# Clinic

Implement a system to manage patients of a medical clinic. The classes are hosted in package *`Clinic`*. The system must meet the following requirements.
**Warning**: Error handling is implemented with Swift 2 language (and Xcode 7.x)

## R1 - Patients

The main class of the program is class **`Clinic`**.
Patients are characterized by the first name, the last name, and the unique social security number (SSN). New patients can be added to the system by means of function **`addPatient()`**.
The information about a patient can be retrieved by means of function **`getPatient()`** that given an SSN returns an object of class **`Person`**. Such class provides the getter functions for SSN, first and last names. If the patient does not exist, error **`NoSuchPatient`** is thrown.

## R2 - Doctors

Doctors are characterized by the first name, last name, SSN, unique badge ID, and the specialization (e.g. "cardiologist", "dentist", etc.). New doctors can be added to the system by means of function **`addDoctor()`**.
Function **`getDoctor()`**, given a badge ID, returns an object of class **`Doctor`**. Such class that extends class Person and provide the getter functions for the id and specialization.
If the doctor does not exist, error **`NoSuchDoctor`** is thrown.
Keep in mind that doctors can be patients of the same clinic they work in.

## R3 - Patient registration

When accepted, a patient is assigned to one of the clinic doctors. To this aim, function **`assignPatientToDoctor()`** is provided. Patient is identified by means of her SSN, and doctor is identified by means of her badge ID. If the doctor does not exist, error **`NoSuchDoctor`** is thrown. Further, if the patient does not exist, error **`NoSuchPatient`** is thrown.
By means of function **`getDoctor()`** of class *`Person`*, it is possible to obtain the Doctor assigned to that person.
By means of function **`getPatients()`** of class *`Doctor`* returns the list of all patients of that doctor.

## R4 - Statistics

The function **`idleDoctors()`** returns the collection of doctors that have no patient at all, sorted in alphabetic order.
The function **`busyDoctors()`** returns the collection of doctors that are assigned a number of patients larger than the average.
The function **`doctorsByNumPatients()`** returns list of strings containing the name of the doctor and the relative number of patients with the relative number of patients, sorted by decreasing number.
The string must be formatted as "*`### : ID SURNAME NAME`*" where *`###`* represent the number of patients (printed on three characters).
The function **`countPatientsPerSpecialization()`** computes the number of patients per (their doctor's) specialization. The elements are sorted first by decreasing count and then by alphabetic specialization.
The strings are structured as "*`### - SPECIALITY`*" where *`###`* represent the number of patients (printed on three characters).