Object-Oriented Programming
August 2021

Notes or documents of any kind forbidden. Duration: 2h30. Please answer the questions on separate sheets labeled with your name, section, and student ID.

1. The university needs to develop an application for managing class schedules. For this application, you are asked to program a Java class `RoomAssignments` suited for representing the courses assigned to a given room during a given week. This class must satisfy the following requirements:

   - An instance of `RoomAssignments` assigns one or zero course to each half-day of the week from Monday to Friday (i.e., Monday AM, Monday PM, Tuesday AM, Tuesday PM, ..., Friday AM, Friday PM).
   - A course is represented by an instance of a class `Course` that is supposed to be already available. (In other words, you are not asked to program it.)
   - It must be possible to create an empty instance of `RoomAssignments`, i.e., one in which every half-day is initially free.
   - It must be possible to assign a given course to a given half-day of an existing instance of `RoomAssignments`. The half-day is specified by providing a number corresponding to the day of the week (Monday = 1, Tuesday = 2, ..., Friday = 5), and a Boolean value with the meaning *false* = AM and *true* = PM. This operation succeeds only if the given half-day is still free.
   - It must be possible to check whether a given half-day is free or not for a given instance of `RoomAssignments` (with the same convention as in the previous operation).
   - It must be possible to compute the first half-day of the week that is free for a given instance of `RoomAssignments`. This operation fails if all half-days are already busy. You can freely choose how the result of this operation is returned.
   - Instances of `RoomAssignments` must be clonable, comparable to each other, and serializable. It must be possible to manipulate them simultaneously from separate threads. (You can assume that instances of `Course` are immutable and serializable.)
   - In case of any error, a dedicated exception should be thrown.

   Note: You are free to implement any additional classes required by your solution. Details not specified in this problem statement can be freely chosen. Your solution cannot rely on external data structures such as `Vector` or `HashMap`.

2. (a) How would you define a generic subclass `GenericRoomAssignments<T>` of `RoomAssignments` suited for assigning objects of type `T` to half-days (as opposed to only objects of type `Course`). (You do not need to fully program `GenericRoomAssignments<T>`; it is sufficient to explain what you would do.)

   (b) Which application of inheritance did you use in your answer to (a)? Is the substitution principle satisfied? (Justify your answer.)

3. You are asked to program in Java a class `AccessControl` for managing the access to a virtual meeting room. This class must satisfy the following requirements:

   - Each instance of `AccessControl` manages a separate meeting room.
   - An instance of `AccessControl` is characterized by an integer number $c > 0$, fixed when this instance is created, that represents the capacity of its meeting room, i.e., the maximum number of participants that may be present at the same time in this room.
   - The class provides two methods `enter()` and `leave()`, without argument or return value, invoked by the participants (from different threads) when they respectively enter and leave a meeting room.
   - If `enter()` is invoked when the corresponding meeting room is full (that is, when it has reached its capacity c), then this method waits until someone leaves the room.
   - For the sake of simplicity, you can assume that `leave()` is never invoked on an empty room.
   - It is important that the capacity of a room is never exceeded, and that anyone waiting to enter a room will eventually succeed.

   Suggestion: Use the same approach as for the communication channel of capacity one studied in the course.