**Esercizio 1**

Scrivere una classe java Portata che rappresenta una portata ordinabile in un ristorante. La portata e' caratterizzata da una descrizione (stringa), dal prezzo (double) e da un booleano che posto uguale a TRUE rappresenta il fatto che la portata e' una bevanda. La classe ha un costruttore che presi in ingresso una stringa, un double e un boolean li imposta come campi d'esemplare del nuovo oggetto, e in piu' fornisce i seguenti metodi:

* isBevanda() che restituisce un boolean che rappresenta il fatto che la portata su cui si invoca il metodo e' una bevanda
* getPrezzo() che restituisce il prezzo della portata
* setPrezzo(d) che preso in ingresso un double d lo imposta come nuovo prezzo della portata

Scrivere poi una classe java Ordinazione. Un'ordinazione e' caratterizzata da una strnga rappresentante il nome del cliente che ha effetuato l'ordinazione e da un array (inizialmente vuoto) di portate ordinate, ognuna delle quali ha associata la quantita' ordinata. La classe ha un costruttore che prende in ingresso il nome del cliente e inizializza la nuova ordinazione, e inoltre fornisce i seguenti metodi:

* aggiungiPortata(p,q) che presi in ingresso una portata p e un intero q aggiunge la portata p all'array delle portate ordinate, con quantita' corrispondente q
* conto() che restituisce il conto totale dell'ordinazione
* contoBevande() che restituisce il costo totale delle bevande contenute nell'ordinazione
* numeroTotalePortate() che restituisce la somma delle quantita' ordinate di tutte le portate
* portateOrdinate() che restituisce il numero delle diverse portate ordinate

Si scrivano le classi in modo da potercompilare ed eseguire la seguente classe di prova senza doverla modificare.

public class **ProvaEserc1** {  
 public static void **main**(String[] args) {  
 Portata p1 = new Portata("antipasto",8.5,false);  
 Portata p2 = new Portata("matriciana",12.5,false);  
 Portata p3 = new Portata("bistecca",25,false);  
 Portata p4 = new Portata("insalata",3,false);  
 Portata p5 = new Portata("negramaro",15.5,true);  
 Portata p6 = new Portata("acqua",2.5,true);  
 Ordinazione o = new Ordinazione("luca");  
 o.aggiungiPortata(p1, 3);  
 o.aggiungiPortata(p2, 4);  
 o.aggiungiPortata(p3, 3);  
 o.aggiungiPortata(p4, 1);  
 o.aggiungiPortata(p5, 3);  
 o.aggiungiPortata(p6, 2);  
 System.out.println("Conto: " + o.conto());  
 System.out.println("Conto bevande: " + o.contoBevande());  
 System.out.println("Portate: " + o.portateOrdinate());  
 System.out.println("Totale portate: " + o.numeroTotalePortate());  
 p1.setPrezzo(10);  
 p2.setPrezzo(10);  
 p3.setPrezzo(10);  
 p4.setPrezzo(10);  
 p5.setPrezzo(10);  
 p6.setPrezzo(10);  
 System.out.println("Conto: " + o.conto());  
 System.out.println("Conto bevande: " + o.contoBevande());  
 System.out.println("Portate: " + o.portateOrdinate());  
 System.out.println("Totale portate: " + o.numeroTotalePortate());   
 }   
}

**Esercizio 2**

Scrivere una classe Java Eserc2.java contenente un metodo statico somma che, ricevuti come parametri due array bidimensionali di interi m1 ed m2 rappresentanti due matrici (non necessariamente quadrate) *aventi le stesse dimensioni*, restituisca un nuovo array bidimensionale di interi m3, che rappresenti la somma di m1 ed m2. Ad esempio, se m1 ed m2 sono le matrici seguenti:

m1

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 1 | 0 | 5 |
| 8 | 8 | 0 | 0 |
| 1 | 7 | 8 | 3 |

m2

|  |  |  |  |
| --- | --- | --- | --- |
| -2 | 3 | 2 | -10 |
| 3 | 1 | 2 | 0 |
| 2 | 2 | 8 | 4 |

la matrice risultante m3 deve essere:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 4 | 2 | -5 |
| 11 | 9 | 2 | 0 |
| 3 | 9 | 16 | 7 |

