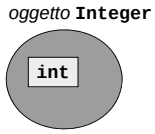
TIPI PRIMITIVI E CLASSI "WRAPPER"

- **In varie situazioni, può essere comodo poter *trattare i tipi primitivi come oggetti***
  - **per passarli per riferimento a una funzione**
  - **quando una funzione pretende come parametro (o restituisce) un `Object`** (vedi Collection, List, …)
- **Una *classe "wrapper"* incapsula una variabile di un tipo primitivo**
  - **la classe `Boolean` incapsula un `boolean`**
  - **la classe `Double` incapsula un `double`**
  - **...**
- La classe wrapper ha nome (quasi) identico al tipo primitivo che incapsula, ma con l'iniziale maiuscola.

TIPI PRIMITIVI E CLASSI "WRAPPER"

Tipo primitivo	Classe "wrapper" corrispondente
`boolean`	`Boolean`
`char`	`Character`
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`



Ogni classe wrapper definisce metodi per estrarre il valore della variabile incapsulata e viceversa.

TIPI PRIMITIVI E CLASSI "WRAPPER"

- **Per estrarre il valore incapsulato:**
  - `Integer` fornisce il metodo `intValue()`
  - `Double` fornisce il metodo `doubleValue()`
  - `Boolean` fornisce il metodo `booleanValue()`
  - `Character` fornisce il metodo `charValue()`
  - `...`
- **Per creare un oggetto da un valore primitivo:**
  - `Integer i = new Integer(`*valore int*`)`
  - `Double d = new Double(`*valore double*`)`
  - `Boolean b = new Boolean(`*valore boolean*`)`
  - `Character c = new Character(`*valore char*`)`
  - `...`

CLASSI WRAPPER - ESEMPIO

```
public class EsempioWrapper {
 public static void main(String args[]){
  int x = 35;
  Integer ix = new Integer(x);
  x = 2*ix.intValue();

  System.out.println("ix =" + ix);
  System.out.println("x =" + x);
 }
}
```

Costruisce un oggetto **`Integer`** a partire da un valore **`int`**

Estrae da un oggetto **`Integer`** il valore **`int`** incapsulato al suo interno, mediante **`intValue()`**

```
ix = 35
x = 70
```

TIPI PRIMITIVI E CLASSI "WRAPPER"

- **La classe wrapper contiene anche *funzioni che operano sul tipo primitivo corrispondente***
  - **in particolare, conversione stringa / numero ...**
  - **… e viceversa**
- **Attenzione: le funzioni che operano sul tipo primitivo sono *funzioni statiche, di classe***
  - infatti, non esiste alcun oggetto su cui invocare metodi
- **Quindi, varie funzionalità sono disponibili in *doppia versione:***
  - funzione statica (per variabili di tipo primitivo)
  - metodi (per oggetti che siano istanze del tipo wrapper)

TIPI PRIMITIVI E CLASSI "WRAPPER"

Esempio: *`toString()` nella classe `Integer`*

- **versione statica:**

```
public static String toString(int x);
```

  - prende un valore **`int`** e ne produce la rappresentazione sotto forma di stringa

- **versione metodo:**

```
public String toString();
```

  - è implicitamente invocato su un oggetto Integer
  - ne recupera il valore e ne produce la rappresentazione sotto forma di stringa.

ESEMPIO

```
public class EsempioWrapper {
 public static void main(String args[]){
  int x = 35;
  Integer ix = new Integ
  x = 2 * ix.intValue();

  System.out.println("ix =" + ix);
  System.out.println("x =" + x);
 }
}
```

Conversione implicita da **`Integer`** a **`String`** (usan-do il metodo **`toString()`** di **`Integer`**)

```
ix = 35
x = 70
```

Conversione implicita da **`int`** a **`String`** (usando la funzione statica **`toString()`** di **`Integer`**)