Opgave 1.

a) Gegeven zijn de class

```
class X {

    String s;
    char c;

    X () {
        s = new String();
        c = 'x';
    }

    X (String string) {
        s = string;
        c = 'x';
    }

    void insert (char ch) {
        s = s + ch;
        c = ch;
    }

}
```

en de declaraties

```
Output out = new Output();
X xObject = new X("opgave 1a");

void m1 (X x) {
    x.s = x.s + "!";
    x.c = '!';
    out.println(xObject.c);
    out.println(xObject.s);
    x.insert('?');
    m2(x.s, x.c);
}

void m2 (String s, char c) {
    s = s + '.';
    c = '.';
}
```

Wat wordt er bij dit gegeven door de onderstaande statements afgedrukt?

```
out.println(xObject.c);
out.println(xObject.s);
m1(xObject);
out.println(xObject.c);
out.println(xObject.s);
```

b) Schrijf voor n>=1 een methode : int berekenReeks (int x, int n)
die de sommatie uitrekent van de eerste n termen van een reeks.
De i-de term (i>=1) van deze reekst is als volgt gedefinieerd.

```
((-1)^i) * i!)/(x^i)
```

Met de notatie a^b wordt "a tot de macht b" bedoeld.
Met de notatie a! wordt "a faculteit" oftewel 1 X 2 X ... X a bedoeld.
Het is niet toegestaan om methodes uit de class Math te gebruiken.

matrix m van gehele getallen het maximum van de kolomsommen van de kolommen met een oneven index retourneert. De kolomsom van een kolom is de sommatie van de getallen in die kolom.

```
int maxOnevenKolomsom (int[][] m)
```

Indien gewenst kan gebruik gemaakt worden van het feit dat de meest negatieve int waarde in Java gelijk is aan: Integer.MIN_VALUE

d) Gegeven is het programma in de onderstaande class:

```
class Opgave_1d {

    Output out;
    int a = -3,
        b = 2;

    Opgave_1d() {
        out = new Output();
    }

    void print (int a, int b) {
        out.println(a);
        out.println(b);
    }

    int methode1(int b) {
        b = b / a;
        a = b - a;
        print(a, b);
        return a;
    }

    int methode2 (int b, int c) {
        int a = b;
        b = a - c;
        c = methode1(a);
        print(a, b);
        return b;
    }

    void start() {
        a = methode1(a);
        b = methode2(a, b);
        print(a, b);
    }

    public static void main(String argv[]) {
        new Opgave_1d().start();
    }
}
```

welke uitvoer geeft dit programma?

Opgave 2.

a) Maak een class Auto voor het opslaan van de volgende gegevens over een auto: merk, cataloguswaarde, wel of niet openstaande boetes. Het merk kan een willekeurige rij van characters zijn.

Voorzie de class Auto van een constructor met parameters om alle velden van de class Auto op zinvolle waarden te initialiseren bij het aanmaken van een nieuw Auto object.

Denk er aan dat er voor iedere class altijd een default constructor gemaakt moet worden die alle variabelen een initiele waarde geeft.

b) Voor deze opgave kunnen op een snelweg maximaal 1.000.000 auto's rijden. Implementeer in Java een model van zo'n snelweg door een class Snelweg te maken waarmee een object gedefinieerd wordt waarin maximaal 1000000 Auto objecten kunnen worden opgeslagen. De class Snelweg bewaart geen informatie over welke auto waar hoe hard rijdt. Het enige dat onthouden wordt is welke auto's op de snelweg rijden.

Voorzie de class Snelweg van een methode voegToe() om een Auto object toe te kunnen voegen aan het Snelweg object.

c) Voeg aan de class Snelweg de onderstaande methode zoek() toe die, gegeven het merk van een auto, in een nieuw Snelweg object, alle auto's van dat merk retourneert die nog openstaande boetes hebben.

```
Snelweg zoek (String merk)
```

Pak dit gestructureerd aan. Voeg indien er behoefte is aan methodes die iets doen met een Auto object deze methodes toe aan de class Auto.

Opgave 3.

a) Schrijf een recursieve methode printXReeks() om de X-reeks voor gehele getallen >= 1 af te drukken, waarbij de X reeks van een getal gedefinieerd is als:

$$\text{X-reeks}(n) = \begin{cases} 1 & \text{als } n = 1 \\ \text{X-reeks}(n \text{ DIV } 2) \; n \; \text{X-reeks}(n \text{ DIV } 2) & \text{als } n > 1 \end{cases}$$

Voorbeelden:

n	x-reeks
1	1
2	1 2 1
3	1 3 1
4	1 2 1 4 1 2 1
6	1 3 1 6 1 3 1
8	1 2 1 4 1 2 1 8 1 2 1 4 1 2 1

b) Deze opgave gaat over int expressies in prefixnotatie, d.w.z. berekeningen met gehele getallen waarbij de uit te voeren operatie (b.v sommatie) niet tussen de operanden (zoals bij 2 + 3) maar voor de operanden staat (dus + 2 3).
De operanden kunnen zelf ook weer expressies in prefixnotatie staan.
De enige operatoren die voor deze opgaven gebruikt worden zijn + en * voor respectievelijk sommatie en product.

Voorbeelden van int-expressies in prefixnotatie zijn:

7	betekent	7
(+2,3)	betekent	(2 + 3)
(*4,5)	betekent	(4 * 5)
(*(+2,3),4)	betekent	((2 + 3) * 4)
(*(+2,3),(*4,5))	betekent	((2 + 3) * (4 * 5))

of, al wat ingewikkelder

```
(+(*3(+(*2,4),(+3,5))),(+(+10,7),6))
```

wat overeenkomt met

```
((3 * ((2 * 4) + (3 + 5))) + ((10 + 7) + 6))
```

Als gegeven is dat de leeswijzer het eerste character van een (NIET-lege) regel invoer aanwijst en dat de invoer een int-expressie in prefixnotatie bevat, schrijf dan een recursieve methode expressie() die de uitkomst van de int-expressie op de invoer retourneert. De heading van de methode expressie() is:

```
int expressie ()
```

HINTS:
Als we naar het voorbeeld (*(+2,3),(*4,5)) kijken dan zien we dat om de totale expressie uit te rekenen, de operanden (+2,3) en (*4,5) uitgerekend moeten worden. Deze twee operanden zijn zelf ook weer (kleinere) int-expressies in prefixnotatie. Hier herkennen we dus de recursiestap.
Bij het uitrekenen van b.v. (+2,3) hoeven de operanden 2 en 3 slechts ingelezen te worden om hun waarde te bepalen, m.a.w. als een operand niet met een '(' begint is de waarde ervan te bepalen met in.readInt(). Dit leidt tot de stopconditie.

Waardering

Opgave	a	b	c	d	totaal
1.	4	4	4	4	16
2.	2	4	6		12
3.	3	5			8
					-- +
					36

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