



UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Università degli Studi di Salerno

Dipartimento di Informatica

Programmazione ad Oggetti

a.a. 2023-2024

Modificatore Static

Docente: Massimo Ficco

E-mail: *mficco@unisa.it*

IL MODIFICATORE STATIC

Il **modificatore** può essere definito: **public**, **private**, **static**, (public per default)

- Campi e metodi dichiarati *static*, **sono associati alla classe e non a una particolare istanza**

```
class MyClass {  
    static int a;  
    ...  
    static void MyMethod() {  
        a = a+1;  
    }  
}
```

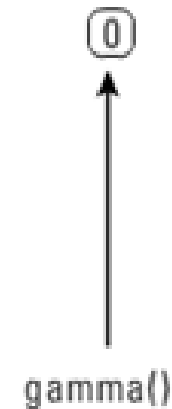
- Pertanto: esiste una sola copia di un campo statico, condiviso da tutte le istanze della classe; non occorre istanziare un oggetto per usare un membro statico; metodi statici possono accedere solo a membri statici della classe
- Sono qualificati con il nome della classe, e non della istanza

```
MyClass.a = MyClass.a + 1;
```



II MODIFICATORE STATIC

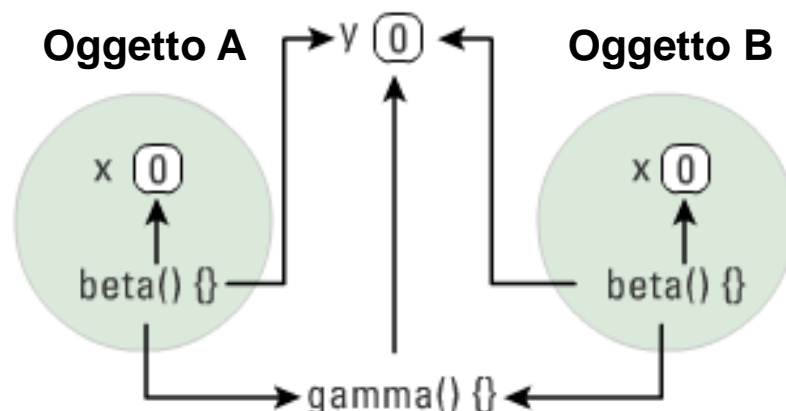
```
public class Alpha {  
    int x = 0;  
    static int y = 0;  
    public void beta () {  
        //...modifica x  
    }  
    public static void gamma () {  
        //... modifica y  
    }  
}
```



II MODIFICATORE STATIC

```
public class Alpha {  
    int x = 0;  
    static int y = 0;  
    public void beta () {  
        //...modifica x  
    }  
    public static void gamma () {  
        //... modifica y  
    }  
}
```

*Alpha A = new Alpha();
Alpha B = new Alpha();*



Applicazione con metodi e attributi static

```
public class AlphaTest {  
    public static void main ( String args [ ] )  
    {  
        Alpha.y = 7;  
        Alpha.gamma ( );  
        Alpha a = new Alpha ( );  
        a.x=5;  
        a.beta ( );  
    }  
}  
  
public class Alpha {  
    int x = 0;  
    static int y = 0;  
    public void beta ( ) {  
        //...modifica x  
    }  
    public static void gamma ( ) {  
        //...modifica y  
    }  
}
```



Static

Abbiamo definito la classe auto, ma non abbiamo nessun oggetto fino a quando non lo creiamo con un **new**.

A questo punto possiamo utilizzarlo ed invocarne i metodi.

Ci sono due situazioni in cui questo non basta:

- Vogliamo definire un area di memoria (un attributo comune a tutti gli oggetti di una stessa classe)
- Vogliamo richiamare un metodo o un attributo senza far riferimento ad un particolare oggetto



Static

Definire un metodo o un attributo static significa dire che questo appartiene alla classe non è legato ad alcun oggetto in particolare

Un metodo static non può accedere a elementi non-static perché questi non vengono creati fino a che non viene creato un oggetto

Un attributo o un metodo statico può essere chiamato sia in riferimento ad un oggetto sia in riferimento alla classe



Esempio

```
class StaticTest {  
    static int i = 47;  
    int j = 0;  
    static void incr(){i++;}  
}  
  
public class Test{  
    public static void main(String args[]){ //il main è statico  
        System.out.println(StaticTest.i);  
        StaticTest st1 = new StaticTest();  
        StaticTest st2 = new StaticTest();  
        st1.j++; st2.j++;  
        st1.i++; st2.i++; // StaticTest.i++, oppure StaticTest.incr(), oppure st1.incr()  
        System.out.println(st1.i);  
        System.out.println(st2.i);  
    }  
}
```



Attributi statici e non statici

Un metodi statico può accedere solo agli attributi statici:

```
class StaticTest {  
    static int i = 47;  
    Int j;  
    static void incr(){i++;}  
}  
public class Test{  
    public static void main(String args[]){ //il main è statico
```

//Quale tra queste operazione non consentita?

```
    System.out.println(StaticTest.j);  
    StaticTest.incr();  
    System.out.println(StaticTest.i);  
    StaticTest st=new StaticTest();  
    System.out.println(st.i);  
    System.out.println(st.j);  
    st.incr();  
}  
}
```



Attributi statici e non statici

Un metodo statico può accedere solo agli attributi statici:

```
class StaticTest {  
    static int i = 47;  
    int j;  
    static void incr(){i++;}  
}  
public class Test{  
    public static void main(String args[]){ //il main è statico  
  
        System.out.println(StaticTest.j); //operazione non consentita  
        StaticTest.incr(); //operazione consentita  
        System.out.println(StaticTest.i); //operazione consentita  
        StaticTest st=new StaticTest();  
        System.out.println(st.i); //operazione consentita  
        System.out.println(st.j); //operazione consentita  
        st.incr(); //operazione consentita  
    }  
}
```



Attributi statici e non statici

Un metodo statico può accedere solo agli attributi statici:

```
public class Test{
    static int i = 47;
    int j;
    static void incr(){i++;}

    public static void main(String args[]){ //il main è statico

        System.out.println(j);
        System.out.println(i);  i++;
        incr();
        Test t=new Test();
        System.out.println(t.i);
        System.out.println(t.j);

    }
}
```



Attributi statici e non statici

Un metodo statico può accedere solo agli attributi statici:

```
public class Test{  
    static int i = 47;  
    int j;  
    static void incr(){i++;}
```

```
public static void main(String args[]){ //il main è statico
```

```
    System.out.println(j); //operazione non consentita
```

```
    System.out.println(i); i++; //operazione consentita
```

```
    incr(); //operazione consentita
```

```
    Test t=new Test();
```

```
    System.out.println(t.i); //operazione consentita
```

```
    System.out.println(t.j); //operazione consentita
```

```
}
```

```
}
```

Perché non uso il punto ??????????????????????



Classi Interne

- ▶ Le classi interne non possono dichiarare metodi statici. Ma è possibile che una classe interna sia static.
- ▶ Un'istanza di una classe interna implicitamente mantiene un riferimento all'oggetto della classe esterna che lo ha creato. Questo non è vero per classi interne statiche:
 - ▶ Non si ha bisogno di un oggetto della classe esterna per creare un oggetto della classe interna;
 - ▶ Non è possibile accedere ad un oggetto della classe esterna da un oggetto della classe interna static;
 - ▶ Può contenere attributi statici.
- ▶ Non è consentito inserire codice all'interno di una interfaccia, ma una classe interna statica può essere parte di una interfaccia.