Scrivere le classi in modo da poter compilare ed eseguire il seguente programma di prova senza doverlo modificare.

public class **ProvaEserc2** {  
  
 public static void main(String[] args) {  
 int [][] mat1 = {{8, 2, -1, 0}, {7, 8, 0, 12}, {0, 0, 8, 8}};  
 int [][] mat2 = {{-1, 3, -10, 2}, {3, 4, 13, 1}, {-2, 0, -4, 2}};  
  int [][] mat3 = Eserc2.somma(mat1, mat2);  
 System.out.println("Risultato corretto: ");  
 System.out.println("7 5 -11 2");  
 System.out.println("10 12 13 13");  
 System.out.println("-2 0 4 10");  
 System.out.println();  
   
 // Il risultato calcolato deve essere uguale a quello corretto  
 System.out.println ("Risultato trovato: ");  
 for (int i=0; i<mat3.length; i++) {  
 for (int j=0; j<mat3[0].length; j++) {  
 System.out.print(mat3[i][j]+ " ");  
 }  
 System.out.println();  
 }  
 System.out.println();  
 }  
  
}

**Esercizio 3**

Si consideri la classe Prodotto.java, riportata di seguito per comodità, che rappresenta informazioni su prodotti in vendita presso un esercizio commerciale. In particolare, per ogni prodotto interessa conoscere il codice (una stringa univoca che lo individua), una sua descrizione (anch'essa una stringa), nonche' la quantità di confezioni disponibile in magazzino.

public class **Prodotto** {  
    String codice;  
    String desc;  
    int quant;  
    Prodotto prox;  
     
    public Prodotto(String c, String d, int q) {  
        codice = c;  
        desc = d;  
        quant = q;  
        prox = null;  
    }  
}

Scrivere una classe Magazzino (quindi in un file Magazzino.java) che gestisce una lista dei prodotti disponibili nel magazzino dell' esercizio commerciale. La classe ha un campo inizio di classe Prodotto, contenente il riferimento all'inizio della lista, e un costruttore che inizializza una lista vuota di prodotti. La classe ha inoltre i seguenti metodi pubblici:

- boolean esiste(String c): riceve in ingresso come parametro una stringa che rappresenta il codice di un prodotto e restituisce true qualora il prodotto sia presente in magazzino, false altrimenti.  
- void aggiornaMagazzino(String c, String d, int q): riceve in ingresso come parametri due stringhe che rappresentano il codice e la descrizione di un prodotto e un intero che rappresenta un incremento (non negativo) nella quantità del prodotto. Se il prodotto è già presente in magazzino (dunque esiste un prodotto con lo stesso codice) allora il metodo deve semplicemente incrementarne la quantità del valore q, mentre il valore del parametro d va ignorato. Altrimenti esso deve creare un nuovo oggetto di classe Prodotto e aggiungerlo alla lista, inizializzandone i campi ai valori passati come parametri.  
- int quantita(String c): riceve in ingresso come parametro una stringa c che rappresenta il codice di un prodotto e resituisce la quantità del prodotto. Qualora il prodotto non sia presente, il metodo deve restituire 0.  
- void rimuoviProdotto(String c): riceve in ingresso come parametro una stringa c che rappresenta il codice di un prodotto da rimuovere dall'inventario di magazzino (anche se presente in quantità maggiore di 0). Il metodo rimuove il prodotto avente il codice ricevuto come parametro dalla lista. Qualora la lista non contenga un prodotto avente codice  c il metodo non apporta alcuna modifica.

Scrivere le classi in modo da poter compilare ed eseguire il seguente programma di prova senza doverlo modificare.

public class ProvaEserc3 {  
  
 public static void main(String[] args) {  
 Magazzino myMag = new Magazzino();  
   
 System.out.println("Aggiungo tre prodotti alla lista, tutti per la prima volta...");  
 myMag.aggiornaMagazzino("ga1", "Fringles", 100);  
 myMag.aggiornaMagazzino("det1", "Morbidissimo", 50);  
 myMag.aggiornaMagazzino("ga2", "Cric crac", 120);  
 System.out.println("... Devono comparire tutti nella lista");  
 stampa(myMag);  
 System.out.println("-------------------------");  
  
