


Esame di Programmazione Internet

CdL in Informatica per il Management

Tempo a Disposizione: 1h30 secondo parziale (solo esercizio 4) – 3h esame completo

Leggete BENE tutte le istruzioni PRIMA di contattare il docente. NON scrivere su questo foglio e RESTITUIRLO sempre a fine compito (anche se vi ritirate)

- Prendete posto e **loggatevi su EOL** all'indirizzo <http://eol.unibo.it>. Cliccate sull'appello con la data odierna. Se non riuscite a connettervi, contattate il docente
- Cliccate sul tasto **"Windows"**  della tastiera, appariranno diverse opzioni:
 - **Nautilus**: esplora file
 - **Firefox/Chromium**: browser
 - **jEdit, editor java**
 - **gedit**: editor di testo

Aprire **jEdit** (non gedit) e **attendete istruzioni** dal docente. Se non si apre, contattate il docente. **NOTA: Spesso capita che jEdit sembra "scompare" dallo schermo. In questo caso premere ALT+TAB: per passare da un'applicazione all'altra e tornare a jEdit**

- Quando tutti sono pronti, il docente **carica il testo dell'esame su EOL** e dà il via all'esame. Il tempo indicato è **tassativo**: EOL registra l'orario di consegna, eventuali **ritardi comporteranno penalizzazioni o annullamento del compito**
- L'esame consiste nella scrittura di codice ***.java compilabile**. Si consiglia di **salvare e compilare spesso** (anche se l'esercizio non è terminato). Si può fare copia-incolla del testo come commento, in modo da non dover aprire ogni volta il pdf con il testo dell'esercizio
- **NOTA: Nel caso si dovesse bloccare il sistema di sviluppo (es. loop infinito) aprite il terminale (CTRL+ALT+t) e digitate "pkill java" o "pkill jedit". Si dovrebbe chiudere jEdit: a questo punto scrivete "jedit &" e ripartite.**
- Verranno valutate, oltre alle **competenze** di programmazione, la capacità del candidato di **comprendere il testo** del compito (ad es., programmi che girano ma **non** fanno quanto richiesto sono considerati insufficienti) e di scrivere **codice pulito, ordinato e ottimizzato**.
- Nel caso di **esame parziale**, se svolgete un esercizio che non dovete svolgere tale esercizio verrà **annullato**
- Una volta terminato, fare **l'upload** solamente dei **file *.java su EOL**. **Non consegnare file *.class e nominare i file in modo chiaro (i numeri degli esercizi devono corrispondere, evitare spazi nel nome)**. Si consiglia di eseguire **l'upload almeno 5 minuti prima del termine del compito**. Eventuali **.class senza corrispondenti .java non verranno presi in considerazione!**

NON è consentito l'uso di testi, appunti cartacei e altre risorse on-line.

STUDENTI DSA:

- Gli studenti DSA che svolgono il **secondo esame parziale** hanno a disposizione tutte le **3 ore**
- Gli studenti DSA che svolgono l'**esame completo** hanno a disposizione **3 ore** ma **non** devono svolgere l'**esercizio 1**.

TESTO DEL COMPITO

Esercizio 1

Leggere numeri interi da tastiera e inserirli in un vettore X fino a che viene inserito un numero minore di 0 (da non inserire). Quindi stampare X[0], X[X[0]], X[X[X[0]]] ecc... fino a che si trova un elemento X[i] tale che X[i] = i, oppure X[i] è maggiore o uguale alla lunghezza di X. Ad esempio se vengono letti:

3
20
8
2
-10

Vanno stampati, uno sotto l'altro:

3
2
8

poiché X[0] = 3, X[3] = 2, X[2] = 8 e poi ci si ferma perchè 8 è maggiore o uguale alla dimensione di X (4 in questo caso, perchè -10 non va inserito).

Esercizio 2

Generare casualmente una matrice NxM di Boolean, dove N ed M sono interi positivi inseriti da utente (non si richiedono controlli). Stampare quindi il risultato della congiunzione logica (AND) della disgiunzione logica (OR) di tutte le righe della matrice. Ad esempio se la matrice è:

false true false
false false false
true true false

Va stampato false, poiché: (false OR true OR false) AND (false OR false OR false) AND (true OR true OR false) = true AND false AND true = false

Non è possibile utilizzare metodi ausiliari per calcolare il risultato.

Esercizio 3

Definire un metodo **ricorsivo** che prende in input 2 matrici di double A e B e ritorna la matrice somma A+B. Se le dimensioni di A e B sono diverse, stampare un messaggio di errore. E' possibile fare overload del metodo, ma non utilizzare metodi ausiliari. Ad esempio:

A			B			A + B	
5	-2.4		2.9	-10		7.9	-12.4
3.5	0.6	+	4	7.2	=	7.5	7.8

Suggerimento: si può utilizzare un parametro aggiuntivo per la matrice somma

Esercizio 4

Si realizzi un'applicazione Java per gestire gli ordini in un negozio online di articoli da mare.

Ogni articolo è caratterizzato da un codice identificativo (stringa) e un prezzo (double).

Ogni ordine è caratterizzato da: codice identificativo (stringa), elenco degli articoli ordinati e un prezzo totale (double). E' possibile ordinare un numero di articoli illimitato.

Semplificazione: assumere che gli identificativi degli articoli e degli ordini siano univoci; non è richiesto fare controlli nel codice.

Gli ordini sono gestiti con una politica stravagante: di base FIFO (First-In, First-Out) per cui sono evasi in ordine di arrivo; se un ordine però supera i 500 euro ha la priorità e sarà il prossimo ad essere evaso (se nel frattempo arriva un altro ordine superiore a 500 euro, quest'ultimo sarà quindi evaso per primo).

Esistono tre tipi di articoli:

- *Standard*, per il quale si memorizza il prezzo base
- *Scontato*, per il quale è necessario memorizzare anche lo sconto applicato (valore intero, in percentuale); lo sconto è deciso alla creazione dell'articolo ma è modificabile in seguito
- *Tris*, composto da 3 articoli Standard; in questo caso l'articolo meno costoso è in regalo e il prezzo è determinato dalla somma dei prezzi degli altri due.
Un articolo "Tris" non può includere articoli già scontati o a loro volta di tipo "Tris".

Definire le classi utili per modellare questa situazione, con gli opportuni costruttori e metodi.

Implementare la classe `Ordine` che espone i metodi per:

- aggiungere un articolo all'ordine ; NON è richiesto creare interattivamente gli ordini
- restituire il prezzo complessivo dell'ordine, come somma dei prezzi degli articoli, eventualmente scontati

Implementare la classe `Negoziomare` che memorizza la lista degli ordini ed espone i metodi per:

- aggiungere un ordine, in modo che potrà essere evaso secondo la politica descritta sopra (FIFO + priorità se il prezzo è maggiore di 500)
- esportare su file le informazioni relative ai prossimi N ordini da evadere; se sono stati effettuati meno di N ordini esportare tutti i dati; il formato di salvataggio dei dati è arbitrario - restituisce un booleano per indicare se l'operazione è andata a buon fine o meno

Si realizzi una classe demo con dati arbitrari per testare il corretto funzionamento del codice. E' ammesso aggiungere metodi non richiesti ma utili per mostrare i risultati.