



UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Fondamenti di Informatica (117IN)

A.A. 2022 / 2023

Lezione 9 - Classi e orientamento agli oggetti (Parte 1)

Sylvio Barbon Junior
sylvio.barbonjunior@units.it



Sommario:

- 1) Lezione scorsa;
- 2) Classi, variabili e oggetti;
- 3) Visibilità;
- 4) Distruttori;
- 5) Esempio;

1) Lezione scorsa

```
1 public class L7_e1{
2
3     public static void main(String[] args) {
4         int numA = 10;
5         int numB = 12;
6         int result = sommaNum(numA, numB);
7         System.out.println("The result is:"+result);
8     }
9
10    public static int sommaNum(int x, int y){
11        int output = x + y;
12        return(output);
13    }
14 }
15
16
17
```

Classe (indicated by a purple bracket on the left)

- tipo risultato** (pink box) points to `void` in the `main` method signature.
- nome** (blue box) points to `main` in the `main` method signature.
- parametro** (cyan box) points to `String[] args` in the `main` method signature.
- chiamata della funzione.** (yellow box) points to the `sommaNum` call in line 6.
- Funzione `main`** (pink box) points to the entire `main` method body.
- tipo risultato** (pink box) points to `int` in the `sommaNum` method signature.
- parametri** (cyan box) points to `int x, int y` in the `sommaNum` method signature.
- Funzione `sommaNum`** (pink box) points to the entire `sommaNum` method body.
- ritorno** (orange box) points to `return(output);` in the `sommaNum` method body.
- nome** (blue box) points to `sommaNum` in the `sommaNum` method signature.
- tipo** (pink box) points to `public static` in the `main` method signature.

2) Classi, variabili e oggetti

- Una classe è un modello che **describe** una certa categoria di oggetti.
- Essa comprende un certo numero di **membri**:
 - variabili membro o campi dati;
 - funzioni membro o metodi;
 - costruttori (metodo chiamato per la creazione degli oggetti)

```
class MiaClasse{
    int n;
    MiaClasse(){
        /* ... */
    }
    int fai(int a){
        /* ... */
    }
}
```

← dichiarazione della classe

← dichiarazione della variabile

← dichiarazione del costruttori

← dichiarazione del metodo

2) Classi, variabili e oggetti

- Una variabile di un tipo classe (brevemente, variabile classe) è un riferimento di oggetti di quella classe;
- Un oggetto di un tipo classe (brevemente, oggetto classe) è un'istanza della classe (oggetto istanza): viene ottenuto per mezzo dell'operatore **new**;

```
MiaClasse varA;
```

```
varA = new MiaClasse();
```

```
MiaClasse varB = new MiaClasse();
```

2) Classi, variabili e oggetti

classe

metodo costruttore (stesso
nome della classe)

metodo "non" statico >>> oggetto

metodo statico >>> classe

```
1 public class MiaClasse{
2
3     public MiaClasse(){
4         System.out.println("##Costruttore");
5     }
6
7     public int getValore(){
8         return(10);
9     }
10
11     public static int getValoreStatic(){
12         return(0);
13     }
14
15     public static void main(String args[]){
16         System.out.println("#Main 1 \n");
17         System.out.println("#"+MiaClasse.getValoreStatic());
18         MiaClasse miaClass = new MiaClasse();
19         System.out.println("#"+miaClass.getValore());
20         System.out.println("#Main \n");
21     }
22 }
```

```
barbon@barbon-PC:~/Scaricati/FI/codici$ java M
#Main 1
#0
##Costruttore
#10
#Main
```

2) Classi, variabili e oggetti

```
1 public class Cane{
2
3     String nome;
4     int eta;
5
6     public Cane(){
7
8     }
9
10    public Cane(String n){
11        nome = n;
12        System.out.println("Il cane "+nome);
13    }
14
15    public Cane(String n, int e){
16        nome = n;
17        eta = e;
18        System.out.println("Il cane "+nome+" ha "+e);
19    }
20
21 }
```

```
1 public class Main{
2     public static void main(String args[]){
3         Cane cane1 = new Cane();
4         Cane cane2 = new Cane("Chaise");
5         Cane cane3 = new Cane("Sky", 4);
6     }
7 }
```

```
barbon@barbon-PC:~/Scaricati/FI/codici$ javac Cane.java
barbon@barbon-PC:~/Scaricati/FI/codici$ javac Main.java
barbon@barbon-PC:~/Scaricati/FI/codici$ java Main
Il cane Chaise
Il cane Sky ha 4
barbon@barbon-PC:~/Scaricati/FI/codici$
```

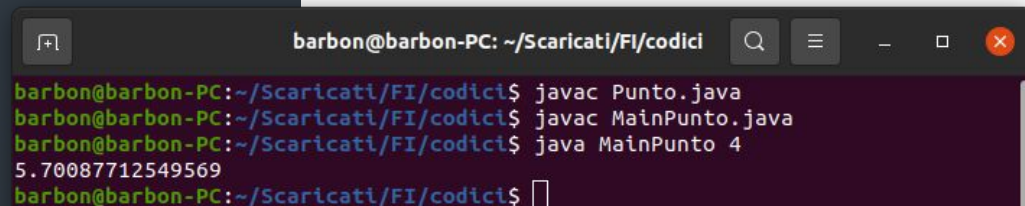
3) Visibilità

- L'identificatore della classe e quello dei membri della classe sono sempre visibili all'interno della classe;
- Un membro di una classe può avere uno dei modificatori:
 - **modificatore public** : membro visibile alle altre classi;
 - **modificatore private**: membro non visibile alle altre classi;

