

Corso di Laurea in Informatica
Prova Scritta di Metodologie di Programmazione - Primo Canale

Sapienza Università di Roma

17 Maggio 2023

Durante l'esame non è consentito l'utilizzo di alcunché. Non è consentito inoltre l'utilizzo della matita o di penne il cui colore sia diverso dal nero o dal blu. Il ritiro dalla prova equivale al mancato superamento dell'esame.

Nome e Cognome:

Matricola:

Domanda	1	2	3	4	5	6	Totale
Punteggio Totale	10	4	3	5	12	8	32
Punteggio Ottenuto							

1. 10 punti Per ogni domanda, indicare con una X la risposta desiderata. Si ricorda che ogni domanda ha al più una risposta corretta. L'assegnazione dei punti alle risposte è la seguente. Verranno attribuiti 2 punti per ogni risposta esatta; -0.4 punti per ogni risposta errata; 0 punti per ogni risposta omessa. Al fine del superamento della soglia è necessario totalizzare un punteggio di almeno 5 punti.
 - (a) Un ciclo for è un costrutto di iterazione ☐ Indeterminato ☐ Determinato ☐ Ciclico
 - (b) Una costante in Java è indicata dal modificatore ☐ static ☐ abstract ☐ final
 - (c) I metodi di un'interfaccia sono ☐ protected ☐ public ☐ private
 - (d) Il contenuto di due stringhe è confrontabile tramite il metodo ☐ compareTo ☐ equals ☐ corresponds
 - (e) Un metodo static può essere accessibile tramite ☐ la sua classe ☐ il suo oggetto ☐ l'uso di alcun modificatore di visibilità
2. 4 punti Qual è la differenza tra overloading e overriding? Fornire un esempio minimale, scritto in Java, che descriva quanto richiesto.

Overloading e Overriding sono concetti legati al polimorfismo il quale, nella programmazione ad oggetti, si riferisce alla capacità di una classe A di assumere i valori di un qualunque tipo descritto dalla classe B, con B sottoclasse di A.

*Si parla di **overloading** quando sono presenti più metodi all'interno di una classe o di una sua superclasse con lo stesso nome, tipo di ritorno ma con tipo e/o numero di parametri diverso. Si parla invece di **overriding** quando un metodo della classe derivata *B* sovrascrive il metodo della classe *A* mantenendo quindi la stessa firma ma cambiando il corpo della funzione.*

Un esempio minimale in Java di quanto spiegato è fornito nella cartella *Es₁*.

3. 3 punti Per ogni costrutto iterativo, indicare il numero di volte per il quale viene eseguito il suo corpo. Se non diversamente indicato, si assume che la variabile contatore non venga modificata all'interno del corpo di ciascun costrutto iterativo.

- (a) `for(int i = 12; i <= 7; i++){...}`
- (b) `for(int i = 0; i < 10; i++){...}`
- (c) `for(int i = -10; i >= 0; i++){...}`
- (d) `for(int i = -2; i >= 0; i++){...}`
- (e) `for(int i = -8; i <= 3; i = i + 4){...}`
- (f) `for (int k = 0; k < 20; k+=2) {
 if (k + 3 == 1) System.out.println(k + " ");
}`

4. 5 punti Spiegare le principali differenze tra `ArrayList` e `Set` in Java.

`ArrayList` implementa l'interfaccia `List` appartenente a sua volta all'interfaccia `Collection` di Java. Un `ArrayList` è una collezione di oggetti memorizzati sequenzialmente secondo una politica FIFO. Esso può contenere valori il cui tipo può essere primitivo o meno, ammettendo duplicati. La dimensione dell'`ArrayList` è dinamica e l'accesso ai dati è randomico (posizionale).

I `Set`, appartenenti anch'essi all'interfaccia `Collection`, rappresentano un insieme di valori senza duplicati. Tale interfaccia `Set` è a sua volta implementata da `HashSet` e `TreeSet`: la prima utilizza una tabella hash, offrendo quindi un tempo di esecuzione costante per le operazioni di *add*, *remove*, *size*, *contains*. `TreeSet`, al contrario, utilizza un albero interno per le operazioni di memorizzazione offrendo quindi la possibilità di avere un ordinamento. `HashSet` permette la presenza di un valore nullo, mentre `TreeSet` no.

5. 12 punti Realizzare la classe *Customer* al fine di gestire le informazioni relative ad un cliente di una campagna di promozione commerciale. Tale campagna si articola in questo modo: dopo aver effettuato una serie di acquisti per almeno \$100, il cliente riceverà uno sconto di \$10 sull'acquisto successivo. Progettare quindi i seguenti metodi:

```
public void makePurchase(double amount) // registra un acquisto
public boolean discountReached() // true se e solo se l'utente ha raggiunto lo sconto
```

Al fine di verificare la correttezza della propria implementazione, si realizzi un programma di collaudo (test). Esso dovrà rappresentare la seguente situazione: un utente dovrà ottenere uno sconto, il quale lo utilizzerà per fare degli acquisti. L'importo totale derivante da tali acquisti dovrà essere maggiore di \$90 ma minore di \$100 in modo tale da non poter ottenere uno sconto. Successivamente, l'utente dovrà effettuare un altro acquisto, il quale lo porterà a godere dello sconto sull'acquisto successivo. cliente

6. 8 punti Un supermercato ha accesso ad un database contenente tutte le informazioni sui propri prodotti. Tali informazioni sono rappresentate in formato tabellare, in cui ciascuna riga contiene i seguenti campi separati da uno spazio:

```
codiceProdotto1 NomeProdotto1 Quantità1 PrezzoPerUnità1
codiceProdotto2 NomeProdotto2 Quantità2 PrezzoPerUnità2
...
```

Scrivere un programma che legga un file di testo con questa struttura, segnalando un opportuno errore in caso di file inesistente. Successivamente, visualizzare:

- Le informazioni relative al prodotto più costoso;
- Il prezzo medio per ciascun prodotto, calcolato come il rapporto tra prezzo per unità e la quantità disponibile. Gestire gli eventuali casi speciali
- I nomi dei prodotti esauriti, ossia con una quantità pari a zero