ESERCIZI di preparazione all'esame

Esercizio 1.

Si progetti una classe Articolo con tre variabili di istanza: descrizione, prezzo unitario e quantità.
Si progetti una classe Magazzino con due variabili d'istanza: un nome di tipo String e una lista di articoli di tipo array di Articolo.
Si scrivano i metodi della classe Magazzino per calcolare:

- dato un parametro intero n, il numero di articoli la cui quantità è minore o uguale a n (ritorna un intero);
- il valore totale degli articoli del magazzino (ritorna un double);
- se la quantità di tutti gli articoli è diversa da 0 (ritorna un boolean).

Infine si progetti una classe di test che costruisce un magazzino con tre articoli, invoca tutti i metodi, e ne stampa i risultati.

Esercizio 1.

Si progetti una classe InsiemeLimitato che rappresenti un insieme di al più n elementi di tipo intero. La classe InsiemeLimitato utilizza un array di interi di lunghezza n per memorizzare gli elementi dell'insieme ed una variabile numElementi per tenere traccia del numero di elementi contenuti nell'insieme.
Si progetti un costruttore con un parametro n che indica il numero massimo di elementi che l'insieme può contenere.
Si progettino i seguenti metodi della classe InsiemeLimitato:

- getNumElementi(): restituisce il numero di elementi contenuti nell'insieme;
- getSize(): restituisce il numero massimo di elementi che l'insieme può contenere;
- boolean contenuto(int e): restituisce true se l'elemento e è contenuto nell'insieme, false altrimenti;
- boolean vuoto(): restituisce true se l'insieme è vuoto, false altrimenti;
- boolean pieno(): restituisce true se l'insieme è pieno, false altrimenti;

Si progetti una classe di test che contenga i seguenti metodi statici:

- boolean confronta(InsiemeLimitato i, InsiemeLimitato j): restituisce true se gli insiemi i e j hanno gli stessi elementi, false altrimenti;
- InsiemeLimitato differenza(InsiemeLimitato i, InsiemeLimitato j): restituisce l'insieme differenza formato da tutti gli elementi che appartengono ad i e non appartengono a j.

Infine si scriva il metodo main nel quale si creano due insiemi e si chiamano tutti i metodi.

Esercizio 1.

Si progetti una classe RegistroVendite per registrare le vendite di un negozio (vettore di importi double).
Si progetti un costruttore in modo tale che un RegistroVendite possa registrare al massimo 10 vendite e che l'utente debba specificate quante vendite vuole inserire (max 10) e fornisca gli importi.
Si progettino inoltre i seguenti metodi per calcolare alcune statistiche sui dati contenuti nel vettore;

- double totaleVendite(): restituisce l'importo totale delle vendite
- int venditeAlDiSopra(double valoreDiRiferimento)restituisce il numero di vendite che hanno un valore superiore a valoreDiRiferimento, richiesto in input all'utente
- double mediaVendite(): restituisce il valore medio di tutte le vendite
- double importoPiùAlto(): restituisce il valore più alto fra tutte le vendite
- void inserisci(double nuovoEl): inserisce una nuova vendita, se è possibile, altrimenti non fa niente.

Infine si progetti una classe di test che costruisce un RegistroVendite, invoca tutti i metodi, ed eventualmente ne stampa i risultati.

Esercizio 1.
Un multinsieme è una generalizzazione del concetto di insieme che permette elementi ripetuti. Ad esempio [[1, 2, 2, 2, 5, 6, 6]] è un multinsieme dove il numero 2 è ripetuto tre volte.
Si progetti una classe Multinsieme con una variabile d'istanza di tipo array di interi che conterrà gli elementi del multinsieme.
Si scrivano un costruttore ed i metodi della classe Multinsieme per:

- inserire un nuovo elemento (non è necessario mantenere l'ordinamento tra gli elementi);
- calcolare il numero di elementi contenuti nel multinsieme, tenendo conto delle ripetizioni (esempio: il multinsieme [[1, 2, 2, 2, 5, 6, 6]] contiene 7 elementi);
- eliminare una occorrenza di un elemento: il metodo deve ritornare true se è stato possibile eliminare l'elemento, false altrimenti (es. eliminando 2 da [[1, 2, 2, 2, 5, 6, 6]] otteniamo [[1, 2, 2, 5, 6, 6]] ed il metodo ritorna true);
- dato un numero “n”, calcolare il numero di occorrenze di “n” (esempio: le occorrenze di 2 in [[1, 2, 2, 2, 5, 6, 6]] sono tre).

Infine si progetti una classe di test che costruisce un multinsieme, invoca tutti i metodi, e ne stampa i risultati.

Esercizio 1.
Si realizzi una classe Compiti per la rappresentazione di compiti da eseguire. Un compito è caratterizzato dalla descrizione (stringa), dalla data di scadenza (tre interi con giorno, mese, anno) e dalla priorità (un intero: più è grande, maggiore è la priorità). La classe deve fornire i seguenti metodi:

- Compito(String descr, int g, int m, int a, int p): costruttore che presi in ingresso una stringa descr e quattro interi g, m, a e p, li imposta rispettivamente come descrizione, giorno, mese, anno di scadenza e priorità.
- i metodi get corrispondenti a tutte le variabili d'istanza;
- primaDi(Compito c) che restituisce un booleano risultato della comparazione tra la scadenza e la priorità dell'istanza e del compito c: true se la scadenza dell'istanza è precedente a quella di a oppure se la scadenza è uguale e la priorità è maggiore o uguale, false altrimenti.

Si realizzi poi una classe ListaCompiti per la rappresentazione di liste di compiti. Ogni lista di compiti contiene al suo interno un array contenente i compiti che la compongono. Inizialmente è vuota. Ogni lista di compiti conterrà inoltre due interi rappresentanti rispettivamente la minima e la massima priorità che possono essere utilizzate. La classe deve fornire i seguenti metodi:

- ListaCompiti(int minP, int maxP): costruttore, prende in ingresso la priorità minima minP e massima maxP ed inizializza l'oggetto.
- aggiungiCompito(Compito c): prende in ingresso un oggetto di tipo Compito e lo aggiunge alla lista di compiti. Se la priorità del compito è minore della minima o maggiore della massima per la lista NON deve essere inserito ed il metodo deve restituire false. In caso di inserimento effettuato il metodo restituisce true.
- carico(int p): restituisce il numero di compiti presenti nella lista la cui priorità è maggiore o uguale di p.
- eseguiCompito(): restituisce il compito presente nella lista che ha la minore data di scadenzae la priorità più alta a parità di scadenza; tale compito viene eliminato dalla lista.