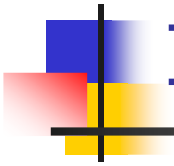
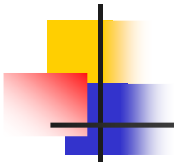


Implementazione di metodi



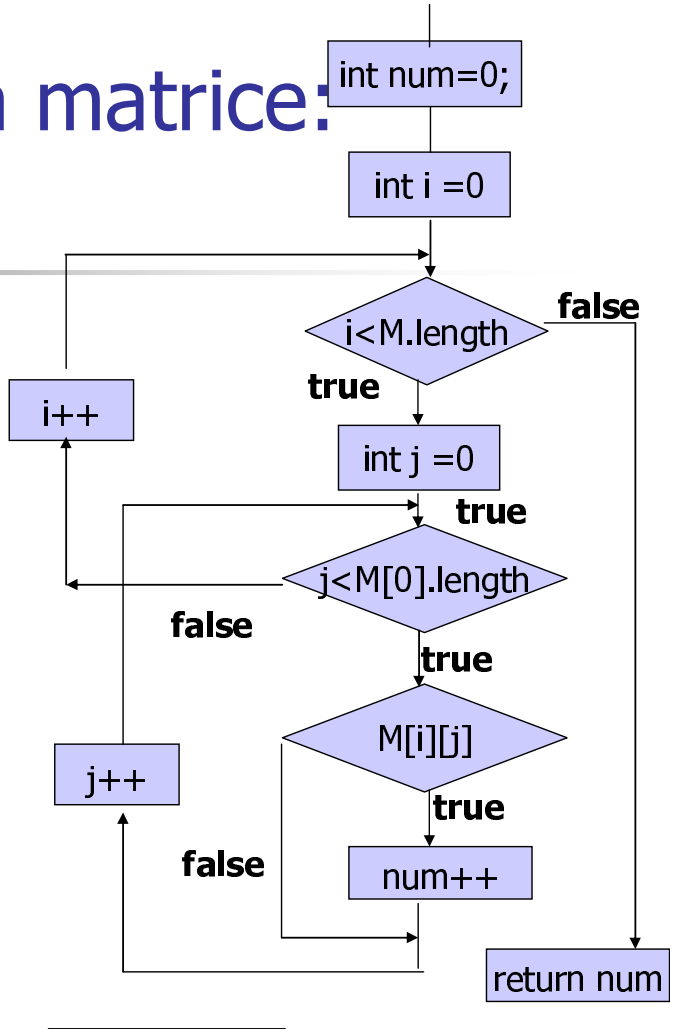
Ricerca in una matrice

- Implementare un metodo che preso in ingresso una matrice di conti i numeri negativi conenuti nella matrice
- disegnare il rispettivo diagramma di flusso

int count(**int** M[][])

Ricerca in una matrice: soluzione

```
int  count(int M[][]){
  int num=0;
  for(int i=0;i<M.length;i++){
       for(int j=0;j<M[0].length;j++){
            if(M[i][j]<0){
                  num++;
                  }
     }
  }
 return num;
}
```



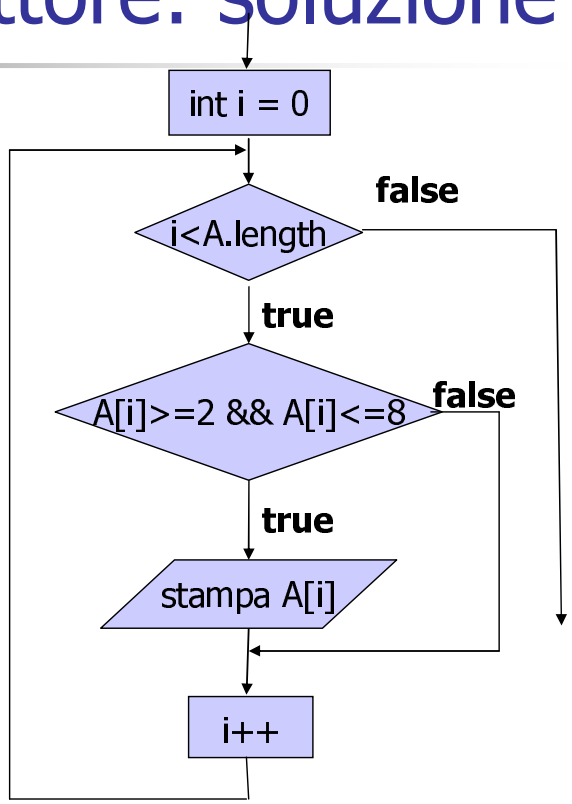
Stampa dei valori compresi fra 2 e 8 di un vettore

- Implementare un metodo che preso in ingresso un vettore, stampa i valori compresi fra 2 e 8
- disegnare il rispettivo diagramma di flusso

```
void  stampaFra2e8(int A[])
```

Stampa dei valori compresi fra 2 e 8 di un vettore: soluzione

```
void  stampaFra2e8(int A[]){
  for(int i=0;i<A.length;i++){
     if(A[i]>=2 && A[i]<=8){
            System.out.println(A[i]);}
     }
}
```



