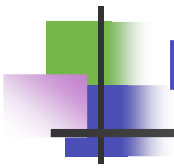
Esercizi



Definizione di classi

Una banca dati che contiene film

- Si vuol realizzare una banca dati che contiene informazioni su film.
- Un fillm è caratterizzato da un nome, un anno di uscita, un produttore e il costo. Si dovranno anche memorizzare gli attori che partecipano al film indicando per ciascuno i ruoli svolti (ad. es. attore protagonista, comparsa,…) e il compenso ricevuto.
- Si dovrà tenere traccia anche di tutti gli attori con il loro nome, cognome, la data di nascita e un curriculum.
- L'applicazione dovrà avere un'interfaccia utente e si vuol seguire il principio di separazione fra interfaccia e dati
- Per l'insieme degli attori si usi una lista per gli altri insiemi dei vettori

- Progettare le classi necessarie, per il momento, indicando solo le variabili e le classi (senza metodi e costruttori)

Una banca dati che contiene film

```
class Film{
   String nome, produttore;
   Date dataUscita:
   int costo;
 }
```

```
class Attore{
   String nome, cognome;
   String curriculum;
   Date dataNascita;
}
```

```
class Partecipazione{
  Attore  p;
  Film  f;
  String ruoli[];
}
```

```
class ListaAttori{
 ItemAttori head;
}
```

```
class ItemAttori{
  Attore  value;
  ItemAttori next;
}
```

```
class BancaDati{
  ListaAttori attori;
  Film films[];
  Partecipazione partecipazioni[];
}
```

```
class interfaccia{
 BancaDati b;
}
```

Una banca dati che contiene film: soluzione 2

```
class Film{
   String nome, produttore;
   Date dataUscita:
   int costo;
   Partecipazione partecipazioni[];
 }
```

```
class ListaAttori{
 ItemAttori head;
}
```

I cambiamenti sono in grassetto

```
class ItemAttori{
  Attore  value;
  ItemAttori next;
}
```

```
class Attore{
   String nome, cognome;
   String curriculum;
   Date dataNascita;
}
```

```
class BancaDati{
  ListaAttori attori;
  Film films[];
   //  Partecipazione partecipazioni[];
}
```

```
class Partecipazione{
  Attore  p;
 // Film f;
 String ruoli[];
}
```

```
class interfaccia{
 BancaDati b;
 }
```


Una banca dati che contiene film: i metodi

- Definire i metodi dell'interfaccia e della classe BancaDati
- Dovranno essere possibile le seguenti operazioni
  - inserire un film
  - rimuovere un film per titolo e anno
  - inserire un attore
  - rimuovere un attore per nome e cognome
  - cercare tutti i film a cui un attore ha partecipato fornendo il nome e il cognome dell'attore

Una banca dati che contiene film: i metodi

```
class BancaDati{
   ListaAttori attori;
   Film films[];
   Partecipazione partecipazioni[];

  void inserisciFilm(Film f);
  void rimuoviFilm(String titolo, Date anno);
  void inserisciAttore(Attore a);
  void rimuoviAttore(String nome, String cognome);
  Film[] searchFilm(String nome, String cognome;
}
```

```
class interfaccia{
  BancaDati b;
  void inserisciFilm();
  void rimuoviFilm();
  void inserisciAttore();
  void rimuoviAttore();
  void  searchFilm();
}
```


Un'applicazione per la biblioteca

- Una biblioteca vuol realizzare un'applicazione per gestire un archivio dei libri disponibili. L'applicazione dovrà memorizzare tutti i libri posseduti, tutti gli autori dei libri e tutti gli editori. Per ogni autore si registra il nome, il cognome, il luogo di nascita e una breve biografia. Per ogni editore si registra il nome e l'indirizzo. Per ogni libro si registra il titolo, gli autori, l'editore e l'ISBN che è un codice di 13 cifre.
- L'applicazione dovrà possedere un menu utilizzando il quale sara' possibile inserire un libro, un autore o un editore. Inoltre, dovrà essere possibile eliminare un libro identificandolo con il codice ISBN. Infine, il menu dovra' permettere di cercare tutti i libri posseduti dalla biblioteca per titolo o autore. Quest'ultima funzione deve visualizzare tutti i libri che soddisfano il criterio di ricerca.
- Per realizzare l'insieme degli editori usare una struttura dati di tipo lista. Per gli altri insieme usare una struttura di tipo vettore.

