


I costruttori delle classi wrapper permettono di creare oggetti che contengono i valori primitivi corrispondenti

Integer x = new Integer (3); oppure Integer x = 3; = 

Double z = new Double (2.1); oppure Double z = 2.1; = 

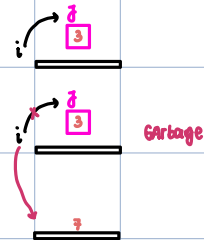
!! non si converte il datatype

ma **AUTOBOXING**: è una sintassi semplificata per nascondere la creazione degli oggetti wrapper

int j = 3;

Integer i = j; **AUTOBOXING** → Integer i = new Integer (j);

i = 7; **AUTOBOXING** → i = new Integer (7);



Come recupero il valore primitivo corrispondente?

Integer x = new Integer (7);
↓ **UNBOXING**
x.intValue(); // 7 di tipo int

posso anche fare:

Integer x = new Integer (7);

int y = x.intValue() + 2; // 9

Il tutto è automatico : Integer x = new Integer (7);

int y = x + 2; // 9

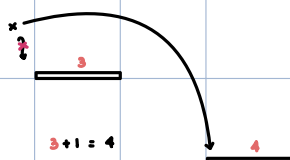
Immutabilità:

Non esiste un modo per modificare il valore di un oggetto wrapper → **immutabili**

Es:

Integer x = 3;

x = x + 1;



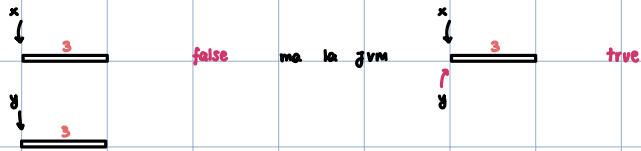
```
System.out.println(x);
```

Confronto:

```
Integer x = 3;
```

```
Integer y = 3;
```

```
if (x == y) {
```



Ricorda se voglio confrontare il contenuto uso il metodo `x.equals(y)`

metodo `toString`

Se `w` è un wrapper

```
w.toString();
```

ritorna la stringa corrispondente al valore primitivo contenuto in `w`

```
Integer x = 3;
```

```
if (x.toString().equals("3")) ...
```

```
System.out.println(x.toString());
```

anche il metodo `toString` è presente in tutte le classi e in alcuni situazioni è invocato automaticamente

metodo `parseInt`

Se `s` è un oggetto di tipo `String`

```
Integer.parseInt(s);
```

ritorna l'intero corrispondente rappresentato alla costante intera rappresentata da `s`, altrimenti viene lanciata un'eccezione

```
int j = Integer.parseInt("13") + 3; // j = 16
```

```
int i = Integer.parseInt(" 13"); // eccezione
```

```
int k = Integer.parseInt("2.1"); // eccezione
```