Exercise 1: På CodeJudge under opgave 7.1 findes skelettet til klassen `FirstClass`. Den skal indeholde to private felter: en streng for dens navn (f.eks. `"beysg"`), og et tal for dens id (f.eks. `21`). I konstruktoren er disse parametre givet som argumenter. Metoden toString returnerer tallet og strengen på følgende form:
`"21 - beysg"`

End of Exercise 1

Exercise 2: I denne opgave laver du klassen `Positiv`, som indeholder et positivt tal. Lav selv et program i en anden klasse, som tester, at den opfører sig som tilsigtet.

a) Lav en klasse `Positiv`, som indeholder et positivt heltal (`int`).

b) Lav en konstruktor. Er der noget, som skal checkes?

c) Lav en accessormetode, der returnerer heltallet (`int`).

d) Implementer den offentlige metode addition, der tager en Positiv som parameter, og opdaterer det positive heltal til at være summen.

e) Gør tilsvarende for multiplikation.

f) Gør tilsvarende for subtraktion. Er der noget, som skal checkes?

End of Exercise 2

Exercise 3:

a) Write a Java class `Cat`, which implements cats. A cat is specified by its colour (e.g. orange), its year of birth (e.g. 1978), its name (e.g., Garfield), its owner (e.g., Harald), its biological sex (e.g. male), and how many times it has given birth to kittens. Define approriate fields for this.

b) Define a constructor. Is there something special that should be checked in the constructor?

c) Decide which fields should never be changed and ensure this.

d) Define and implement useful accessor methods.

e) (*) Decide which fields can be changed, how they can change, and define the appropriate mutator methods.

f) (*) Define a class `CatRegister` which implements a register which can hold up to 10 cats. The class should have methods to add a new cat (we do not allow to remove a cat) and to list all cats.

g) (*) Define a class `CatDemo` which creates a register and a number of cats, adds the cats to the register and prints a list of all cats in the register.

h) (**) Add a search method to `CatRegister`, which allows to search for owners. Given an owner, it returns the first cat who has this owner (if any). Test it from `CatDemo`.

End of Exercise 3

Exercise 4: Define a class `Person`. A `Person` object has two string fields `firstName` and `lastName` and an integer field `id`. In the constructor the two strings are given as arguments. Every `Person` object *automatically* receives an id-number starting with 1 and incrementetd by 1. The class ha a method `toString()` which returns a string consisting of the id first and last name. An example is shown below.

```
23: Jens Jensen
```

End of Exercise 4

Exercise 5: Define a class `Road`. A road object is specified by its length and width in metres, a name, and a surface type. The first two are represented as double the last two as string. All three are arguments in constructor in this order. The length cannot be changed, the other three parameters can. The width cannot be less than a meter. The only allowed surface types are `"asphalt"`, `"concrete"`, `"gravel"` and `"dirt"`. The `toString` method returns

```
Road <name> l=<length> w=<width> surface=<surface>
```

Length and width should be with two decimals. When one tries to create a road with illegal parameters then the name should be `"ILLEGAL"` and all other parameters set to zero or the empty string. Trying to change the parameters of an existing road to illegal values should result in printing the message `"No, this is not allowed."`. All parameters should be readable from other applications.

End of Exercise 5

Exercise 6: Write a class `TextAnalysis` which reads a text file and allows some text mining.

The class has to have a constructor

```
public TextAnalysis(String sourceFileName, int maxNoOfWords)
```

The argument `sourceFileName` is the name of the source file, if necessary with the path. The parameter `maxNoOfWords` is an upper bound on the number of words in the file which might help in your implementation.

Implement the following instance methods: (A *word* is a non-empty string consisting of only of letters (a,...,z,A,...,Z), surrounded by blanks, punctuation, hyphenation, line start, or line end.)

```
public int wordCount() // returns the number of words in the file
public boolean contains(String word) // returns whether word is contained
public String mostFrequent() // returns the most frequently occuring word.
```

End of Exercise 6