3) Visibilità

```
1 public class Punto{
2     private double x, y;
3
4     // costruttore
5     public Punto (double a, double b ){
6         x = a;
7         y = b;
8     }
9
10    // metodo: calcola la distanza dall'origine
11    double distanza(){
12        return Math.sqrt(x*x + y*y);
13    }
14
15    // metodo: trasla le coordinate del punto
16    // della quantità t
17    void traslazione(double t){
18        x += t;
19        y += t;
20    }
21 }
```

```
1 public class MainPunto{
2     public static void main(String[] args){
3         double d = Double.parseDouble(args[0]);
4         Punto p1 = new Punto(1, 2); // azione 1)
5         d = p1.distanza(); // azione 2)
6         Punto p2 = p1; // azione 3)
7         p2.traslazione(2.5);
8         System.out.println(p2.distanza());
9     }
10 }
```

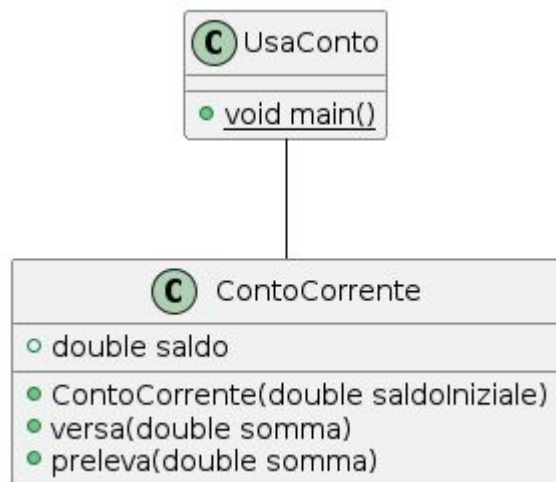


```
barbon@barbon-PC: ~/Scaricati/FI/codici
barbon@barbon-PC:~/Scaricati/FI/codici$ javac Punto.java
barbon@barbon-PC:~/Scaricati/FI/codici$ javac MainPunto.java
barbon@barbon-PC:~/Scaricati/FI/codici$ java MainPunto 4
5.70087712549569
barbon@barbon-PC:~/Scaricati/FI/codici$
```

4) Distruttori

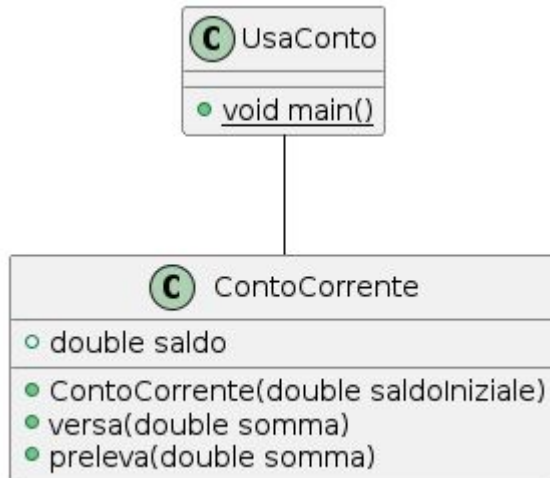
- In Java **non esistono distruttori** (come per esempio in C++), per cui non avviene (come per la creazione) una distruzione esplicita degli oggetti ad opera del programmatore;
- Il recupero della **memoria dinamica** non più utilizzata viene effettuata dalla piattaforma Java per mezzo di una specifica routine (**Garbage Collector**)

5) Esempi

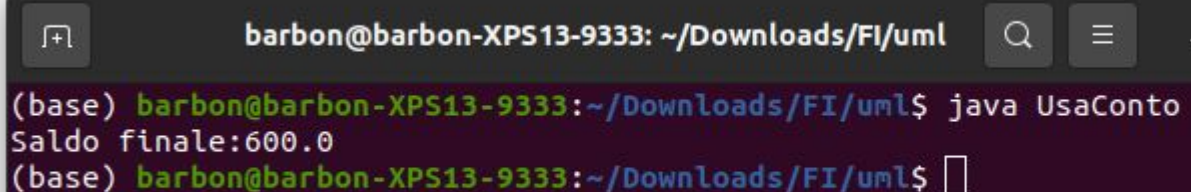


```
1 public class ContoCorrente {
2
3     public double saldo ;
4
5     public ContoCorrente ( double saldoIniziale ) {
6         saldo = saldoIniziale ;
7     }
8
9     public void versa ( double somma ) {
10        saldo += somma ;
11    }
12
13    public void preleva ( double somma ) {
14        saldo -= somma ;
15    }
16 }
```

5) Esempi



```
1 public class UsaConto {
2     public static void main ( String [] args ) {
3         ContoCorrente cc = new ContoCorrente (1000);
4
5         cc.versa(700);
6
7         if (cc.saldo >200)
8             cc.preleva(200);
9         if (cc.saldo >900)
10            cc.preleva(900);
11        System.out.println("Saldo finale:"+cc.saldo);
12    }
13 }
```



```
barbon@barbon-XPS13-9333: ~/Downloads/FI/uml
(base) barbon@barbon-XPS13-9333:~/Downloads/FI/uml$ java UsaConto
Saldo finale:600.0
(base) barbon@barbon-XPS13-9333:~/Downloads/FI/uml$
```

Grazie!