 System.out.println("Aggiornamento per i prodotti aventi codici ga2 e det1");  
 System.out.println("Entrambi sono presenti nella lista ... devo aggiornare soltanto le quantità");  
 myMag.aggiornaMagazzino("ga2", "Cric crac", 20);  
 myMag.aggiornaMagazzino("det1", "Morbidissimo", 15);  
 System.out.println("... I prodotti devono comparire con info su quantità...");  
 stampa(myMag);  
 System.out.println("-------------------------");  
   
 System.out.println("Quantità del prodotto con codice det1: " + myMag.quantita("det1"));  
 System.out.println("-------------------------");  
 System.out.println();  
 System.out.println("Quantità del prodotto con codice ga2: " + myMag.quantita("ga2"));  
 System.out.println("-------------------------");  
   
 System.out.println("Esiste un prodotto con codice gx25? Deve restituire false. " + "Restituisce " + myMag.esiste("gx25"));  
 System.out.println("-------------------------");  
  
 System.out.println("Rimuovo le patatine Fringles dalla lista...");  
 myMag.rimuoviProdotto("ga1");  
 System.out.println("...Quindi non devono comparire nella stampa");  
 stampa(myMag);  
 System.out.println("-------------------------");  
   
 System.out.println("Aggiungo i bottoncini di cioccolato Stupids con codice ga4...");  
 myMag.aggiornaMagazzino("ga4", "Bottoncini di cioccolato Stupids", 50);  
 stampa(myMag);  
 }  
   
 private static void stampa(Magazzino myMag) {  
 Prodotto p = myMag.inizio;  
 while (p != null) {  
 System.out.println("Codice prodotto: " + p.codice + "; " + "Descrizione prodotto: " + p.desc + "; " + "Quantità: " + p.quant);  
 System.out.println();  
 p = p.prox;  
 }  
   
 }  
  
}

**Esercizio 4**

public class **NodoBin** {  
 int valore;  
 NodoBin sin;  
 NodoBin des;  
   
 public **NodoBin**(int val, NodoBin s, NodoBin d) {  
 valore = val;  
 sin = s;  
 des = d;  
 }  
}

Data la rappresentazione collegata di alberi binari presentata sopra, i cui nodi contengono un'informazione di tipo intero, scrivere nella classe Eserc4.java un metodo statico public static int minore(NodoBin radice, int x) che preso in ingresso il riferimento alla radice di un albero binario e un numero intero x, restituisca il numero di nodi dell'albero il cui valore e' **minore o uguale** di x.  
Scrivere la classe in modo da poter compilare ed eseguire il seguente programma di prova senza doverlo modificare.

public class ProvaEserc4 {  
  
 public static void main(String[] args) {  
 NodoBin n7 = new NodoBin(200,null, null);  
 NodoBin n6 = new NodoBin(80,n7,null);  
 NodoBin n5 = new NodoBin(120,null,null);  
 NodoBin n4 = new NodoBin(100,null,null);  
 NodoBin n3 = new NodoBin(15,n6,null);  
 NodoBin n2 = new NodoBin(2000,n4,n5);  
 NodoBin n1 = new NodoBin(10,n2,n3);  
   
 System.out.println("Vi sono 4 nodi con valore minore o uguale di 100");  
 System.out.println("Il programma restituisce il valore " + Eserc4.minore(n1, 100));  
 System.out.println();  
 System.out.println("Vi sono 0 nodi con valore minore o uguale di 1");  
 System.out.println("Il programma restituisce il valore " + Eserc4.minore(n1, 1));  
 System.out.println();  
 System.out.println("Vi sono 7 nodi con valore minore o uguale di 10000");  
 System.out.println("Il programma restituisce il valore " + Eserc4.minore(n1, 10000));  
 System.out.println();  
 }  
  
}

**Esercizio 1**

Scrivere una classe java Prodotto che rappresenta un prodotto ordinabile in un sito internet per vendite on-line. Il prodotto e' caratterizzato da una descrizione (stringa), dal prezzo (double) e da un booleano che posto uguale a TRUE rappresenta il fatto che il prodotto e' in promozione. La classe ha un costruttore che presi in ingresso una stringa, un double e un boolean li imposta come campi d'esemplare del nuovo oggetto, e in piu' fornisce i seguenti metodi:

