Kursus 02100 — Ugeseddel-Uge-06-Formiddag

I det følgende skrives arrays som `[1, 2, 3]` (bogen benytter mængdenotation).

Opgave 1

Løs nedenstående opgave.

Write a method called `lastIndexOf` that accepts an array of integers and an integer value as its parameters and returns the last index at which the value occurs in the array. The method should return `-1` if the value is not found. For example, in the array `[74, 85, 102, 99, 101, 85, 56]`, the last index of the value `85` is `5`.

Opgave 2

Løs nedenstående opgave.

Write a method called `countInRange` that accepts an array of integers, a minimum value, and a maximum value as parameters and returns the count of how many elements from the array fall between the minimum and maximum (inclusive). For example, in the array `[14, 1, 22, 17, 36, 7, 43, 5]`, for minimum value `4` and maximum value `17`, there are four elements whose values fall between `4` and `17`.

Opgave 3

Løs nedenstående opgave.

Write a method called `mode` that returns the most frequently occurring element of an array of integers. Assume that the array has at least one element and that every element in the array has a value between `0` and `100` inclusive. Break ties by choosing the lower value. For example, if the array passed contains the values `[27, 15, 15, 11, 27]`, your method should return `15`. (*Hint:* You may wish to look at the `Tally` program from this chapter to get an idea of how to solve this problem.)

Skriv dette program inden du fortsætter, og uden at sortere arrayet.

Can you write a version of the method that does not rely on the values being between `0` and `100`?

Dette kræver en omskrivning af programmet. Benyt metoden `sort` fra klassen `Arrays`, som ligger i pakken `java.util`. Kommandoen `Arrays.sort(a)` sorterer elementerne i arrayet `a`. Tillad også at arrayet har længden 0 (der kan så bare returneres 0).

Opgave 4

Løs nedenstående opgave.

Java's type `int` has a limit on how large an integer it can store. This limit can be circumvented by representing an integer as an array of digits. Write an interactive program that adds two integers of up to 50 digits each.

Det er i orden kun at håndtere ikke-negative heltal:

```
1st non-negative integer: 12345678901234567890123456789012345678901234567890
2nd non-negative integer: 12345678901234567890123456789012345678901234567890
Sum: 24691357802469135780246913578024691357802469135780
```

Benyt en konstant `DIGITS` med værdien `50` i programmet.

Det er i orden at kaste undtagelsen `RuntimeException` i tilfælde af "overflow" (altså over 50 cifre i resultatet). Navnet `RuntimeException` er lidt pudsigt, da alle undtagelser jo er "runtime" — men det er Java's navn for den mest generelle såkaldte "unchecked" undtagelse (se mere under "checked exception" i bogen).

PS:

Package `java.math` Description — For your information but must not be used in the exercise :-)

Provides classes for performing arbitrary-precision integer arithmetic (BigInteger) and arbitrary-precision decimal arithmetic (BigDecimal)...

Example

```
import java.math.*;

public class Factorial {
    public static void main(String[] args) {
        BigInteger n = BigInteger.ONE;
        for (int i = 1; i <= 90; i++) {
            n = n.multiply(BigInteger.valueOf(i));
            System.out.println(i + "! = " + n);
        }
    }
}
```

```
1! = 1
2! = 2
3! = 6
4! = 24
...
42! = 1405006117752879898543142606244511569936384000000000
...
```