# Strutture Software 1

Docente: Fabio Solari
Tel.: 010-3532289
E-mail: fabio@dibe.unige.it
URL: http://www.pspc.dibe.unige.it

## INTRODUZIONE

- Presentare i principali metodi utilizzati per organizzare e rappresentare l'*informazione* (le strutture dati) al fine di ottenerne una *elaborazione* efficiente (gli algoritmi).
- Gli algoritmi vengono presentati dal punto di vista delle strutture dati.
- Le strutture dati vengono presentate in un ambiente orientato agli oggetti.



## INTRODUZIONE

- Le strutture dati sono introdotte prima come tipi di dati astratti, definite mediante la loro interfaccia.
- In seguito si passa alla loro realizzazione concreta in un linguaggio di programmazione.
- Infine si mostrano alcuni esempi significativi di un loro utilizzo in ambito ingegneristico.



## SOMMARIO

- Richiami di programmazione: I/O dati, array, ricorsione.
- Analisi della complessità degli algoritmi: notazione O-grande.
- Ordinamento e ricerca: algoritmi elementari e algoritmi efficienti.
- Pile e code: specifica e implementazione.
- Liste : specifica e implementazione.
- Insiemi e tabelle hash : specifica e implementazione.
- Java Collections Framework.

# RIFERIMENTI BIBLIOGRAFICI

- Dispense fornite a lezione.
- Testi di consultazione:
  - Adam Drozdek. Algoritmi e strutture dati in Java. Apogeo, 2001.
  - Robert Sedgewick. Algoritmi in Java. Addison Wesley, 2003.
- Per i linguaggi *C* e *Java* si fa riferimento agli insegnamenti di "Informatica 1" e "Programmazione ad oggetti per sistemi elettronici 1".



# SOMMARIO: richiami di programmazione

- I/O dati:
  - tastiera e file.
  - Conversione tra stringhe e numeri.
  - Argomenti alla linea di commando.
- Generazione di numeri pseudo-casuali.
- Array:
  - 1D e 2D di dati primitivi e oggetti.
  - Matrici.
- Ricorsione: analisi del comportamento in casi specifici.



# I/O DATI: introduzione

- In genere, per avere un'applicazione pratica i programmi devono avere uno scambio di dati con l'esterno: in modo interattivo con la tastiera e il monitor, oppure attraverso file.
- Vediamo alcuni esempi in modo da uniformare le modalità di gestione dell'input/output di dati.
- Il primo esempio riguarda la lettura da tastiera di due valori e la loro visualizzazione a monitor.



# I/O INTERATTIVO

```
import java.io.*;

public class Es1{
   public static void main(String[] args) throws
                                        IOException{
        BufferedReader stdin = new BufferedReader(new
                     InputStreamReader(System.in));
        System.out.print("Inserisci un float: ");
        float x = Float.parseFloat(stdin.readLine());
        System.out.println("Inserito " + x);
        System.out.print("Inserisci una stringa: ");
        String str = stdin.readLine();
        System.out.println("Inserito " + str);
   }
}
```

```
java Es1
Inserisci un float: -1.2e2
Inserito -120.0
Inserisci una stringa: fine
Inserito fine
```

# I/O CON FILE

- Leggere un file (`file1.txt`) contenete un elenco di articoli e relativi prezzi e scrivere un nuovo file (`file2.txt`) con i prezzi maggiorati del 20%.

`file1.txt`

```
Articolo1
1.2
Articolo2
4.6
Articolo3
10.0
```

`file2.txt`

```
Articolo1
1.44
Articolo2
5.52
Articolo3
12.0
```



# I/O CON FILE

```
import java.io.*;
public class Es2{
   public static void main(String[] args) throws
                                        IOException{
        BufferedReader in = new BufferedReader(new
              FileReader("file1.txt"));
        PrintWriter out = new PrintWriter(new
              BufferedWriter(new FileWriter("file2.txt")));
        String str1, str2;
        while( ((str1=in.readLine()) != null) &&
                       ((str2=in.readLine()) != null)){
               out.println(str1);
               float x= Float.parseFloat(str2);
               x=x*1.2f;
               out.println(x);
        }
        in.close();
        out.close();
   }
}
```