* isInPromozione() che restituisce un boolean che rappresenta il fatto che il prodotto su cui si invoca il metodo e' in promozione
* getPrezzo() che restituisce il prezzo del prodotto
* setPrezzo(d) che preso in ingresso un double d lo imposta come nuovo prezzo del prodotto

Scrivere poi una classe java Carrello. Un carrello e' caratterizzato da una strnga rappresentante il nome del cliente che sta acquistando e da un array (inizialmente vuoto) di prodotti ordinati, ognuno dei quali ha associata la quantita' ordinata. La classe ha un costruttore che prende in ingresso il nome del cliente e inizializza il nuovo carrello, e inoltre fornisce i seguenti metodi:

* aggiungiProdotto(p,q) che presi in ingresso un prodotto p e un intero q aggiunge il prodotto p all'array dei prodotti ordinati, con quantita' corrispondente q
* totale() che restituisce il totale dell'ordine
* totalePromozione() che restituisce l'importo totale dei prodotti in promozione contenuti nell'ordine
* numeroTotaleProdotti() che restituisce la somma delle quantita' ordinate di tutti i prodotti
* prodottiOrdinati() che restituisce il numero dei diversi prodotti ordinati

Si scrivano le classi in modo da potercompilare ed eseguire la seguente classe di prova senza doverla modificare.

public class **ProvaEserc1** {  
 public static void **main**(String[] args) {  
 Prodotto p1 = new Prodotto("ram",22.5,false);  
 Prodotto p2 = new Prodotto("hd",62.5,false);  
 Prodotto p3 = new Prodotto("alimentatore",74,false);  
 Prodotto p4 = new Prodotto("monitor",300,false);  
 Prodotto p5 = new Prodotto("tastiera",25.5,true);  
 Prodotto p6 = new Prodotto("mouse",12.5,true);  
 Carrello o = new Carrello("sergio");  
 o.aggiungiProdotto(p1, 3);  
 o.aggiungiProdotto(p2, 4);  
 o.aggiungiProdotto(p3, 3);  
 o.aggiungiProdotto(p4, 1);  
 o.aggiungiProdotto(p5, 3);  
 o.aggiungiProdotto(p6, 2);  
 System.out.println("Totale: " + o.totale());  
 System.out.println("Totale promozione: " + o.totalePromozione());  
 System.out.println("Prodotti: " + o.prodottiOrdinati());  
 System.out.println("Prodotti ordinati: " + o.numeroTotaleProdotti());  
 p1.setPrezzo(10);  
 p2.setPrezzo(10);  
 p3.setPrezzo(10);  
 p4.setPrezzo(10);  
 p5.setPrezzo(10);  
 p6.setPrezzo(10);  
 System.out.println("Totale: " + o.totale());  
 System.out.println("Totale promozione: " + o.totalePromozione());  
 System.out.println("Prodotti: " + o.prodottiOrdinati());  
 System.out.println("Prodotti ordinati: " + o.numeroTotaleProdotti());   
 }   
}

**Esercizio 2**

Scrivere una classe Java Eserc2.java contenente un metodo statico maggiori che, ricevuti come parametri due array bidimensionali di interi m1 ed m2 rappresentanti due matrici (non necessariamente quadrate) *aventi le stesse dimensioni*, restituisca un nuovo array bidimensionale di interi m3, tale che l'elemento di m3 in posizione (i, j) vale 1 se m(i, j) > m2(i, j), 0 se m1(i, j) = m2(i, j) e -1 se m1(i, j) < m2(i, j). Ad esempio, se m1 ed m2 sono le matrici seguenti:

m1

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 1 | 0 | 5 |
| 8 | 8 | 0 | 0 |
| 1 | 7 | 8 | 3 |

m2

|  |  |  |  |
| --- | --- | --- | --- |
| -2 | 3 | 2 | -10 |
| 3 | 1 | 2 | 0 |
| 2 | 7 | 8 | 4 |

la matrice risultante m3 deve essere:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | -1 | -1 | 1 |
| 1 | 1 | -1 | 0 |
| -1 | 0 | 0 | -1 |

