

## 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](http://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](http://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

Un treno è composto da un certo numero di vagoni, numerati a partire da 0. Ogni vagone contiene 60 posti che sono inizialmente tutti liberi. I clienti possono visualizzare lo stato dei posti del treno, selezionare un posto e completare l'acquisto di un posto precedentemente selezionato mediante un pagamento.

Realizzare una classe *Treno* dotata almeno dei seguenti metodi e costruttori:

- *Treno(int n)*: crea un treno con  $n$  vagoni.
- *void visualizza(int n)*: stampa a video lo stato dei posti del vagone  $n$ . La stampa deve avere il formato illustrato dal seguente esempio:

```
LLLLLL
LSLLLL
PPSLLP
...
SSPLLP
```

Il carattere **L** indica i posti liberi, **S** quelli selezionati, **P** quelli definitivamente prenotati.

- *boolean seleziona(int n, int p)*: seleziona il posto  $p$  del vagone  $n$ . Il metodo restituisce *true* se il posto è libero, *false* altrimenti.
- *boolean paga(int n, int p)*: completa l'acquisto relativo al posto  $p$  del vagone  $n$ , precedentemente selezionato. Il metodo restituisce *true* se la prenotazione viene completata con successo, *false* altrimenti.

Un posto selezionato per cui non viene completata la procedura di acquisto mediante il pagamento ritorna automaticamente libero dopo 5 secondi (calcolati a partire da quando il posto è stato selezionato).

Realizzare anche una semplice classe *Cliente* che

- visualizza lo stato di un vagone;
- seleziona un posto;
- effettua il pagamento per il posto precedentemente selezionato.

## 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. È possibile consultare esclusivamente le Java API e le slide del corso fornite dal docente in formato pdf.

| EMAIL             | DEPOSITO |
|-------------------|----------|
| emma@roma.it      | 200      |
| ethan@istanbul.tr | 270      |
| isabelle@paris.fr | 300      |
| jacob@london.uk   | 230      |
| michael@boston.us | 250      |

Deposito minimo:

0

Seleziona

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

| EMAIL             | DEPOSITO |
|-------------------|----------|
| ethan@istanbul.tr | 270      |
| isabelle@paris.fr | 300      |
| michael@boston.us | 250      |

Deposito minimo:

230

Seleziona

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> |