Opgave 1.

a) Gegeven zijn de onderstaande classes

```
class Naam {
    String s;

    Naam (String s) {
        this.s = s;
    }

    int aantalCharacters () {
        return s.length();
    }
}

class V {
    Naam naam;
    int i;

    V () {
        naam = new Naam("wim");
        i = naam.aantalCharacters();
    }

    void geefNaam (Naam n) {
        naam = n;
        i = naam.aantalCharacters();
    }
}
```

en een programma dat de classes Naam en V gebruikt waarin de volgende declaraties staan

```
Naam naam = new Naam("xyz");
V v = new V();

void m1 (V v, Naam n, String s1, String s2) {
    n = new Naam(s1 + s2);
    v.geefNaam(n);
    print(v, n);
}

void m2 (V v, Naam n, String s) {
    n.s = s;
    v = new V();
    print(v, n);
}

void print (V v, Naam naam) {
    out.printf("%s, %d\n", v.naam.s, v.i);
    out.printf("%s\n", naam.s);
}
```

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

```
void doeIets () {
    print(v, naam);
    m1(v, naam, "abc", "def");
    print(v, naam);
    m2(v, naam, "XXX");
    print(v, naam);
}
```

b) Gegeven is de onderstaande heading van een methode

```
double benaderLnXPlus1 (double x, int aantalTermen)
```

Deze methode moet ln(x+1) benaderen m.b.v. de volgende ontwikkeling

```
ln(x+1) = x - 1/2*x^2 + 1/3*x^3 - 1/4*x^4 + 1/5*x^5 - ....
```

Hierbij is x^i de notatie voor "x tot de macht i".
Het aantal uit te rekenen termen is gelijk aan de waarde van de formele parameter aantalTermen.
Programmeer deze methode zonder gebruik te maken van de class Math.
Ga uit van aantalTermen >= 1 en x > 0.

c) Declareer een variabele "matrix" die een tweedimensionaal array van integers met 15 rijen en 35 kolommen bevat.
Doe dit gestructureerd, d.w.z. gebruik indien nodig constanten.

Schrijf een methode "gebalanceerdeMatrix" die voor een willekeurige vierkante matrix van integers kan controleren of deze matrix de eigenschap "gebalanceerd" heeft.
Voor deze opgave wordt de "linkerdiagonaal" gedefinieerd als de elementen op de diagonaal van linksboven naar rechtsonder.
Eveneens alleen voor deze opgave wordt de eigenschap gebalanceerd gedefinieerd als dat de som van de getallen boven de linkerdiagonaal gelijk is aan de som van de getallen onder de linkerdiagonaal.

d) Gegeven is het programma in de onderstaande class:

```
import java.io.PrintStream;
class Opgave1d {
    PrintStream out;
    int a = -3,
        b = 5;

    Opgave1d() {
        out = new PrintStream(System.out);
    }

    int methode1(int a) {
        a = a + a;
        b = 3 * b;
        out.printf("%d\n%d\n", a, b);
        return b - a;
    }

    int methode2 (int b, int a) {
        a = methode1(a);
        b = b / a;
        out.printf("%d\n%d\n", b, a);
        return b;
    }

    void start() {
        a = methode1(a);
        b = methode2(b, b);
        out.printf("%d\n%d\n", a, b);
    }

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

Geef de uitvoer van dit programma.

Opgave 2.

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

```
class Student {

    String naam;
    double ectsPerJaar;
    int aantalJaarIngeschreven.

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

class Faculteit {

    static final int MAX_AANTAL_STUDENTEN = 1500;

    Student[] studentArray;
    int aantalStudenten;

    ...

}
```

Schrijf een default constructor en een methode voegToe() voor de class Faculteit. De default constructor moet het Faculteit-object initialiseren op een lege faculteit met 0 studenten waaraan maximaal 1500 studenten kunnen worden toegevoegd. De methode voegToe moet de mogelijkhied bieden om 1 Student-object aan een faculteit-object toe te voegen.

b) Voorzie de class Faculteit van een methode

```
Faculteit goedeStudenten ()
```

welke in een nieuw Faculteit object retourneert welke studenten van de faculteit goed studeren. Voor deze opgave is gedefinieerd dat een student goed studeert als het aantal ects per jaar van die student groter is dan 54.

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

c) Voeg vervolgens aan de class faculteit een methode topstudenten() toe die, in een nieuw Faculteit object, alle topstudenten retourneert. Voor deze opgave is een topstudent gedefinieerd als een goede student die minimaal 3 jaar is ingeschreven.

Gebruik de methode goedeStudenten() uit onderdeel b) voor de oplossing.

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

Opgave 3.

a) Schrijf een recursieve methode cijferSom() die de som van de cijfers van een positief geheel getal uitrekent.

```
Voorbeelden:    cijferSom(3)     = 3
                cijferSom(34)    = 7
                cijferSom(11111) = 5
```

b) Gebruik de functie cijferSom van onderdeel a) om een recursieve methode negenProef() te schrijven, die van een meegegeven natuurlijk getal test of dit getal voldoet aan de negenproef.
Een getal voldoet aan de negenproef als het deelbaar is door negen en de som van de cijfers van dat getal ook voldoet aan de negenproef.

c) Gegeven is de onderstaande hulpmethode

```
String aantalKeer (char c, int x) {
    String resultaat = "";
    for (int i = 0; i < x; i++) {
       resultaat += c;
    }
    return resultaat;
}
```

Geef een recursieve oplossing voor de onderstaande methode

```
String voegHaakjesToe (String s, int i)
// s bevat alleen cijfers, 0 <= i <= s.length()
```

die voor een String die alleen cijfers bevat, voor alle cijfers vanaf het cijfer op de positie i, ronde haakjes openen en sluiten toevoegt. Het aantal haakjes openen en sluiten is gelijk aan de waarde van het cijfer (bij het cijfer 6 worden er dus 6 ronde haakjes openen en 6 ronde haakjes sluiten toegevoegd). Alle ronde haakjes sluiten komen aan het eind van de resultaatstring.

Voorbeelden:

de aanroep	voegHaakjesToe("0", 0)	levert	"0"
	voeghaakjesToe("1", 0)		"(1)"
	voeghaakjesToe("2", 0)		"((2))"
	voeghaakjesToe("12", 0)		"(1((2)))"
	voeghaakjesToe("231", 0)		"((2(((3(1)))))))"
	voeghaakjesToe("231", 1)		"(((3(1))))"
	voeghaakjesToe("231", 2)		"(1)"
	voeghaakjesToe("231", 3)		""
	voeghaakjesToe("40", 0)		"((((40))))"

N.B. De waarde van de variabele i na de assignment

```
int i = '5' - '0';
```

is 5.

Waardering

Opgave	a	b	c	d	totaal
1.	4	4	4	4	16
2.	2	4	6		12
3.	2	2	4		8
					-- +
					36

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