Scrivere le classi in modo da poter compilare ed eseguire il seguente programma di prova senza doverlo modificare.

public class **ProvaEserc2** {  
  
 public static void main(String[] args) {  
 int [][] mat1 = {{8, 2, -1, 0}, {7, 8, 0, 12}, {0, 0, 8, 8}};  
 int [][] mat2 = {{-1, 3, -10, 2}, {3, 4, 13, 1}, {-2, 0, -4, 2}};  
 int [][] mat3 = Eserc2.maggiori(mat1, mat2);  
 System.out.println("Risultato corretto: ");  
 System.out.println("1 -1 1 -1");  
 System.out.println("1 1 -1 1");  
 System.out.println("1 0 1 1");  
 System.out.println();  
   
 // Il risultato calcolato deve essere uguale a quello corretto  
 System.out.println ("Risultato trovato: ");  
 for (int i=0; i<mat3.length; i++) {  
 for (int j=0; j<mat3[0].length; j++) {  
 System.out.print(mat3[i][j]+ " ");  
 }  
 System.out.println();  
 }  
 System.out.println();  
 }  
  
}

**Esercizio 3**

Si consideri la classe Libro.java, riportata di seguito per comodità, che rappresenta informazioni su prodotti in vendita presso un esercizio commerciale. In particolare, per ogni prodotto interessa conoscere il codice (una stringa univoca che lo individua), una sua descrizione (anch'essa una stringa), nonche' la quantità di confezioni disponibile in magazzino.

public class Libro {  
    String codice;  
    String titolo;  
    int quant;  
    Libro prox;  
     
    public Libro(String c, String t, int q, Libro p) {  
        codice = c;  
        titolo = t;  
        quant = q;  
        prox = p;  
    }  
}

Scrivere una classe Libreria (quindi in un file Libreria.java) che gestisce una lista dei libri disponibili nella libreria. La classe ha un campo inizio di classe Libro, contenente il riferimento all'inizio della lista, e un costruttore che inizializza una lista vuota di libri. La classe ha inoltre i seguenti metodi pubblici:

- boolean esiste(String c): riceve in ingresso come parametro una stringa che rappresenta il codice di un libro e restituisce true qualora il libro sia disponibile in libreria, false altrimenti.  
- void aggiornaLibreria(String c, String t, int q): riceve in ingresso come parametri due stringhe che rappresentano il codice e il titolo di un libro e un intero (positivo, nullo o negativo) che rappresenta il numero di copie che sono state acquistate dalla libreria (se positivo) o vendute ai clienti (se negativo). Se il libro è già presente in libreria (dunque esiste un libro con lo stesso codice) allora il metodo deve semplicemente incrementarne la quantità del valore q, mentre il valore del parametro d va ignorato. Qualora la quantità risultante dopo la somma sia nulla o negativa, il libro va rimosso dalla lista, in quanto non più disponibile. Se invece il libro non è presente in libreria e q *è positivo*, il metodo deve creare un nuovo oggetto di classe Libro e aggiungerlo alla lista, inizializzandone i campi ai valori passati come parametri. Infine, se il libro non è presente in libreria e q *è negativo o nullo*, il metodo non deve apportare alcuna modifica.  
- int quantita(String c): riceve in ingresso come parametro una stringa c che rappresenta il codice di un libro e resituisce il numero di copie disponibilo. Qualora il libro non sia presente, il metodo deve restituire 0.

Scrivere le classi in modo da poter compilare ed eseguire il seguente programma di prova senza doverlo modificare.

public class **ProvaEserc3** {  
  
 public static void main(String[] args) {  
 Libreria myLib = new Libreria();  
   
 System.out.println("Aggiungo tre libri alla lista, tutti per la prima volta...");  
 myLib.aggiornaLibreria("lib1", "2000 leghe sotto i mari", 10);  
 myLib.aggiornaLibreria("lib2", "Il rosso e il nero", 20);  
 myLib.aggiornaLibreria("lib3", "Cuore di tenebra", 10);  
 System.out.println("... Devono comparire tutti nella lista");  
 stampa(myLib);  
 System.out.println("-------------------------");  
  