Un'applicazione per la biblioteca

- Progettare le classi Java necessarie per realizzare l'applicazione.
- Si richiede la definizione delle classi, variabili e metodi. Non è necessario implementare i metodi.
- Indicare anche la classe che contiene il main.

Un'applicazione per la biblioteca

```
class Autore{

   String nome, cognome, biografia;
   Date dataNascita:
}
```

```
class Editore{

   String nome, indirizzo;
}
```

```
class  Libro{

   String titolo;
   Autore autoriLibro[];
   Editore editoreLibro;
   int  ISBN[];
}
```

```
class EditorList{

  EditorItem head;

   void insert(int v);
   void remove(int v);
    EditorItem search(int v);
}
```

```
class EditorItem

   Editore value;
   EditorItem next;
}
```

Un'applicazione per la biblioteca

```
class Biblioteca{

   Libro libri[];
   Autore autore[];
   EditorList editori[];

   void inserireAutore(String nome, String cognome,
                       String biografia, Date dataNascita);
   void inserireEditore(String nome, String indirizzo);
   void inserireLibro(String titolo, Autore autori[],
                      Editore editore, int  ISBN[]);
   void rimuoviLibro(int ISBN[]);
   Libro[] cerca(String titolo);
   Libro[] cerca(Autore autore);
}
```


Un'applicazione per la biblioteca

```
class Menu{


   Biblioteca biblioteca;

   void inserireAutore();
   void inserireEditore();
   void inserireLibro();
   void rimuoviLibro();
   void cerca1();
   void cerca2();

   static void main(String argv);

}
```

Un'applicazione per la segreteria studenti

- Una segreteria studenti vuol realizzare un'applicazione per gestire un archivio con gli studenti, i corsi e gli esami di un anno accademico. L'applicazione dovrà memorizzare tutti gli studenti, tutti gli esami e tutti i corsi di una facoltà. Per ogni corso si registra il nome e il docente. Per ogni studente si registra il nome, il cognome, la matricola e la data di nascita. Per ogni esame si registra il voto, la data e ovviamente il corso e lo studente.
- L'applicazione dovrà possedere un menu utilizzando il quale sarà possibile inserire uno studente, un esame o un corso. Inoltre, dovrà essere possibile ricercare uno studente per matricola o cognome e nome. Allo stesso modo dovrà essere possibile rimuoverlo indicando il numero di matricola. Infine, l'applicazione dovrà permettere di calcolare le media di ogni studente e ogni corso
- Per realizzare l'insieme degli studenti, degli esami e dei corsi usare una struttura dati di tipo vettore

Un'applicazione per la segreteria studenti

- Progettare le classi Java necessarie per realizzare l'applicazione.
- Si richiede la definizione delle classi, variabili e costruttori.
- Non è necessario implementare i metodi. Indicare anche la classe che contiene il main.

Un'applicazione per la segreteria studenti

```
class Studente{
   String nome, cognome, matricola;
   Date dataNascita:
 }
```

```
class Corso{
   String nome, docente;
}
```

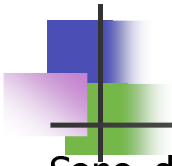
```
class Esame{
   int voto;
   boolean lode;
   Date data;
   Studente studente;
   Corso corso;
}
```

```
class Archivio{

   Studente studenti[];
   Corso corsi[];
   Esame esami[];

void inserireStudente(String nome, String cognome,
                      String matricola, Date dataNascita);
   void inserireCorso(String nome, String docente);
   void inserireEsame(Studente s,
                      int voto, boolean lode, Date data);
   Studente cerca(String matricola);
   Studente cerca(String cognome, String nome);
   void rimuovi(String matricola);
   float mediaStudente(String matricola);
   float mediaCorso(string nome);
}
```


