# Programmazione 13/06/2017

**Note sullo svolgimento della prova**

- Non è consentito l'uso di vecchi testi d'esame, libri, o appunti.
- Utilizzare NetBeans o Notepad++ come editor, e finestre DOS per compilare ed eseguire i programmi.
- La cartella C:\prg contiene il JDK da utilizzare e la documentazione delle Java API. Eventuali file ausiliari preparati dal docente devono essere scaricati da elearn.ing.unipi.it
- I file sorgenti relativi all'esercizio 1 devono essere posti nella cartella C:\prg\myapps\esercizio1 mentre quelli relativi all'esercizio 2 devono essere posti nella cartella C:\prg\myapps\esercizio2.
- Al termine della prova, dopo la sottomissione degli elaborati su elearn.ing.unipi.it, il docente mostrerà una possibile soluzione. Dopo aver visto la soluzione, gli studenti avranno la possibilità di riprendere il proprio elaborato.

**Esercizio 1**

Facciamo riferimento a un sistema per la prenotazione di posti su voli aerei. Ogni volo è identificato da una sequenza alfanumerica univoca (per esempio AB1234). Ogni volo dispone di 60 posti, organizzati su 10 file e 6 colonne. Le file sono identificate da numeri che vanno da 1 a 10, mentre le colonne sono identificate dalle lettere A, B, …, F. I clienti possono: selezionare un volo, prenotare un posto su un volo precedentemente selezionato, deselezionare un volo. Un volo può essere selezionato da al più tre clienti contemporaneamente. Realizzare una classe *Gestore* che consente ai clienti di eseguire le seguenti operazioni (realizzare anche eventuali classi ausiliarie):

- *Gestore(String[] v)*: crea un gestore che opera sui voli i cui codici sono forniti come argomento. I posti dei voli sono inizialmente tutti liberi.
- *void seleziona(Cliente x, String c)*: il cliente *x* seleziona il volo con codice *c*. Lancia *VoloNonEsistenteException* se il cliente prova a selezionare un volo che non esiste. Il metodo è bloccante nel caso in cui altri tre clienti abbiano selezionato il medesimo volo.
- *void deseleziona(Cliente x)*: il cliente che chiama il metodo non ha più un volo correntemente selezionato. Risveglia eventuali altri clienti bloccati su una selezione.
- *boolean prenota(Cliente x, int r, char s)*: il cliente *x* prenota, se disponibile, il posto sulla fila *r* e colonna *s*. La prenotazione avviene sul volo precedentemente selezionato dal cliente. Restituisce il valore *true* se il posto viene prenotato, *false* altrimenti (era già prenotato). Lancia *VoloNonSelezionatoException* se nessun volo è stato precedentemente selezionato e *IllegalArgumentException* se i parametri forniti non sono corretti.

**Esercizio 2**

Realizzare un'applicazione *Depositi Bancari* per consultare un archivio dei depositi dei clienti, in accordo ai casi d'uso di Fig.1 e Fig.2, ai requisiti di Tab.1, e seguendo i medesimi criteri di qualità del progetto del corso. Un cliente è identificato dall'indirizzo di email, un deposito è rappresentato come un numero intero non negativo.
È possibile consultare esclusivamente le Java API e le slide del corso fornite dal docente in formato pdf.

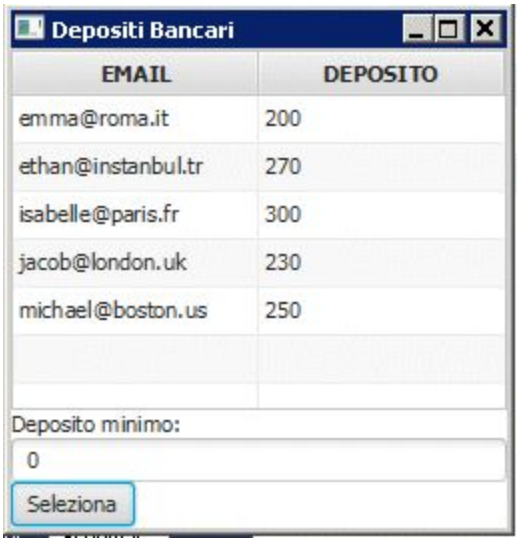


*Fig.1 - Primo avvio dell'applicazione:*

1. L'Utente avvia l'applicazione
2. FOR EACH utente archiviato
   2.1 Il Sistema visualizza identificativo utente (email) e deposito

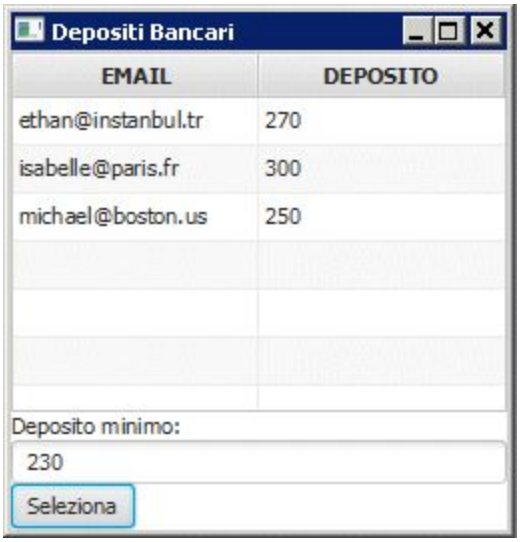


Fig.2 - *Selezione dei clienti con un deposito minimo:*

1. L'Utente inserisce il Deposito minimo
2. L'Utente preme Seleziona
3. FOR EACH utente archiviato con deposito maggiore di quello minimo
   3.1 Il Sistema visualizza identificativo utente (email) e deposito

*Tab.1 - Principali responsabilità e requisiti delle classi da realizzare*

| **Classe** | **Principali responsabilità e requisiti** |
|---|---|
| ConsultazioneDepositiClienti | Costruisce e inizializza il front end dell'applicazione e configura le azioni per ogni evento |
| TabellaVisualeDepositiClienti | Costruisce e inizializza la tabella dei depositi, la aggiorna a partire da un oggetto List<Cliente> |
| Cliente | Classe bean |
| DataBaseDepositiClienti | Costruisce e inizializza la connessione al database e gli statement necessari, riutilizzandoli ad ogni interrogazione e restituendo un oggetto List<Cliente> |

È conveniente sviluppare l'applicazione adoperando l'ambiente netbeans, ma la stessa applicazione deve essere eseguibile adoperando il seguente file make.bat collocato nella medesima cartella dei file sorgenti:

```
c:\prg\jdk8\bin\javac -classpath ".;c:\prg\libs\*" *.java
pause
c:\prg\jdk8\bin\java -classpath ".;c:\prg\libs\*" ConsultazioneDepositiClienti
Pause
```

Fig.3 - File make.bat per il collaudo, già fornito, da non modificare.