 System.out.println("Aggiornamento per i libri aventi codici lib2 e lib1");  
 System.out.println("Entrambi sono presenti nella lista ... devo aggiornare soltanto le quantità");  
 myLib.aggiornaLibreria("lib2", "Il rosso e il nero", 5);  
 myLib.aggiornaLibreria("lib1", "2000 leghe sotto i mari", 10);  
 System.out.println("... I libri devono comparire con info su quantità...");  
 stampa(myLib);  
 System.out.println("-------------------------");  
   
 System.out.println("Quantità del libro con codice lib1: " + myLib.quantita("lib1"));  
 System.out.println("-------------------------");  
 System.out.println();  
 System.out.println("Quantità del libro con codice lib2: " + myLib.quantita("lib2"));  
 System.out.println("-------------------------");  
   
 System.out.println("Esiste un libro con codice gx25? Deve restituire false. " + "Restituisce " + myLib.esiste("gx25"));  
 System.out.println("-------------------------");  
  
 System.out.println("Vendute 5 copie di 2000 leghe sotto i mari...");  
 myLib.aggiornaLibreria("lib1", "2000 leghe sotto i mari", -5);   
 System.out.println("Questo e' lo stato risultante dei libri in magazzino...");  
 stampa(myLib);  
 System.out.println("-------------------------");  
   
 System.out.println("Aggiungo 20 copie di Moby Dick...");  
 myLib.aggiornaLibreria("lib4", "Moby Dick", 20);  
 System.out.println("Questo e' lo stato risultante dei libri in magazzino...");  
 stampa(myLib);  
 System.out.println("-------------------------");  
 System.out.println("Vendute altre 15 copie di 2000 leghe sotto i mari...libro non piu' in magazzino");  
 myLib.aggiornaLibreria("lib1", "2000 leghe sotto i mari", -15);   
 System.out.println("Questo e' lo stato risultante dei libri in magazzino...");  
 stampa(myLib);  
 System.out.println("-------------------------");  
 }  
   
 private static void stampa(Libreria myLib) {  
 Libro p = myLib.inizio;  
 while (p != null) {  
 System.out.println("Codice libro: " + p.codice + "; " + "Titolo: " + p.titolo + "; " + "Quantità: " + p.quant);  
 System.out.println();  
 p = p.prox;  
 }  
   
 }  
  
}

**Esercizio 4**

public class **NodoBin** {  
 int valore;  
 NodoBin sin;  
 NodoBin des;  
   
 public **NodoBin**(int val, NodoBin s, NodoBin d) {  
 valore = val;  
 sin = s;  
 des = d;  
 }  
}

Data la rappresentazione collegata di alberi binari presentata sopra, i cui nodi contengono un'informazione di tipo intero, scrivere nella classe Eserc4.java un metodo statico public static int somma(NodoBin radice) che preso in ingresso il riferimento alla radice di un sottoalbero di un albero binario, restituisca la somma dei valori dei nodi nel sottoalbero considerato.   
Scrivere la classe in modo da poter compilare ed eseguire il seguente programma di prova senza doverlo modificare.

public class **ProvaEserc4** {  
  
 public static void main(String[] args) {  
 NodoBin n7 = new NodoBin(200,null, null);  
 NodoBin n6 = new NodoBin(80,n7,null);  
 NodoBin n5 = new NodoBin(120,null,null);  
 NodoBin n4 = new NodoBin(100,null,null);  
 NodoBin n3 = new NodoBin(15,n6,null);  
 NodoBin n2 = new NodoBin(2000,n4,n5);  
 NodoBin n1 = new NodoBin(10,n2,n3);  
   
 System.out.println("La somma di tutti i valori è 2525");  
 System.out.println("Il programma restituisce il valore " + Eserc4.somma(n1));  
 System.out.println();  
 System.out.println("La somma dei valori nel sottoalbero avente radice n2 è 2200");  
 System.out.println("Il programma restituisce il valore " + Eserc4.somma(n2));  
 System.out.println();  
 System.out.println("La somma dei valori nel sottoalbero avente radice n3 è 295");  
 System.out.println("Il programma restituisce il valore " + Eserc4.somma(n3));  
 System.out.println();  
 }  
  
}