Un'applicazione per la segreteria studenti

```
class Menu{

    Archivio archivioSegreteria;

   void inserireStudente();
   void inserireCorso();
   void inserireEsame();
   void cercaStudentePerMatricola();
   void cercaStudentePerNome();
   void rimuoviStudente();
   void mediaStudente();
   void mediaCorso();

  static void main(String arg[]);

}
```

Uso di classi definite



Uso di una pila

- Sono date le classi che implementano una pila di messaggi: creare il codice Java che
- crea una pila con al più 10 messaggi
- inserisce i messaggi:
  - "messaggio 1", inviato a "marte", indirizzo ip "10.0.0.1"
    da "giove", indirzzo ip "10.0.0.2"
  - "risposta 1", inviato a "giove", indirizzo ip "10.0.0.2"
    da "marte", indirzzo ip "10.0.0.1"
- estrarre dalla pila tutti i messaggi stampando il testo, il nome del destinatario e del mittente

```
class Pila{
   Pila(int maxMessaggi);
   Messaggio Pop();
   Messaggio Top();
   void Push(Messaggio m);
   boolean isEmpty();
 }
```

```
class Messaggio{
  String testo;
  Indirizzo destinatario, mittente;
  Messaggio(Indirizzo mitt, Indirizzo dest, String testo);
}
```

```
class Indirizzo{
   String nome, ip;
   Indirizzo(String nome, String ip)
 }
```

Uso di una pila

```
Pila p=new Pila(10);
Indirizzo marte=new Indirizzo("marte","10.0.0.1");
Indirizzo giove=new Indirizzo("giove","10.0.0.2");
Messaggio m1=new Messaggio(giove,marte,"messaggio 1");
Messaggio m2=new Messaggio(marte,giove,"risposta 1");
p.push(m1);
p.push(m2);
while(!p.isEmpty()){
        Messaggio m=p.pop();
        System.out.print(m.testo)
        System.out.print(" da "+m.mittente.nome)
        System.out.println(" a "+m.destinatario.nome)
}
```

Uso di un archivio studenti

- Sono date le classi che implementano un archivio degli studenti. Scrivere il codice Java che
- crea un archivio
- inserisce
  - "Paolo", "Rossi" residente in "via Verdi 16, Siena"
  - "Roberto", "Bianchi" residente in "via Gialli 10, Siena, 53110"
- cerca tutti gli studenti abitanti a Siena e ne stampa il cognome

```
class Studente{
  String Nome, Cognome;
  Indirizzo residenza
  Studente(String nome, String cognome, String residenza)
}
```

```
class Indirizzo{
   String via, numero, citta;
   Indirizzo(String via, String numero, String citta)
}
```

```
class Archivio{
   private Studente studenti[];
  Archivio();
  void inserisci(Studente s)
  Studente [] cerca(String city)

 }
```

Uso di un archivio studenti

```
Archivio a=new Archivio();
Indirizzo i1=new Indirizzo("via Verdi","16","Siena");
Indirizzo i2=new Indirizzo("via Gialli","10","Siena");
Studente paoloRossi=new Studente("Paolo","Rossi",i1);
Studente robertoBianchi=new Studente("Roberto","Bianchi",i2);
a.inserisci(paoloRossi);
a.inserisci(robertoBianchi);
Studente [] studentiDiSiena=a.search("siena");
for(int i=0;i<studentiDiSiena.length;i++){
        System.out.println(studentiDiSiena[i].cognome);
}
```


