# *Terzo Appello - Prova di Livello Base-Intermedio*

*Cognome e Nome:* _____ *Matricola*: _____

**Tempo a disposizione: 3 ore**

È necessario scrivere un'applicazione che simula il frammento di un sistema informativo per gestire gli orari di ricevimento dei docenti di un corso di laurea. Ogni docente ha un nome, cognome, l'argomento principale che insegna. Ad esempio, un possibile docente è {"Michele", "Bianchi", "psicologia" }. Per ogni docente bisogna tener traccia delle prenotazioni effettuate per il ricevimento. Ogni prenotazione ha la data e ora di inizio, la data e ora di fine e la matricola dello studente. Ad esempio, una possibile prenotazione è { inizio 01/09/2024 16:30, fine 01/09/2024 16:45, matricola 54321}.

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 docenti"**

- l'utente inserisce
  - una data e ora (giorno, mese, anno, ore minuti) *NOTA: per la data utilizzare cinque JTextField, per il giorno, mese e anno, ore e minuti. I campi devono essere pre-popolati con la data e ora correnti.*
  - un argomento
- il sistema mostra in una tabella la lista dei docenti con le seguenti caratteristiche:
  - hanno come argomento quello scelto;
  - sono liberi in quell'orario, ovvero non hanno alcuna prenotazione in quell'orario (nota: nell'esempio precedente, il docente Michele Bianchi risulterebbe impegnato alle 01/09/2024 16:40 in quanto ha un ricevimento che inizia alle 16:30 e finisce alle 16:45).
- la tabella che mostra la lista dei docenti deve essere ordinata per cognome crescente.

"**Utente inserisce nuova prenotazione"**

- l'utente seleziona un docente dai risultati
- il sistema mostra, in un nuovo pannello, i dati del docente
- il sistema calcola e mostra inoltre, per il docente selezionato e per ogni giorno della settimana il numero di prenotazioni ricevute. I dati saranno mostrati in una tabella ordinata per numero di prenotazioni crescente. Per il giorno usare il nome esteso nella localizzazione dell'utente. Un esempio di tabella è la seguente

| **Giorno** | **Numero di prenotazioni** |
|---|---|
| *Mercoledì* | 3 |
| *Giovedì* | 2 |

- l'utente può decide di inserire i dati di una nuova prenotazione, specificando solo la matricola e la data e ora di inizio. La data di fine dev'essere calcolata automaticamente considerando un ricevimento di 15 minuti. Dopo aver effettuato l'inserimento, il sistema aggiorna i dati della tabella sulle statistiche

Scenario alternativo

- dati scorretti: il sistema mostra un messaggio di errore all'utente

"**Utente verifica archivio"**

- selezionando una voce di menu, l'utente calcola il docente che ha avuto più prenotazioni da studenti diversi (ovvero matricole diverse)
- il sistema mostra un messaggio con il risultato calcolato al passo precedente. Ad esempio "Il docente Mario Bianchi ha ricevuto 5 prenotazioni da studenti diversi""

---

"**Utente calcola periodi pausa" – Per la prova di livello intermedio**

- il sistema calcola il periodo più lungo nel quale non c'è alcuna prenotazione in archivio
- il sistema mostra il messaggio con il risultato calcolato. Ad esempio: "Il periodo più lungo senza prenotazioni va da 1 agosto 2024 al 15 settembre 2024"
- ***Nota****: è necessario sviluppare i relativi test di regressione*

"**Utente carica un archivio JSON" – Per la prova di livello intermedio**

- Modificare il caso d'uso "Utente carica un archivio" per permettere all'utente di caricare l'archivio da un file json, facendolo selezionare dal disco

- ***Nota:*** *per inizializzare il file si consiglia di scrivere un test per salvare il contenuto dell'archivio mock in un file json*

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** presentata a lezione, rispettando i requisiti per l'accessibilità e l'usabilità.
- sviluppare i test di regressione utilizzando la libreria JUnit
- per effettuare operazioni di logging utilizzare le librerie SLF4J e LogBack
- utilizzare **Gradle** come sistema di costruzione del codice

Per le librerie, se necessario, utilizzare le seguenti dipendenze

- org.slf4j:slf4j-api:2.0.1
- ch.qos.logback:logback-classic:1.3.1
- ch.qos.logback:logback-core:1.3.1
- org.junit.jupiter:junit-jupiter:5.8.2

Per utilizzare Lombok, applicare il seguente plugin gradle

- id "io.freefair.lombok" version "6.5.1"

E' possibile trovare il materiale necessario nelle seguenti cartelle

- materiale didattico: **c:\documentazione\POO1 e POO2**
- file di esempio di logback.xml: **c:\documentazione\POO1 - materiale software aggiuntivo\java**