Stampa dei valori compresi fra 2 e 8 di una lista

- Implementare un metodo che preso in ingresso una lista di interi, stampa i valori compresi fra 2 e 8
- disegnare il rispettivo diagramma di flusso

```
void  stampaFra2e8(List A[])
```

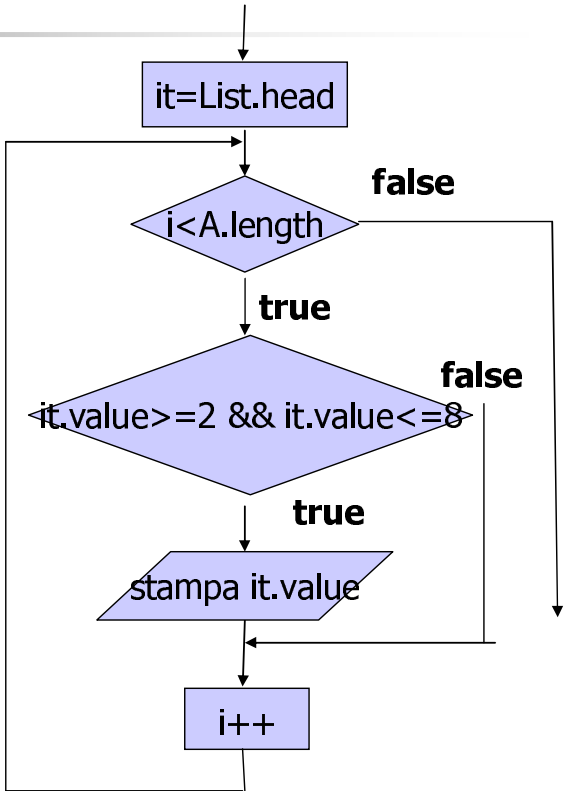
Classi disponibili

```
List {
   Item head;
}

Item {
  Item next;
  int value;
}
```

Stampa dei valori compresi fra 2 e 8 di un vettore: soluzione

```
void  stampaFra2e8(List A){
  for(Item it=List.head;it!=null;it=it.next){
     if(it.value>=2 && it.value<=8){
           System.out.println(it.value);}
     }
}
```



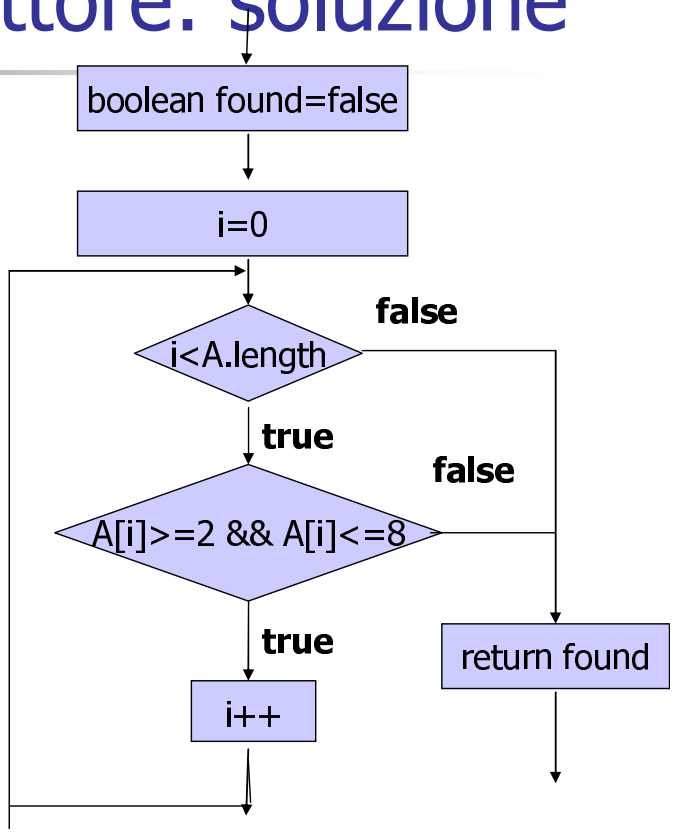
Ricerca dei valori compresi fra 2 e 8 di un vettore

- Implementare un metodo che preso in ingresso un vettore, restituisce vero o falso a seconda che contenga o meno un valore compreso fra 2 e 8
- disegnare il rispettivo diagramma di flusso

```
boolean contieneFra2e8(int A[])
```

Stampa dei valori compresi fra 2 e 8 di un vettore: soluzione

```
boolean contieneFra2e8(int A[]){
  boolean found=false;
  for(int i=0;i<A.length;i++){
     if(A[i]>=2 && A[i]<=8){
            found=true;
            break;}
     }
 return found;
}
```



Media di un vettore

- Implementare un metodo che presa in ingresso un vettore, calcoli la media degli elementi maggiori di 7
- disegnare il rispettivo diagramma di flusso

```
float media(int A[])
```

Media di una lista

- Implementare un metodo che preso in ingresso una lista di interi ne calcoli la media
- disegnare il rispettivo diagramma di flusso

