# Practice Exam OOS: Programming Part

**This part of the practice exam covers Java programming. It is meant to be solved in 3 hours. You can use a 4-page summary and the Java documentation. The programming part counts for 70% of your total score on the exam.**

## Problem description

A camping site wants to automate the management of its reservations. Currently the camping provides pitches (empty places where visitors can place their own tent or caravan) and fully equipped tents. However, they want the system to be easily extendable to other types like bungalows. Every available place on the camping has a code (a character + a number) and a capacity (maximum number of people that can use this place). For pitches, they want to store whether or not electricity is available. For tents, a list of possible extras that can be added to the tent (baby bed, microwave…) needs to be stored (each extra is a String). Also for tents, the year of their construction is stored.

The camping is open from May 1st until September 30th (included). Every place keeps the information about which day it is already occupied. People can only make a reservation for the current season. It should be possible to check if a place is free for a certain period, to make a reservation and to cancel a reservation. Periods are defined by a start day and an end day (=day you leave the camping site, so the place stays free), where days are numbered as integers: May 1st = 0, May 2nd = 1 and so on till September 30th = 152. They also want to know how many days a place has been occupied. The base price of a pitch is €12/day, for tents €50/day. However, for tents you get 1 day free for every week you stay. It should be possible to calculate the price for a given period.

A camping stores its name and address. Of course, you should be able to add places (of whatever type) and to know the total number of places. Places with the same code cannot be added twice. To facilitate reservations, it should be possible to get a list of all free places during a certain period. There should also be the possibility to check if a place with a given code is available on the camping.

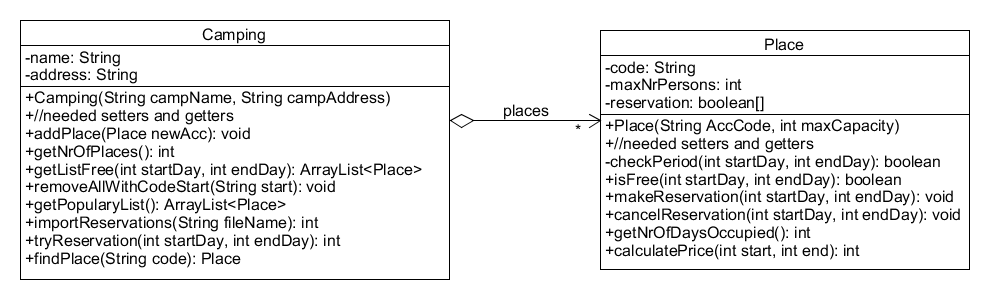
From time to time the organization of the camping site may change, so they want to have the possibility to remove all places where the code starts with a given combination. They are also interested in the popularity of the places, so they want a list of all places ordered by number of days occupied with least occupied first.

For regular customers, the camping allows to make reservations for the next year. This information is stored in a text file. So at the start of every new season, this file is used to import existing reservations. This method returns the number of imported reservations. If something goes wrong while importing it returns the negative value of the number of successful imports.

# Practice Exam OOS: Programming Part

## Implementation

Start from the project ‘campingstart’ form Toledo, and t from the following class diagram. ***You may NOT modify the signature of the provided methods nor the type of the fields. It is possible that you have to add extra attributes or methods to meet all the demands.***

******

As a start, no distinction is made between pitches and tents. Some more info about the tests

* Test3 checks if you can make and cancel a reservation and checks if you can calculate the number of occupied days with getNrOfDaysOcuupied(). For every method in which you need to make a reservation, you have to make sure that the given period is valid. Implement the needed tests in a separate method checkPeriod() and call it where needed. Don’t forget to check if a place is still free (method isFree()) if you want to make a new reservation. If something goes wrong when making or cancelling reservations, print a message to the screen.
* The format of the import file is as follows  
   - code of place  
   - start day  
   - end day  
  This structure repeats any number of times. The method returns the number of imported reservation (= number of periods). If the method finds invalid information it returns the number of imports that has already been done as a negative number and skips the import.
* In test10 we get a list of all places ordered by popularity, with least popular places first. The order of places with the same popularity is unspecified.
* The final challenge is to implement the method int tryReservation(int startDay, int endDay). With this method it should be possible to find a solution, even if there is not one single place free for the whole period. It will look for a solution using multiple places. You may start from the first day and look for the place with the highest availability. Then you repeat this for the day following this period and so on till you made a reservation the whole period. The method returns the number of places needed. If really no solution is found, the method returns -1.
* Keep in mind if a certain test does not pass that the reason may be twofold: you did something wrong in your syntax (like = instead of ==) resulting in different behavior or the logic you implemented is not correct or complete. Use available tools to check where the problem occurs.