Implementazione di metodi

Somma con una lista di operazioni

- Sono date le classi che implementano una lista di operazioni (ad esempio di un conto corrente)
- Ogni operazione è caratterizzata dalla quantità e da un carattere che è 'P' se l'operazione è un prelievo e 'D' se l'operazione è un deposito
- Si implementi (in maniera iterativa) un metodo che calcola totale delle operazioni contano come negativi i prelievi e positivi i depositi

```
class ListaOperazioni{
   itemOperazione head;
 }
```

```
class itemOperazione{

   operazione value;
   itemOperazione next;
}
```

```
class operazione{
  operazione(float quantita, char pd);
  char getPd();
  float getQuantita();
}
```

float calcolaTotale(ListaOperazioni A)

Somma con una lista di operazioni

```
float calcolaTotale(ListaOperazioni A){

     float somma=0;
     for(itemOperazione it=A.head; it!=null; it=it.next){
          if(it.value.getPd()=='D') {
                    somma=somma+it.value.getQuantita();}
          else{
                     somma=somma-it.value.getQuantita();}
    }
    return somma;
}
```

Somma con una lista di operazioni II

- La stessa funzione potrebbe essere implementata in maniera ricorsiva. Per farlo occorre implementare il metodo: calcolaTotaleRicorsivo

```
float calcolaTotale(ListaOperazioni A){
        return calcolaTotaleRicorsivo(A.head);
}

float  calcolaTotaleRicorsivo(ItemOperazioni it)
….
```


Somma con una lista di operazioni II

```
float  calcolaTotaleRicorsivo(ItemOperazioni it){
  if(it==null) {
        return 0;
        }
    else{if(it.value.getPd()='D'){
                return it.value.getQuantita()+calcolaTotaleRicorsivo(it.next);
                }
        else{
                return -it.value.getQuantita()+calcolaTotaleRicorsivo(it.next);
                }
        }
}
```

Profondità di un albero binario

- Sono date le classi che implementano un albero binario di interi
- Si implementi un metodo ricorsivo che calcola la profondità dell'albero
- Si realizzi il metodo profonditaRicorsiva

```
class Node{
   int value;
   nodo left, right;
}
```

```
class Tree{
  root;
}
```

```
int profondita(Tree T){
        return profonditaRicorsiva(T.root);
}

int profonditaRicorsiva(nodo n);
```


Profondità di un albero binario

```
int profonditaRicorsiva(nodo n){
    if(n==null){return 0;}
      else{
          int pDestra= profonditaRicorsiva(n.right);
          int pSinistra= profonditaRicorsiva(n.left);
          if(pDestra>pSinistra) {
                  return pDestra+1;}
          else{
                  return pSinistra+1;}
          }
}
```

Norma uno

- Scrivere un metodo che scorre una matrice a due dimensioni "float B[][]" e ne calcola la norma 1.
- La norma 1 di una matrice e' il massimo per colonna della somma dei moduli degli elementi

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,m} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,m} \end{pmatrix} \qquad \|B\|_1 = \max_{i=1}^{n} \left(\sum_{j=1}^{m} |b_{i,j}| \right)$$

Norma uno

```
float normaUno(float B[][]){
  float norm=0;
   for(int i=0;i<B.length;i++){
       float col=0;
       for(int j=0;j<B[i].length;j++){
           col=col+Math.abs(B[i][j]);
       }
       if(col>norm){norm=col;}
  }
  return norm;
 }
```

Norma uno II

- La stessa funzione potrebbe essere implementata in maniera ricorsiva. Per farlo occorre implementare il metodo normaUnoRicorsivo che calcola ricorsivamente la somma di ogni colonna

```
float normaUno(float B[][])){
        float a= normaUnoRicorsivo(B,0,B.length);
        float max=Integer.MAX_VALUE;
        for(int i=0;i<B[0].length;i+){
                if(max<a[i]) {max=a[i];}
        }
        return max;
}

float []  normaUnoRicorsivo (float B[][], int minRow, int maxRow)
….
```