Classi disponibili

```
List {
   Item head;
}

Item {
  Item next;
  int value;
}
```

```
float media(List A)
```

Ricerca di elementi comuni in due array

- Implementare un metodo che presi in ingresso due vettori, restituisca un vettore che contiene l'intersezione dei due vettori, cioè i valori contenuti in entrambi i vettori
- disegnare il rispettivo diagramma di flusso

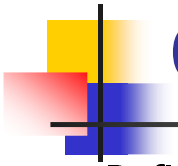
```
int [] intersection(int A[], int B[])
```

Ricerca di elementi comuni in due array

- Implementare un metodo che presi in ingresso due vettori, restituisca vero o falso a secondo se i due vettori contengono un valore in comune
- disegnare il rispettivo diagramma di flusso

```
boolean  isIntersectionEmp(int A[], int B[])
```

Definizione classi

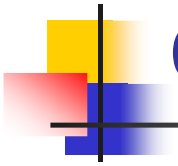


Classe libro

Definire una classe che rappresenti un libro. Si richiede che

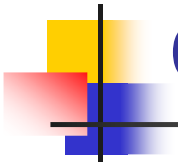
- Si memorizzino il titolo, il numero di catalogo e le pagine del libro (definite dalla classe a pagina successiva)
- Si possa creare un oggetto libro indicandone il titolo e il numero di catalogo
- Si possa inserire una nuova pagina
- Si possa rimuovere una pagina indicandone il numero
- Si possa ricercare tutte le pagine che contengono una data parola
- Si possa ricercare una pagina indicandone il numero
- Si possa calcolare il numero delle pagine

Definire metodi, costruttori e variabili della classe senza implementarli



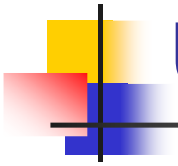
Classe pagina

```
class Pagina {
    String testo;
    int numero;
    Pagina(String testo, int numero);
}
```



Classe libro: soluzione

```
class Libro{
  String titolo;
  String numeroCatalogo;
  Libro(String titolo,String numeroCatalogo);
  Pagina  pagine[];
  void inserisci(Pagina p);
  void rimuovi(Pagina p);
  Pagina [] ricerca(String parola);
  Pagina ricerca (int numero);
  int numeroPagine();
}
```



Uso della classe libro

Dopo aver scritto la definizione della classe libro, come descritto nel precedente esercizio, si scriva il codice che

- crea un oggetto libro dal titolo "Informatica 1" avente come numero di catalogo 10
- inserisce la pagina 1 contentente il testo "bla bla"
- inserisce la pagina 2 contentente il testo "ri-bla bla"
- stampi il numero delle pagine

Uso della classe libro: soluzione

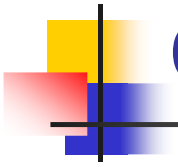
```
Libro l= new Libro("Informatica 1","10");
Pagina p1=new Pagina("bla bla",1);
Pagina p2=new Pagina("ri-bla bla",2);
l.inserisci(p1);
l.inserisci(p2);
System.out.println(l. numeroPagine());
```

Classe nazione

Definire una classe che rappresenti una nazione. Si richiede che

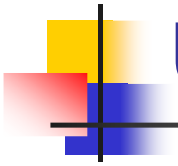
- Si memorizzino il nome, il numero di abitanti e le città (definite dalla classe a pagina successiva)
- Si possa creare un oggetto nazione indicandone il nome e il numero di abitanti
- Si possa inserire una nuova città
- Si possa rimuovere una città indicandone il nome
- Si possa ricercare tutte le città di una provincia
- Si possa ricercare una città indicandone il nome
- Si possa calcolare il numero delle città

Definire metodi, costruttori e variabili della classe senza implementarli



Classe Citta

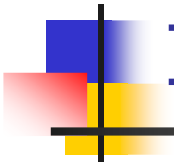
```
class Citta {
    String nome;
    String provincia;
    Pagina(String nome, String provincia);
}
```



Uso della classe nazione

Dopo aver scritto la definizione della classe nazione, come descritto nel precedente esercizio, si scriva il codice che

- crea un oggetto nazione dal nome "italia" avente 59.000.000 abitanti
- inserisce la città "Fucecchio" provincia di Firenze
- inserisce la città "San Gimignano" provincia di Siena
- stampi il numero delle città



Implementazione di metodi

Algebra booleana

- Date le variabili di fianco, scrivere l'espressione Java tale che
  1. è falsa se e solo se "a è nell'intervallo chiuso [3,7] oppure a è minore di b"
  2. è vera se e solo se "d è il carattere A e c non è nessuno dei caratteri A,B,C,D oppure"
  3. è falsa se e solo se "a è un numero visibile per 8 compreso fra 10 e 60, estremi compresi, e c non è il carattere A"
  4. è vera se e solo se "c è uno dei caratteri R,S oppure a non è la metà di b"
- Trasformare poi le espressioni usando la legge di De Morgan fino a eliminare le occorrenze dei simboli ! e lasciando solo &&,!=, >=, >, < ,<=

```
int a, b;
char c,d;
```