

Elementi d'istanza e di classe

Ogni istanza di una classe (oggetto) ha una propria "copia" dei **fields** su quell'oggetto, possono essere "applicati" delle azioni, tramite invocazioni di **metodi**

Tutti i fields e metodi visibili fino ad ora sono detti **di istanza**

• **CAMPO d'istanza**: ogni istanza derivata dalla classe mantiene una copia dei campi d'istanza.

• **METODO d'istanza**: metodi applicabili ad un'istanza della classe

1) Punto

```
public class Punto {  
  
    private int x;  
  
    private int y;  
  
    public static int allocs = 0;  
  
    public Punto() { // il costruttore senza parametri è un caso particolare di costruttore con parametri  
        this.x = 0; // this(0, 0); // viene delegato al secondo costruttore  
        this.y = 0; }  
  
    public Punto (int x, int y) {  
        this.x = x;  
        this.y = y;  
        allocs++; }  
  
    public int getX() {  
        return x; }  
  
    public void setX (int x) {  
        this.x = x; }  
  
    public int getY() {  
        return y; }
```

public: ci accedo anche dall'esterno

private: solo dall'interno

condiviso da tutte le classi

```

public void setY (int y) {

    this.y = y; }

public String toString () {

    return "(" + x + ", " + " + y + ")";

}

public Punto clone () {

    return new Punto (this.x, this.y); }

public boolean equals (Punto other) {

    return this.x == other.x && this.y == other.y; }

public static void main (String [] args) {

    Punto p1 = new Punto (1,2);

    Punto p2 = new Punto (3,4);

    System.out.println (p1);

    System.out.println (p2);

}

}

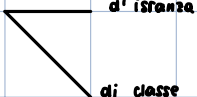
```

Campi di classe:

condivisi fra tutte le istanze di classe

→ esiste un'unica copia di campo di classe

I fields di una classe possono quindi essere



d'istanza
di classe

ognuno con la propria visibilità

Metodi di classe

Si applicano alla classe e non ad un oggetto

→ non ha un riferimento a this

→ non è possibile accedere ai campi d'istanza

`private static int allocs = 0;` [riferito a Punto]

`public static int getNumberOfAllocations () {`

`return allocs; }`

← e se ci voglio accedere dall'esterno

`System.out.println (Punto.getNumberOfAllocations ())`

↑
essendo un metodo di classe, ci accedo tramite la classe

e non l'istanza.

metodo main

viene chiamato dall'ambiente esterno al programma.

non ha quindi bisogno di creare un oggetto della classe che lo contiene

RICORDA: se è const in C++, è final in Java [`private static final double gravity = 9.81;`]

↑
è immutabile / costante però static

La classe ArrayUtility

compare

search

sort

`public class ArrayUtility <T> {`

`public static boolean compare (int[] a, int[] b)`

`if (a.length != b.length) {`

`return false; }`

`for (int i = 0; i < a.length; i++) {`

`if (a[i] != b[i]) {`

la classe è parametrica rispetto a

un tipo T (template)

`public static <T> boolean compare (T[] a, T[] b)`

`if (a.length != b.length) {`

`return false; }`

`for (int i = 0; i < a.length; i++) {`

`if (a[i].equals (b[i])) {`

con T, mi implica che a e b sono oggetti di una classe

```
return false;}
```

```
}
```

```
return true;}
```

```
public static boolean search (int[] a, int elem){
```

```
for (int i=0; i < a.length; i++){
```

```
if (a[i] == elem) {
```

```
return true;}
```

```
}
```

```
return false;}
```

```
public static void sort (int[] a){
```

```
private int save=0;
```

```
for (int i=0; i < a.length; i++){
```

```
if (a[i] > a[i+1]){
```

```
save = a[i];
```

```
a[i] = a[i+1];
```

```
a[i+1] = save;}
```

```
}
```

```
}
```

```
return false;}
```

```
}
```

```
return true;}
```