Aalborg University
Department of Computer Science
Object-Oriented Programming
Autumn 2023

Trial Exam

Instructions:

- The exam **must be done individually**. Even if you (partially) use your solutions to some of the activities done during the semester as part of a group (e.g., the mini-project), the answers to this assignment have to be written by each student independently and any use of group solutions must be declared explictly.
- During the exam, you are **allowed** to use books, notes, slides, etc, but you cannot use tools that suggest solutions or partial solutions to the tasks.
- You are not allowed to contact physically or virtually anyone to get help with the assignment.
- The exam assignment must be completed within **eight (8) hours**.
- Your solution must include: (i) a ZIP file with the source code: .java files (or .cs files for those students that have taken this course before E23). (ii) a readme.txt file with instructions on how to compile and run your program (without using an IDE). (iii) a searchable PDF document that explains how your code provides solutions for each of the tasks in this assignment. The text must relate specific fragments of code to specific tasks and provide clear and concise answers to all the requests stated as part of the task descriptions. This PDF file must contain at most 19000 characters.
- The context of the tasks in this assignment is an application that allows for reserving rooms. A short description of the application is presented on page 2 and the tasks that you have to complete are described on pages 3-4. Please read all these pages before starting to work on your solution.
- An example of the CSV files is available in Moodle: `https://www.moodle.aau.dk/mod/folder/view.php?id=1651499`. You must support this format.
- Some of the classes must implement the Identifiable and TimeSlot interfaces. These interfaces are available at `https://www.moodle.aau.dk/mod/folder/view.php?id=1659226`.
- As part of your solution to this assignment you must write a program that follows the principles of the object-oriented programming paradigm as well as possible. The program must be written in Java if you are taking this course for the first time in E23. If you have taken this course before E23, your program must be written in C# or Java.
- You are encouraged to use classes or interfaces in the packages in the java.base module, e.g., java.util, java.lang, java.io (or .NET API types if you are using C#).
- If you decide to use other libraries besides those mentioned above, make sure that you understand what the library does and demonstrate that knowledge when providing answers to the tasks.
- The exam will be evaluated over 100 points and the passing grade is 50 points.
- You can provide a partial solution to a task and you will get a fraction of its value, but the fraction can be zero (0/N) if your answer is not relevant for the task.
- You can ask for clarifications about this assignment via email (gmontoya@cs.aau.dk).
- If you have to make an assumption to provide an answer, please **clearly state the assumption** next to your answer.

Booking System

Rooms can be reserved by groups to work on their group assignments. Reservations are requested in advance and stored as pending requests until an administrator asks the system to process those requests. Only available rooms can be reserved. The number of groups can be larger than the number of rooms.

You can choose to use the following assumptions:

- Each reservation is at most 120 minutes long.
- Rooms can be reserved from 8:00 until 17:00.

Tasks

1. **(12 points)** Define a class Reservation that represents the request done by a group to use a room from a start time to an end time. The class must implement the TimeSlot interface, override the toString (in C#: ToString) method inherited from Object, and provide the following method:

   - `public void approve()`: updates the time used by the group when the reservation is approved.

2. **(4 points)** Define a class FreeSlot that represents an available time slot from a start time to an end time. The class must implement the TimeSlot interface. The class must override the toString (in C#: ToString) method inherited from Object.

3. **(25 points)** Define a class Room that represents a room that can be reserved by groups. The class must implement the Identifiable interface. This class must have a calendar that includes the information of reserved and available time slots. This class must override the toString (in C#: ToString) method inherited from Object and provide the following methods:

   - `public boolean available(TimeSlot t)`: returns true if and only if the room is available at the time period represented by t.
   - `public void book(Reservation r)`: adds the reservation r to the room's calendar.
   - `public String toString(LocalDate date)`: string representation of the room including its calendar with busy and available times on a date.

   LocalDate is a type in the java.time package (in C#: DateOnly in the System namespace).

4. **(8 points)** Define a class Group that represents the groups that can reserve rooms. Each group keeps track of its group id, the time it has reserved, its number of members, and the student ids of its members. The class implements the interface java.lang.Comparable (in C#: System.IComparable) and provides the following methods:

   - `public void addTime(double d)`: increases the time that group has reserved.
   - `public void clearTime()`: sets to zero the time that the group has reserved. This method is to be used by administrators each time a new reservation period starts.

   Given groups g1 and g2, g1 precedes g2 if and only if g1 has reserved less time than g2. If g1 and g2 have reserved the same amount of time, then the group with more members precedes the other group.

5. **(4 points)** Modify the Reservation class to implement the java.lang.Comparable (in C#: System.IComparable) interface. Given reservations r1 and r2, r1 precedes r2 if and only if the group requesting r1 precedes the group requesting r2.

6. **(25 points)** Define a class BookingSystem that includes the following fields:

   - `private List<Room> rooms`: rooms that can be reserved
   - `private Map<String, List<Group>> groups`: course ids are mapped to the list of groups registered in the course
   - `private Map<Room, List<Reservation>> requestedReservations`: rooms are mapped to the reservations requested for the room

   Map, List are types in the java.util package (in C#: IDictionary, IList in the System.Collections namespace).

   BookingSystem has at least the following methods:

   - `public void bookInAdvance()`: allows to request the reservation of a room by a group
   - `public void showCalendars()`: shows the available and busy times of the rooms
   - `public void handleRequests()`: handles all the reservation requests in the order defined by the compareTo method

- `public void loadState()`: loads the information about rooms, groups, and reservations from the CSV files input\rooms.csv, input\groups.csv, input\reservations.csv
- `public void startNewReservationPeriod()`: sets the reserved time to zero for all the groups
- `public void addGroup()`: adds a new group of students to a course
- `public static void main(String[] args)`: allows the user to interact with the booking system, the user interactions may require any sequence of calls to the previous methods.

Your implementation of these methods must make appropriate use of the classes and methods you have written as solutions to the previous tasks.

7. **(4 points)** Given a reservation period where group oop-01 has reserved room 1.1.01 next Monday, Wednesday, and Friday from 9:00 until 11:00, group oop-02 has reserved 1.1.02 next Tuesday from 13:00 until 15:00, and group oop-03 hasn't reserved any room, the following requests are made:

   - group oop-01 requests the reservation of room 1.1.03 on Thursday from 9:00 until 11:00.
   - group oop-02 requests the reservation of room 1.1.03 on Thursday from 8:30 until 10:30.
   - group oop-03 requests the reservation of room 1.1.03 on Thursday from 9:30 until 10:30.

   When these requests are handled, which of them would be approved by your system? Justify your answer and possible deviations from what is described in this assignment.

8. **(5 points)** Identify at least one design pattern that you used in your solutions to the previous tasks. If you didn't use any design pattern, propose changes to your code to follow at least one design pattern (you do not have to update the code). Describe in a concise but precise manner the use of the design pattern, possible advantages and limitations.

9. **(8 points)** Assess how well your system ensures reservation consistency, i.e., rooms can only be reserved when they are available. If your system doesn't ensure reservation consistency, propose changes to your code to ensure reservation consistency (you do not have to update the code). Describe in a concise but precise manner the way your code (or proposed changes) prevents that inconsistent reservations are accepted and the possible feedback given to the client code and user interacting with the system.

10. **(5 points)** How well does the code that you wrote or used to implement the showCalendars method follow the principles of the object-oriented paradigm?