gegeven eindindex, m.a.w. van beginindex t/m eindindex-1

Voorbeelden:

```
"12345".substring(0, 3).equals("123")
"12345".substring(2, 3).equals("3")
"12345".substring(1, 4).equals("234")
"12345".substring(0, 5).equals("12345")
"12345".substring(4, 4).equals("")
```

Verder is gegeven dat twee waarden, waarvan er 1 een String-waarde is, in java met de concatenatie-operator '+' aan elkaar geplakt kunnen worden.

Voorbeelden:

```
String s1 = '1' + "23",   // s1.equals("123")
       s2 = "" + '2',     // s1.equals("2")
       s3 = "12" + "34",  // s3.equals("1234")
       s4 = "1234" + '5'; // s4.equals("12345")
```

Schrijf nu een recursieve oplossing voor de onderstaande methode buitensteVooraan()

```
String buitensteVooraan (String s)
```

die van de meegegeven String s een nieuwe String maakt bestaande uit het eerste character van s gevolgd door het laatste character van s gevolgd door de characters tussen het eerste en laatste character van s waarbij op deze tussenliggende characters dezelfde bewerking wordt toegepast. Indien het eerste character ook het enige character is wordt dit character 1 keer aan de resultaatstring toegevoegd.

Voorbeelden:

```
buitensteVooraan("aba").equals("aab")
buitensteVooraan("abcdefghij").equals("ajbichdgef")
buitensteVooraan("12345").equals("15243")
buitensteVooraan("X").equals("X")
buitensteVooraan("").equals("")
```

Bij het schrijven van de methode buitensteVooraan() mag gebruik gemaakt worden van de gegeven methode substring().

Waardering

Opgave	a	b	c	d	totaal
1.	4	4	4	4	16
2.	2	4	2	4	12
3.	2	2	4		8
					-- +
					36

Het eindcijfer E volgt uit het puntentotaal T als volgt : E = T / 4 + 1