**Classi e Oggetti Java -** [**https://www.w3schools.com/java/java\_classes.asp**](https://www.w3schools.com/java/java_classes.asp)

Tutto in Java è associato a classi e oggetti, insieme ai suoi attributi e metodi. Ad esempio: nella vita reale, un'auto è un oggetto. La macchina ha **attributi** , come il peso e colore, e **metodi** , quali motrice, freni.

Una classe è come un costruttore di oggetti o un "progetto" per la creazione di oggetti.

**Creazione una classe**:

Per ***creare*** una classe, utilizzare la parola chiave ***class***:

public class MyClass {  
  int x = 5;  
}

**Creare un oggetto**:

Per creare un oggetto di MyClass, specificare il nome della classe, seguito dal nome dell'oggetto e utilizzare la parola chiave ***new***:

public class MyClass {  
  int x = 5;  
  
  public static void main(String[] args) {  
    MyClass **myObj** = new MyClass();  
    System.out.println(myObj.x);  
  }  
}

Puoi creare più oggetti di una classe:

public class MyClass {  
  int x = 5;  
  
  public static void main(String[] args) {  
    MyClass **myObj1** = new MyClass();  // Object 1  
    MyClass **myObj2** = new MyClass();  // Object 2  
    System.out.println(myObj1.x);  
    System.out.println(myObj2.x);  
  }  
}

Puoi anche creare un oggetto di una classe e accedervi in ​​un'altra classe. Questo è spesso usato per una migliore organizzazione delle classi (una classe ha tutti gli attributi e i metodi, mentre l'altra classe contiene il **main**()).

class OtherClass {  
  public static void main(String[] args) {  
    MyClass **myObj** = new MyClass();  
    System.out.println(myObj.x);  
  }  
}

**Attributi della Classe:** <https://www.w3schools.com/java/java_class_attributes.asp>

Classe chiamata " MyClass" con due attributi: x e y:

public class MyClass {  
  int x = 5;  
  int y = 3;  
}

Un altro termine per attributi di classe è **campi** .

**Accesso agli attributi**:

Puoi accedere agli attributi creando un oggetto della classe e usando la sintassi **punto** ( **.**):

public class MyClass {  
  int x = 5;  
  
  public static void main(String[] args) {  
    MyClass myObj = new MyClass();  
    System.out.println(myObj.x);  
  }  
}

**Modifica degli Attributi**:

Impostare il valore su x su 40:

public class MyClass {  
  int x;  
  
  public static void main(String[] args) {  
    MyClass myObj = new MyClass();  
    myObj.x = 40;  
    System.out.println(myObj.x);  
  }  
}

Se non si desidera la possibilità di sovrascrivere i valori esistenti, dichiarare l'attributo come ***final***:

public class MyClass {  
  **final** int x = 10;  
  
  public static void main(String[] args) {  
    MyClass myObj = new MyClass();  
    myObj.x = 25; // will generate an error: cannot assign a value to a**final** variable  
    System.out.println(myObj.x);   
  }  
}

**Oggetti Multipli**:

Se si creano più oggetti di una classe, è possibile modificare i valori degli attributi in un oggetto, senza influire sui valori degli attributi nell'altro.

Modificare il valore di ***x*** 25 a ***myObj2***, e lasciare ***x*** in ***myObj1*** invariata:

public class MyClass {  
  int x = 5;  
  
  public static void main(String[] args) {  
    MyClass myObj1 = new MyClass();  // Object 1  
    MyClass myObj2 = new MyClass();  // Object 2  
    myObj2.x = 25;  
    System.out.println(myObj1.x);  // Outputs 5  
    System.out.println(myObj2.x);  // Outputs 25  
  }  
}

**Metodi della Classe**: <https://www.w3schools.com/java/java_class_methods.asp>

Crea un metodo denominato ***myMethod()*** in ***MyClass***:

public class MyClass {  
  static void **myMethod()** {  
    System.out.println("Hello World!");  
  }  
}

Per chiamare un metodo, scrivi il nome del metodo seguito da due parentesi () e un punto e virgola ;

public class MyClass {  
  static void myMethod() {  
    System.out.println("Hello World!");  
  }  
  
  public static void main(String[] args) {  
    **myMethod();**  
  }  
}  
// Outputs "Hello World!"

**Static o Public**:

Nell'esempio sopra, abbiamo creato un ***static*** metodo, il che significa che è possibile accedervi senza creare un oggetto della classe, diversamente ***public***, a cui possono accedere solo gli oggetti:

public class MyClass {  
  // Static method  
  static void myStaticMethod() {  
    System.out.println("Static methods can be called without creating objects");  
  }  
  
  // Public method  
  public void myPublicMethod() {  
    System.out.println("Public methods must be called by creating objects");  
  }  
  
  // Main method  
  public static void main(String[] args) {  
    myStaticMethod(); // Call the static method  
    // myPublicMethod(); This would compile an error  
  
    MyClass myObj = new MyClass(); // Create an object of MyClass  
    myObj.myPublicMethod(); // Call the public method on the object  
  }  
}

**Costruttore Java**: <https://www.w3schools.com/java/java_constructors.asp>

Un costruttore in Java è un **metodo speciale** utilizzato per inizializzare gli oggetti. Il costruttore viene chiamato quando viene creato un oggetto di una classe. Può essere utilizzato per impostare i valori iniziali per gli attributi dell'oggetto:

// Create a MyClass class  
public class MyClass {  
  int x;  // Create a class attribute  
  
  // Create a **class constructor** for the MyClass class  
  public MyClass() {  
    x = 5;  // Set the initial value for the class attribute x  
  }  
  
  public static void main(String[] args) {  
    MyClass myObj = new MyClass(); // Create an object of class MyClass (This will **call the constructor**)  
    System.out.println(myObj.x); // Print the value of x  
  }  
}  
  
// Outputs 5

Si noti che il nome del costruttore deve **corrispondere al nome della classe** e non può avere un **tipo restituito** (come void) e viene chiamato quando viene creato l'oggetto.

Tutte le classi hanno costruttori di default: se non crei tu stesso un costruttore di classi, Java ne crea uno per te. Tuttavia, non è possibile impostare i valori iniziali per gli attributi dell'oggetto.

**Parametri del costruttore**:

I costruttori possono anche prendere parametri, che viene utilizzato per inizializzare gli attributi.

public class MyClass {  
  int x;  
  
  public MyClass(int y) {  
    x = y;  
  }  
  
  public static void main(String[] args) {  
    MyClass myObj = new MyClass(5);  
    System.out.println(myObj.x);  
  }  
}  
  
// Outputs 5