# STRINGHE

- Se la linea letta con `readLine()` contenesse diversi elementi, si possono separare come nel seguente esempio:

```
public class Es3{
   public static void main(String[] args) {
        String str = "Una stringa composta da 6 parti";
        String[] strv;
        strv=str.split(" ");
        for (int i=0;i<strv.length;i++){
             System.out.println(strv[i]);
          }
   }
}
```

```
java Es3
Una
stringa
composta
da
6
parti
```



# ARGOMENTI ALLA LINEA DI COMMANDO

- I dati di input vengono forniti direttamente al momento di lanciare l'applicazione da riga di commando.
- Un esempio in cui si controlla che i valori inseriti siano 3 e poi si calcola la loro somma.

```
java EsLinComm  23  -12
Errore numero di argomenti
```

```
java EsLinComm  23  -12   4
La somma: 15.0
```

# ARGOMENTI ALLA LINEA DI COMMANDO

```
public class EsLinComm{
  public static void main(String[] args) {
     if (args.length!=3){
          System.out.println("Errore
               numero di argomenti");
          System.exit(-1);
     }
     float x=Float.parseFloat(args[0]);
     x+=Float.parseFloat(args[1]) +
          Float.parseFloat(args[2]);
     System.out.println("La somma: " +x);
  }
}
```



# GENERAZIONE DI NUMERI PSEUDO-CASUALI

- Utile per verificare algoritmi.
- Viene generata una sequenza di numeri pseudo-casuali identica se viene usato lo stesso seme per inizializzare l'oggetto.

- Un esempio di generazione di interi tra 0 ed un valore fornito al metodo e di `float` tra 0.0 e 1.0.



# GENERAZIONE DI NUMERI PSEUDO-CASUALI: ESEMPIO

```
import java.util.*;
public class Casuali{
  public static void main(String[] args) {
     Random rnd = new Random();
     for(int i=0;i<20;i++)
          System.out.print(rnd.nextInt(7) + " ");
     System.out.println();
     for(int i=0;i<5;i++)
          System.out.print(rnd.nextFloat() + " ");
     System.out.println();
  }
}
```

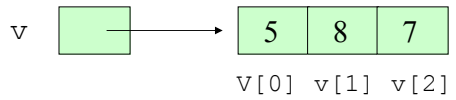
```
java Casuali
2 0 0 5 4 5 0 6 2 1 0 3 0 3 6 6 4 2 0 1
0.5498131 0.08636296 0.6110731 0.061515212 0.6021063
```



# ARRAY 1D: dati primitivi

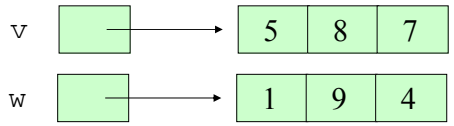
- Gli array sono un insieme di valori dello stesso tipo a cui si può far riferimento attraverso un indice: sono una struttura dati.

```
int[] v = new int[3];
v[0]=5; v[1]=8; v[2]=7;
```

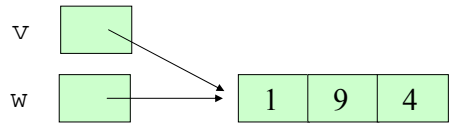


# ARRAY 1D: dati primitivi

```
int[] v = new int[3];
v[0]=5; v[1]=8; v[2]=7;
int[] w = {1,9,4};
```




```
v=w;
```






# ARRAY 1D: esempio

- Trovare il valore massimo e minimo di un array la cui dimensione è specificata in input, l'array viene inizializzato con valori casuali.

```
F:\temp>java MaxMin 5
29 45 35 19 4
(max,indice): 45 , 1
(min,indice): 4 , 4

F:\temp>java MaxMin 20
31 75 0 44 14 9 19 55 64 97 9 79 54 41 7 96 73 72 45 74
(max,indice): 97 , 9
(min,indice): 0 , 2
```



# ARRAY 1D: esempio

```
import java.util.*;

public class MaxMin{
   public static void main(String[] args) {
        Random rnd = new Random();

        int n = Integer.parseInt(args[0]);
        int[] v = new int[n];

        for(int i=0;i<v.length;i++){
               v[i]=rnd.nextInt(100);
               System.out.print(v[i] + " ");
        }
        System.out.println();
                                ...
```