Norma uno II

```
float  normaUnoRicorsivo (float B[][], int minRow, int maxRow){

        int[] res=new int[B[0].length];
        if(maxRow==minRow){
                for(int i=0;i<res.length;i++){
                        res[i]=B[minRow][i];
                }
        }
        else{
                int [] a= normaUnoRicorsivo (float B[][], minRow+1,maxRow);
                for(int i=0;i<res.length;i++){
                        res[i]=B[minRow][i]+a[i];
                }
        }
        return res;

}
```

Somma elementi maggiori di zero

- Sono date le classi che implementano una lista di interi: scrivere il metodo java che calcola la somma

Variabili e metodi della lista

```
class Lista{
    Item head;

   void insert(int v);
   void remove(int v);
   Item search(int v);
}
```

Un elemento della lista

```
class Item{

   int value;
   Item next;
}
```

Il metodo da implementare

```
int somma(List lista)
```


Somma elementi maggiori di zero

```
int somma(List A){
   int res=0;
   for(Item it=A.head; it!=null; it=it.next){
            if(it.value>0) {
                     res=res+it.value;
            }
   }
   return res;
}
```

Somma elementi maggiori di zero II

- La stessa funzione potrebbe essere implementata in maniera ricorsiva. Per farlo occorre implementare il metodo sommaRicorsivo

```
int somma(List lista){
        return sommaRicorsivo(Lista.head);
}

int sommaRicorsivo(Item it)
```


Somma elementi maggiori di zero II

```
int sommaRicorsivo(Item it){

        if (it==null) {
                return 0;}

        if(it.value>=0){
                return it.value+ sommaRicorsivo(it.next);}

        else {
                return  sommaRicorsivo(it.next);}
}
```

Somma del costo di una lista di articoli

- Sono date le classi che implementano una lista di Articoli in vendita: scrivere il metodo che calcola la somma del costo degli articoli di un certo tipo

Variabili e metodi della lista

```
class Lista{
    Item head;

   void insert(int v);
   void remove(int v);
   Item search(int v);
}
```

Un elemento della lista

```
class Item{

   Articolo value;
   Item next;
}
```

Un articolo

```
class Articolo{

   float costo;
   String nome;
   int codice;
}
```

Il metodo da implementare

```
int somma(List lista, int codice)
```


Somma del costo di una lista di articoli

```
float somma(List A, int codice){
   int res=0;
   for(Item it=A.head; it!=null; it=it.next){
            if(it.value.codice==codice) {
                     res=res+it.value.costo;
            }
   }
   return res;
 }
```

Somma del costo di una lista di articoli

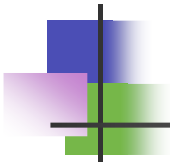
- La stessa funzione potrebbe essere implementata in maniera ricorsiva. Per farlo occorre implementare il metodo sommaRicorsivo

```
float somma(List lista, int codice){
        return sommaRicorsivo(Lista.head,int codice);
}

float sommaRicorsivo(Item it, int codice)
```


Somma elementi maggiori di zero II

```
float sommaRicorsivo(Item it, int codice){

        if (it==null) {
                return 0;}

        if(it.value.codice=codice){
                return it.value.costo+ sommaRicorsivo(it.next);}

        else {
                return  sommaRicorsivo(it.next);}
}
```

Calcolo della complessità



Complessita' di un programma

- Calcolare l'ordine asintotico (notazione O(f(n))) del numero di moltiplicazioni svolte dal seguente metodo rispetto al valore del parametro n

```
void metodo2(int n){

  int i=1;
  int res=1;

  while(i<=n){

       for(int j=0;j<n;j++){
           res=res*j+i;
       }

       i=2*i;
   }

}
```

Il corpo del while viene eseguito $\log_2 n$ volte

Il corpo del for viene eseguito n volte

Risposta
$O(n*\log_2 n)$

Complessita' di un programma

- Calcolare l'ordine asintotico (notazione O(f(n))) del numero di moltiplicazioni svolte dal seguente metodo rispetto al valore del parametro n
- Nel caso servisse, si ricorda che… $\sum_{i=1}^{a} i = \frac{a(a+1)}{2}$

```
void metodo2(int n){

  int res=1;

  for(int i=1;i<=Math.pow(n,2);i++){

       for(j=1;j<=i;j++){
           res=res*2;
       }

   }

}
```

