

Problem 1.

- a) Let the class A and B be

```
class A {
    int i;

    A (int q) {
        i = q;
    }

    void add (int x) {
        i = i*10 + x;
    }
}

class B {
    A a;

    B (A a) {
        this.a = a;
    }

    void add (String s) {
        for (int i = 0; i < s.length(); i++) {
            if (Character.isDigit(s.charAt(i))) {
                int digit = s.charAt(i) - '0';
                a.add(digit);
            }
        }
    }
}
```

Further we have a program with the following statements/declarations

```
A a = new A(5);
B b = new B(a);
PrintStream out = new PrintStream(System.out);

void println (A x, B y) {
    out.printf("%d,  %d\n", x.i, y.a.i);
}

void doSomething () {
    println(a, b);
    b.add("3");
    println(a, b);
    a.add(1);
    println(a, b);
    a = new A(7);
    println(a, b);
    b.add("5");
    println(a, b);
    b.a.add(8);
    println(a, b);
    b.a = a;
    b.add("a1b2c3");
    println(a, b);
    b.a.add(0);
    println(a, b);
}
```

What will be printed when the method doSomething() is called?

- b) The following heading of a method `cosinus()` is given

```
double cosinus (double x, int numberOfTerms)
```

This method should calculate the cosinus of x in `numberOfTerms` terms.

$\text{cosinus}(x) = x^0/0! - x^2/2! + x^4/4! - x^6/6! + x^8/8! - \dots$

x^n is the notation for "x to the power n" and $n!$ is the notation for "the factorial of n". The number of terms that should be calculated is given by `numberOfTerms`. Implement this method without using any methods from the class `Math`. Assume: `numberOfTerms` ≥ 1 .

- c) Give the declaration of a variable "matrix" with as type a 2-dimensional array of `char`'s with 5 rows and 3 columns. Use constants when necessary.

Program a boolean method `normal()` which will be able to tell, for any 2-dimensional array of `int`'s, whether this array has the property "normal"

For this problem a 2-dimensional array has the property normal when it contains no negative integers and when there are no rows that contain more zero's than positive integers.

Examples: array $\begin{pmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{pmatrix}$ is normal and $\begin{pmatrix} 0 & 0 & 0 & 10 \\ 2 & 5 & 7 & 1 \\ 3 & 6 & 9 & 12 \end{pmatrix}$ and $\begin{pmatrix} -1 & 2 \\ 7 & 9 \end{pmatrix}$ are not

- d)
- ```
class Problem_1d {
 PrintStream out;
 int p, q;

 Problem_1d() {
 out = new PrintStream(System.out);
 p = -3;
 q = 4;
 }

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

 void method1 (int p) {
 int q = p + 5;
 p = -p;
 print(p, q);
 }

 int method2 (int q) {
 int p = q + 1;
 q -= 1;
 method1(p);
 print(p, q);
 return q/p;
 }

 void start() {
 method1(q);
 print(p, q);
 q = method2(p);
 print(p, q);
 }

 public static void main(String argv[]) {
 new Problem_1d().start();
 }
}
```
- What will be printed when this program is executed?

Problem 2.

- a) Using the following class

```
class Plant {

 String division, class, order, family, genus, species;
 double usingWind;
 boolean weed;
 // The constructors en methods are omitted as they are not
 // necessary for this problem.
}
```

construct a class BotanicGarden. This class should be able to contain a maximum of 10000 plants. Further the class should have a default constructor and a method add().

The default constructor should initialize the BotanicGarden-object as an empty botanic garden. The method add() should make it possible to add 1 plant to the botanic garden.

- b) Program a method

```
BotanicGarden family (String f)
```

which, in a new BotanicGarden-object, return all the plants that are of the family f.

Program subproblems in separate methods in the correct class. Use constants when necessary.

- c) The following method can be assumed to be present in the class BotanicGarden. It can be used without having programmed it.

```
BotanicGarden usingWind ()
```

This method returns, in a new BotanicGarden-object, all the plants that use the wind as their means of reproduction.

Now add to the class BotanicGarden a method

```
BotanicGarden selectUsingWind (String f)
```

This method should return all the plants that are using the wind for reproduction and are of family f. Program the method selectUsingWind() without using a while-statement, a for-statement or a do-while-statement

- d) Add to the class BotanicGarden a method

```
void weed ()
```

This method should remove all weeds from the botanic garden.

Program subproblems in separate methods in the correct class. Use constants when necessary.

### Problem 3.

- a) Write a recursive method `minimum()` that for an arbitrary array of `int`'s will calculate the minimum value in that array

Don't forget that an array can contain 0 elements.

Think careful about which parameters the method `minimum()` should have.

This method may be programmed under the assumption that the array is fully filled.

HINT1: The minimum of 0 numbers is infinite, e.g. in Java, `Integer.MAX_VALUE`

HINT2: The minimum of two integers can be calculated with the method `min()` from the class `Math`.

`"Math.min(-5, 16) == -5"` and `"Math.min(10, 34) == 10"`

- b) Given is that the class `String` contains a method

```
public String substring (int start, int end)
```

that calculates the substring from `start` till `end-1` (inclusive).

```
examples: "abcdef".substring(3,6) gives "def"
 "abcdef".substring(1,4) gives "bcd"
 "abcdef".substring(0,3) gives "abc"
 "abcdef".substring(2,2) gives ""
```

Write a recursive method `"boolean palindrome(String s)"` that return `true` when the string `s` is a palindrome.

examples of palindromes:

```
""
"a"
"aa"
"pop"
"lepel"
"parterretrap"
```

Implement this method `palindrome()` without using any extra selfmade methodes.

grades:

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

The grade `E` follows from the point `P` with the formula:  $E = P / 4 + 1$