# ARRAY 1D: esempio

```
                ...
        int max=v[0], min=v[0];
        int imax=-1,imin=-1;
        for(int i=0;i<v.length;i++){
               if (max<v[i]){
                      max=v[i];
                      imax=i;
               }
               if (min>v[i]){
                      min=v[i];
                      imin=i;
               }
        }
        System.out.println("(max,indice): "+max+" , "+imax);
        System.out.println("(min,indice): "+min+" , "+imin);
   }
}
```

# ARRAY 1D: oggetti

- Gli array di oggetti contengono riferimenti agli oggetti e i singoli elementi devono essere creati esplicitamente.

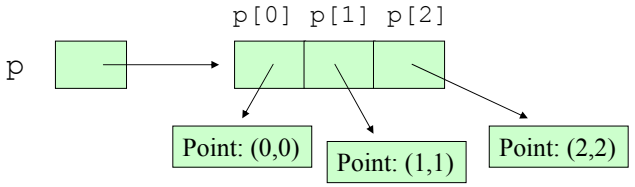
```
public class Point
{
  private int x;
  private int y;

  Point(int x, int y){
    this.x=x; this.y=y;
  }
  public void Set(int x, int y){
    this.x=x; this.y=y;
  }
  public String toString(){
        return "Point: ("+x+","+y+")";
  }
  ...
}
```

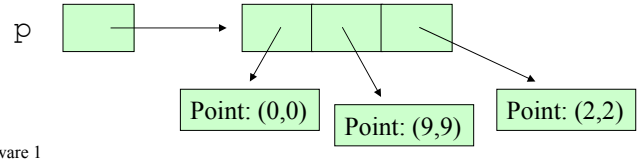


# ARRAY 1D: oggetti

```
Point[] p = new Point[3];
p[0]=new Point(0,0);
p[1]=new Point(1,1);
p[2]=new Point(2,2);
```




```
p[1].Set(9,9);
```






# ARRAY E METODI

- Vengono passati i riferimenti, pertanto sia il metodo chiamante sia il metodo chiamato modificano lo stesso array (oggetto).

```
F:\temp>java Azzera
1 2 3 4 5
0 0 0 0 0
```

- Un metodo può ritornare un riferimento ad un array creato al suo interno.

```
F:\temp>java SottoArray
0 1 2 3 4 5 6 7
2 3 4
```



# ARRAY E METODI

```
public class Azzera{
  public static void main(String[] args) {
      int[] v = {1,2,3,4,5};

      for(int i=0;i<v.length;i++)
            System.out.print(v[i] + " ");
      System.out.println();

      zero(v);

      for(int i=0;i<v.length;i++)
            System.out.print(v[i] + " ");
      System.out.println();
  }

  public static void zero(int[] a){
      for(int i=0;i<a.length;i++)
            a[i]=0;
  }}
```

# ARRAY E METODI

```
public class SottoArray{
   public static void main(String[] args) {
        int[] v = {0,1,2,3,4,5,6,7};
        print(v);
        int[] p = subArray(v,2,5);
        print(p);
   }

   public static int[] subArray(int[] a, int h, int k){
        int[] tmp = new int[k-h]; //...
        for(int i=h,j=0;i<k;i++,j++)
              tmp[j]=a[i];
        return tmp;
   }
   public static void print(int[] a){
        for(int i=0;i<a.length;i++)
              System.out.print(a[i] + " ");
        System.out.println();
   }}
```



# ARRAY 2D

- Consiste di un array i cui elementi sono a loro volta array. Tali sottoarray possono avere anche dimensione diversa tra loro.

```
int[][] m = new int[3][4];//matrice

int[][] p = new int[3][];
p[0] = new int [2];
p[1] = new int [3];
p[2] = new int [4];
```



# ARRAY 2D

```
int[][] m = new int[3][4];
int it=20;
for(int r=0;r<m.length;r++)
     for(int c=0;c<m[r].length;c++)
          m[r][c]=it++;
```