Il corpo del for viene eseguito n^2 volte

Il corpo del for viene eseguito i volte

Poiche'…

$$\sum_{i=1}^{n^2} i = \frac{n^2(n^2+1)}{2}$$

Risposta $O(n^4)$

Complessita' di un programma

- Calcolare l'ordine asintotico (notazione O(f(n))) del numero di somme svolte dal seguente metodo sulla variabile res rispetto al valore del parametro n

```
void metodo3(int n){

  int i=1;
  int res=1;

  while(i<=Math.pow(n,2)){

       for(int j=0;j<Math.pow(n,3);j++;){
           res=res+3;
       }

       i=2*i;
   }

}
```

Il corpo del while viene eseguito 2 $\log_2 n$ volte

Il corpo del for viene eseguito n^3 volte

Risposta $O(n^3 * \log_2 n)$

Complessita' di un programma

- Calcolare l'ordine asintotico (notazione O(f(n))) del numero di moltiplicazioni svolte dal seguente metodo rispetto al valore del parametro n
- Nel caso servisse, si ricorda che... $\sum_{i=1}^{a} i^2 = \frac{2a^3 + 3a^2 + a}{6}$

```
void metodo4(int n){

  int res=1;

  for(int i=Math.pow(n,3);i>=1;i--){

        for(j=1;j<=Math.pow(i,2);j++){
            res=res*2;
        }

   }

}
```

Il corpo del for viene eseguito n^3 volte

Il corpo del for viene eseguito i^2 volte

Poiche'...

$$\sum_{i=n^3}^{1} i^2 = \sum_{i=1}^{n^3} i^2 = \frac{2n^9 + 3n^6 + n^2}{6}$$

Risposta $O(n^9)$

Complessita' di un programma

- Calcolare l'ordine asintotico (notazione O(f(n))) del numero di moltiplicazioni svolte dal seguente metodo rispetto al valore del parametro n

Poiche'...

$$\sum_{i=1}^{n^2} 3^i = \frac{3^{n^2+1} - 1}{3 - 1}$$

```
void metodo5(int n){

  int res=1;

  for(int i=1;i<=n*n;i++){

        for(j=1;j<=Math.pow(3,i);j++){
            res=res+2;
        }

   }

}
```

Il corpo del for viene eseguito n^2 volte

Il corpo del for viene eseguito 3^i volte

Risposta $O(3^{n^2})$

Complessita' di un programma

- Calcolare l'ordine asintotico (notazione O(f(n))) del numero di somme svolte dal seguente metodo sulla variabile res rispetto al valore del parametro n

```
void metodo6(int n){
  int res=1;
  for(int i=1;i<=Math.log10(n);i=i+1){

       for(int j=0;j<Math.pow(2,i);j++;){
           res=res+3;
       }
  }
}
```

Il corpo del while viene eseguito $\log_3 n$ volte

Il corpo del for viene eseguito 2^i volte

Risposta O(n)

Poiche'...

$$\sum_{i=1}^{\log_{10} n} 2^i = \frac{2^{\log_{10} n+1}-1}{3-1} = \frac{2^{\frac{\log_2 n}{\log_2 10}+1}-1}{2} = \frac{2^{\frac{1}{\log_2 10}+1}\,2^{\log_2 n}-1}{2} = 2^{\frac{1}{\log_2 10}} n - 1$$

Formule utili

$$\sum_{i=1}^{a} i = \frac{a(a+1)}{2}$$

$$\sum_{i=1}^{a} i^2 = \frac{2a^3+3a^2+a}{6}$$

$$\sum_{i=1}^{a} b^i = \frac{b^{a+1}-1}{b-1}$$