Problem 1

Define three enumerations:

1. **Sex** with constants F (female) and M (male);
2. **Size** with constants XS, S, M, L, XL;
3. **Country** with constants PL, NL and DE. Override the **toString** method for each constants separately, so it returns the name of the country in this country's language (Polska, Nederland, Deutschland).

Also, define a class **Person** with fields name, sex, size and country and overriding the **toString** method.

In the main class, besides the **main** function, define a *generic*, static function **printArray** taking a message and an array of any type to print on the screen.

The following program

download *EnumsLambdas.java*

```java
import java.util.Arrays;
import java.util.Comparator;

// enums, class Person

public class EnumsLambdas {

    // printArray static function

    public static void main(String[] args) {
        Person[] persons = {
            new Person("Max",  Sex.M, Size.XL, Country.NL),
            new Person("Jan",  Sex.M, Size.S,  Country.PL),
            new Person("Eva",  Sex.F, Size.XS, Country.NL),
            new Person("Lina", Sex.F, Size.L,  Country.DE),
            new Person("Mila", Sex.F, Size.S,  Country.DE),
            new Person("Ola",  Sex.F, Size.M,  Country.PL),
        };

        Comparator<Person> sexThenSize = /* lambda */;
        Arrays.sort(persons, sexThenSize);
        printArray("Persons by sex and then size", persons);

        Arrays.sort(persons, /* lambda */);
        printArray("Persons by size and then name", persons);
```

```
            Country[] countries = Country.values();
            Arrays.sort(countries, /* lambda */);
            printArray("Countries by name", countries);
        }
    }
```

should, after supplying missing definitions, print something like

```
   *** Persons by sex and then size ***
Eva(F, XS, Nederland)
Mila(F, S, Deutschland)
Ola(F, M, Polska)
Lina(F, L, Deutschland)
Jan(M, S, Polska)
Max(M, XL, Nederland)
   *** Persons by size and then name ***
Eva(F, XS, Nederland)
Jan(M, S, Polska)
Mila(F, S, Deutschland)
Ola(F, M, Polska)
Lina(F, L, Deutschland)
Max(M, XL, Nederland)
   *** Countries by name ***
Deutschland
Nederland
Polska
```

Problem 2

Collatz sequence (known also as *hailstone sequence* or Ulam sequence) is a sequence starting from a natural number a_0 and whose terms are calculated according to the rule $a_{n+1} = a_n/2$ for even a_n and $a_{n+1} = 3a_n + 1$ for odd a_n. There is a hypothesis that such a sequence will always reach 1 (and then will become periodic: $1, 4, 2, 1, 4, 2, 1, 4, \ldots$). It has been checked up to astronomically great numbers, but never proved.

For example, if we start from number 5, we get the sequence
$[5, 16, 8, 4, 2, 1, \ldots]$,
and starting from 7 the sequence will be longer:
$[7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, \ldots]$.

Your task is to create a class **Hailstone**, objects of which represent Collatz sequences. The constructor takes the starting number (a_0), which you may assume is a natural number larger than 1. The objects are *iterable*, i.e., the class implements interface **Iterable** and in each iteration returns next element of the sequence, starting from a_0. The iteration stops after returning, as the last value, the number 1.

Do not use any arrays, strings or collections.

Test your class by the following program:

download *SIterTW.java*

```
public class Main {
    public static void main(String... args) {
        int ini = 77031, count = -1, maxel = 0;
        Hailstone hailstone = new Hailstone(ini);
        for (int h : hailstone) {
            if (h > maxel) maxel = h;
            ++count;
        }
        System.out.println(ini + " " + count + " " + maxel);
    }
}
```

It should print, in one line and separated by spaces, three numbers: the starting value (ini, in this example 77031), number of steps until 1 is reached (count) and the value of the maximum element of the sequence (maxel). For example, if the starting value were 10, the sequence would be $[10, 5, 16, 8, 4, 2, 1]$, and therefore the three numbers printed by the program would be 10 6 16.
