CENG 382 LAB 7

1. **Checkout to the lab6 branch you created last week with `git checkout lab6`.**
2. We will continue with the C# project, so you must create a new branch using the lab 6 branch. From the lab6 branch, create the lab7 branch **`git checkout -b lab7`.**
3. In this lab, you must convert the Room and Reservation class to record data type to make them immutable. And also create LogRecord with record datatype. These three classes are your containers for data.
4. Then, create each class using the structure given to you with the UML file. Each class must be in a different file. Create a .cs file for each class.
5. Create ReservationData.json and LogData.json files.
6. Do not commit all of your changes in one go. Consider this lab a significant upgrade in your professional work and act accordingly.
7. Research the Single Responsibility Principle and Dependency Injection Principles. Please explain in the comment section why your code from last week fails to satisfy this principle and why it is essential to use these principles in web applications. **(20 points).**
8. In every add or delete Reservation operation, take a log for that action and update LogData.json.
9. After adding and deleting Reservation, update ReservationData.Json
10. You must also output your logs and reservations to a different JSON file and commit these files.
11. Provided UML is not the optimal solution, it creates too many objects And breaches Dependency Injection Principles and Single Responsibility Principles. Adjust your code without breaching these principles. **(20 Points).**

Explanation of Classes :

- Room: Contains a record definition for Room with properties like ID, name, and capacity.
- Reservation: Contains a record definition for Reservation with properties like Time, Date, ReserverName, and Room.
- LogRecord: Contains a record definition for LogRecord with properties like Timestamp, ReserverName, and RoomName.
- ILogger: Defines a single method Log(LogRecord log) for logging log records.
- FileLogger: Implements the Log method from ILogger, which logs log records into JSON files.
- LogHandler: Contains a method AddLog(LogRecord log), which adds a log record using a provided ILogger instance.
- IReservationRepository: Defines methods AddReservation, DeleteReservation, and GetAllReservations for managing reservations.
- ReservationRepository: Implements methods of IReservationRepository to add, delete, and retrieve reservations stored in memory.
- RoomHandler: Contains methods GetRooms and SaveRooms for reading and writing room data to a JSON file.

- ReservationHandler: Manages reservation operations like adding, deleting, and retrieving reservations and also includes methods to interact with room data.
- IReservationService: Defines methods AddReservation, DeleteReservation, and DisplayWeeklySchedule for reservation service operations.
- ReservationService: Implements methods of IReservationService to handle reservation-related functionalities, delegating to ReservationHandler as needed.
- Program: Contains the Main method where the system components are instantiated, and reservation functionalities are tested.

Useful Links :

- https://learn.microsoft.com/en-us/dotnet/csharp/tutorials/records
- https://stackoverflow.com/questions/16921652/how-to-write-a-json-file-in-c
- SRP : https://blog.ndepend.com/solid-design-the-single-responsibility-principle-srp/#:~:text=The%20Single%20Responsibility%20Principle%20(SRP)%3A%20A%20class%20should%20have,extension%20and%20closed%20for%20modification.
- Singleton Design Pattern: https://www.c-sharpcorner.com/UploadFile/8911c4/singleton-design-pattern-in-C-Sharp/
- DI :https://www.c-sharpcorner.com/article/solid-principles-in-c-sharp-dependency-inversion-principle/#:~:text=The%20Dependency%20Inversion%20Principle%20(DIP)%20states%20that%20a%20high%2D,details%20must%20depend%20upon%20abstractions.
- Factory Design Pattern: https://hackernoon.com/understanding-the-factory-pattern-in-c-with-examples