Programmazione 06/02/2017

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

Una *Lotteria* è dotata di *n* biglietti, associati ai numeri interi che vanno da *0* a *n* (escluso). Dei giocatori possono prendere uno o più biglietti. Quando tutti i biglietti sono stati presi, viene estratto un numero casuale tra *0* e *n* che determina il vincitore.

Scrivere una classe *Lotteria* dotata almeno dei seguenti costruttori e metodi:

- *Lotteria(int n)*: crea una *Lotteria* in cui i numeri possibili vanno da *0* a *n* escluso.
- *Biglietto prendiNumero(int p)*: consente di prendere il numero *p*. Restituisce un oggetto *Biglietto* che rappresenta il numero preso. Lancia *NumeroPresoException* se il numero è già stato preso da qualcun altro o *IllegalArgumentException* se *p* è al di fuori dell'intervallo dei valori validi.
- *boolean attendiEstrazione(Biglietto[] bb)*: chi chiama questo metodo passa l'array di biglietti in suo possesso. Il metodo è bloccante fino a quando tutti i biglietti non sono stati presi e tutti si sono messi in attesa. L'ultimo giocatore che che si mette in attesa causa l'estrazione del biglietto vincente. Il metodo restituisce *true* al vincitore, *false* a tutti gli altri.

Scrivere una classe *Biglietto* ausiliaria utile a realizzare quanto sopra. Scrivere una classe *Giocatore* che si comporta nel seguente modo:

- quando un giocatore viene attivato prende un biglietto;
- si mette in attesa dell'estrazione;
- stampa un messaggio che riporta l'esito dell'estrazione.

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.

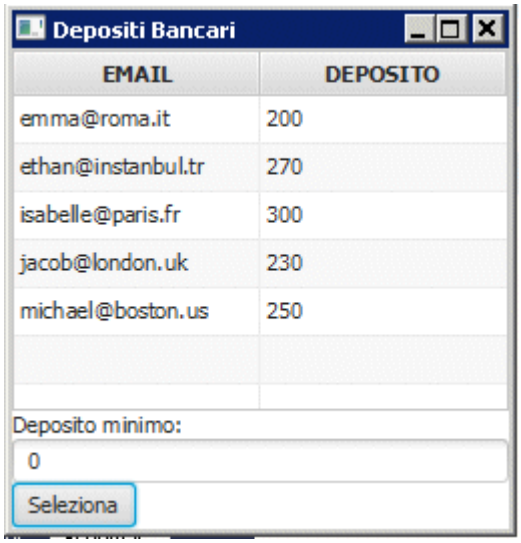

Fig.1 - Primo avvio dell'applicazione:

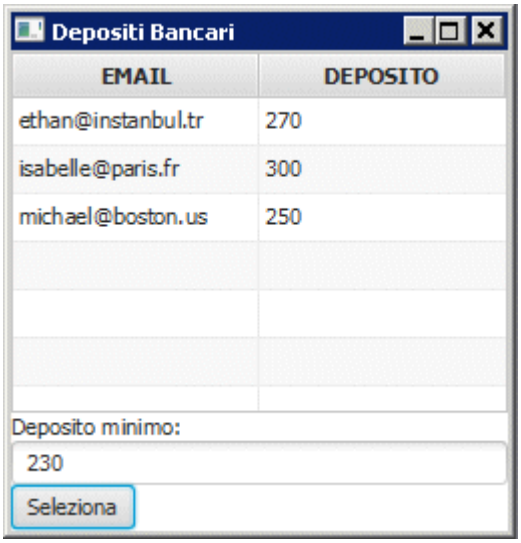

Fig.2 - *Selezione dei clienti con un deposito minimo:*

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

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>