Esercitazione di Laboratorio

Ingressi Aule

È necessario scrivere un'applicazione che simula il frammento di un sistema informativo per la gestione di ingressi alle aule di un'università. Ogni aula è identificata da (a) codice univoco di esattamente 4 caratteri b) il nome dell'aula (c) il piano. Ad esempio alcuni possibili aule sono {"A1", "Laboratorio Didattico", piano 2} e {"AMD", "Aula Magna DiMIE", piano 0}. Per ogni aula bisogna registrare gli accessi degli studenti. Un accesso è descritto da (a) la matricola (b) il nome dello studente (c) il tempo della permanenza in minuti (d) la motivazione (e) data (completa di giorno, ore e minuti) dell'accesso. Le possibili motivazioni "lezione", "ricevimento", "esame". Ad esempio, un possibile accesso è {12345, "Mario Rossi", 120min, "esame", 6 maggio 2021 15:30}. L'applicazione deve consentire di svolgere i seguenti casi d'uso:

"**Utente carica un archivio"**

- l'utente carica l'archivio *NOTA: per velocizzare le operazioni, si suggerisce di utilizzare un "mock object" per simulare il funzionamento del DAO che carica i dati dal disco*

"**Utente cerca aule per piano"**

- l'utente inserisce il numero del piano
- il sistema mostra la lista delle aule presenti su quel piano, o sui piani inferiori. La lista dev'essere ordinata per descrizione crescente

"**Utente inserisce nuovo accesso"**

- l'utente seleziona un'aula dai risultati
- il sistema mostra, in un nuovo pannello, i dati dell'aula, e gli accessi attualmente registrati. Per ogni accesso bisogna mostrare data (giorno, mese, anno, ore e minuti), matricola, durata e motivazione
- l'utente può decide di inserire i dati di un nuovo accesso, specificando tutti i dati. *Nota: Utilizzare cinque campi separati, giorno, mese e anno, ora e minuti, per leggere l'orario dell'accesso*

Scenario alternativo

- dati scorretti: il sistema mostra un messaggio di errore all'utente
- orario scorretto: se l'ora inserita è successiva al momento dell'inserimento bisogna mostrare un messaggio di errore all'utente

"**Utente verifica archivio"**

- utilizzando una voce di menu, l'utente può chiedere al sistema di verificare se di domenica ci sono stati accessi duplicati (ovvero avvenuti dalla stessa matricola e nello stesso giorno, senza considerare ora e minuti)
- il sistema effettua la verifica e mostra un messaggio all'utente

---

Requisiti facoltativi per il livello intermedio

"**Utente ricerca mesi frequenti"**

- il sistema calcola, per ogni mese dell'anno, qual è la tipologia di motivazione più utilizzata in quel mese, e il tempo di permanenza totale.
- i dati saranno mostrati in una tabella ordinata per mese crescente. Un esempio di tabella è la seguente

Mese	Tipologia	Tempo Totale
Gennaio	Lezione	60min
Febbraio	Esame	120min

Requisiti facoltativi per il livello avanzato

"**Utente salva archivio"**

Scenario principale

- L'utente richiede il salvataggio dell'archivio generato specificando un nome di file
- Il sistema salva l'archivio utilizzando il formato jso
  NOTA*: l'operazione di salvataggio dev'essere fatta in modo da non bloccare l'interfaccia utente*

"**Utente carica l'archivio da file"**

Scenario principale

- l'utente carica un archivio selezionando dal disco un file in formato json. ***NOTA**: l'operazione di caricamento dev'essere fatta in modo da non bloccare l'interfaccia utente Si suggerisce di generare un file json da caricare a partire dai dati mock*

"Test di Regressione"
Implementare i test di regressione con Junit per il caso d'uso "Utente verifica archivio" "Utente ricerca mesi frequenti", "Utente salva archivio" e "Utente carica l'archivio da file"

Utilizzare **Gradle** come sistema di costruzione del codice

Per le librerie, utilizzare le seguenti dipendenze
- org.slf4j:slf4j-api:1.7.30
- ch.qos.logback:logback-classic:1.2.3
- ch.qos.logback:logback-core:1.2.3
- com.google.code.gson:gson:2.8.6
- org.jdom:jdom2:2.0.6'
- junit:junit:4.12

Sviluppare l'applicazione che implementa i casi d'uso elencati, seguendo il processo di sviluppo descritto a lezione, e in particolare le seguenti operazioni:
- costruire e documentare il modello concettuale dell'applicazione
- sviluppare il frammento di applicazione che implementa i casi d'uso descritti
- sviluppare l'interfaccia grafica in **Java Swing** utilizzando **l'architettura MVC**
- per effettuare operazioni di logging utilizzare le librerie SLF4J e LogBack
- come sistema di costruzione del codice usare **NetBeans standard** in caso di base/intermedio. Per il caso d'uso avanzato utilizzare un progetto basato su Gradle

E' possibile trovare il materiale necessario nelle seguenti cartelle
- librerie jar: **c:\lib**
- materiale didattico: **c:\documentazione\POO1 e POO2**
- file di esempio di logback.xml: **c:\documentazione\POO1 - materiale software aggiuntivo\java**