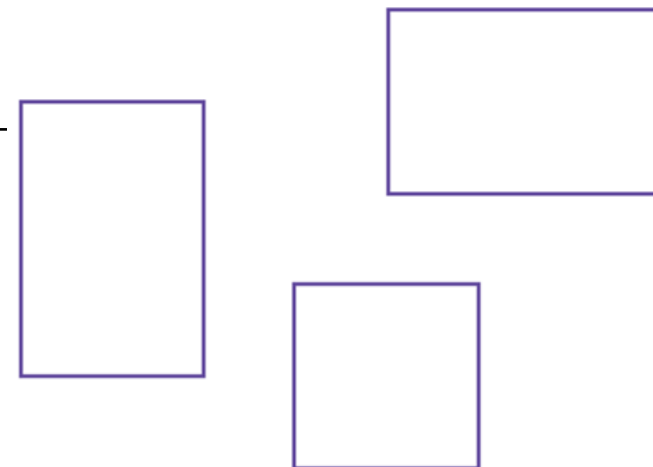


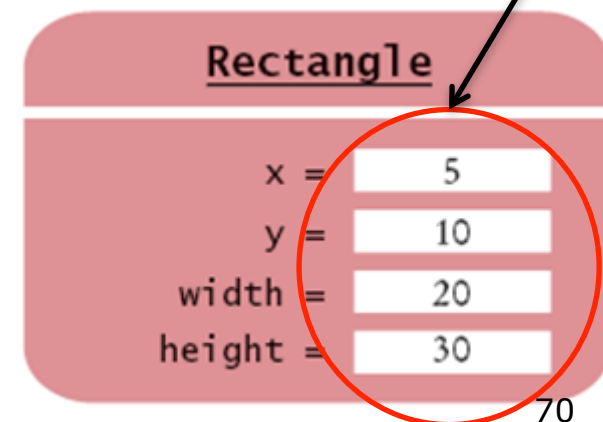
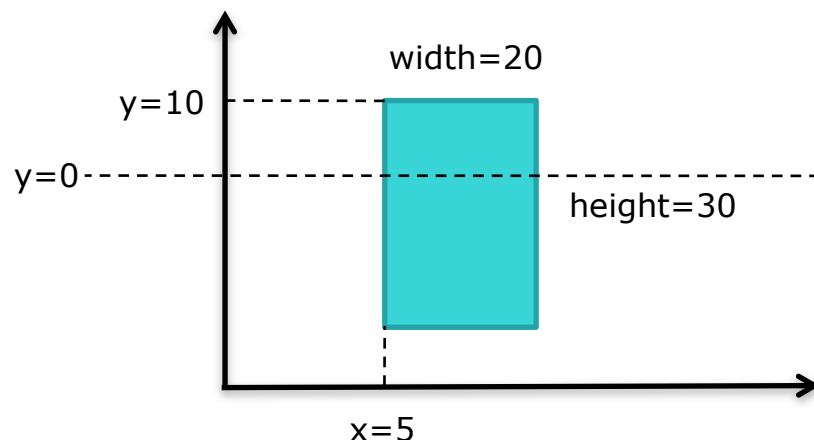
# Classe Rectangle

- Oggetti di tipo rettangolo:



- Per descrivere un rettangolo posso specificare

- L'ascissa x e l'ordinata y del suo angolo superiore sinistro
- il valore della larghezza (width)
- il valore dell'altezza (height)





## Classe Rectangle cont.

---

- Operazioni possibili:
  - traslazione del punto iniziale: `translate(x,y)`
  - recupero valore altezza: `getHeight()`
  - modifica ampiezza e altezza: `resize(w,h)`
  - intersezione con altro rettangolo:  
`intersection(R)`
  - test intersezione non vuota: `intersects(R)`
  - test uguaglianza: `equals(O)`
  - etc.

# Forme rettangolari ed Oggetti Rectangle

- Un oggetto **Rectangle** non è una forma rettangolare – ma un oggetto che contiene un insieme di numeri che descrivono il rettangolo

<u>Rectangle</u>	<u>Rectangle</u>	<u>Rectangle</u>
x = <input type="text" value="5"/>	x = <input type="text" value="35"/>	x = <input type="text" value="45"/>
y = <input type="text" value="10"/>	y = <input type="text" value="30"/>	y = <input type="text" value="0"/>
width = <input type="text" value="20"/>	width = <input type="text" value="20"/>	width = <input type="text" value="30"/>
height = <input type="text" value="30"/>	height = <input type="text" value="20"/>	height = <input type="text" value="20"/>



# Il metodo costruttore

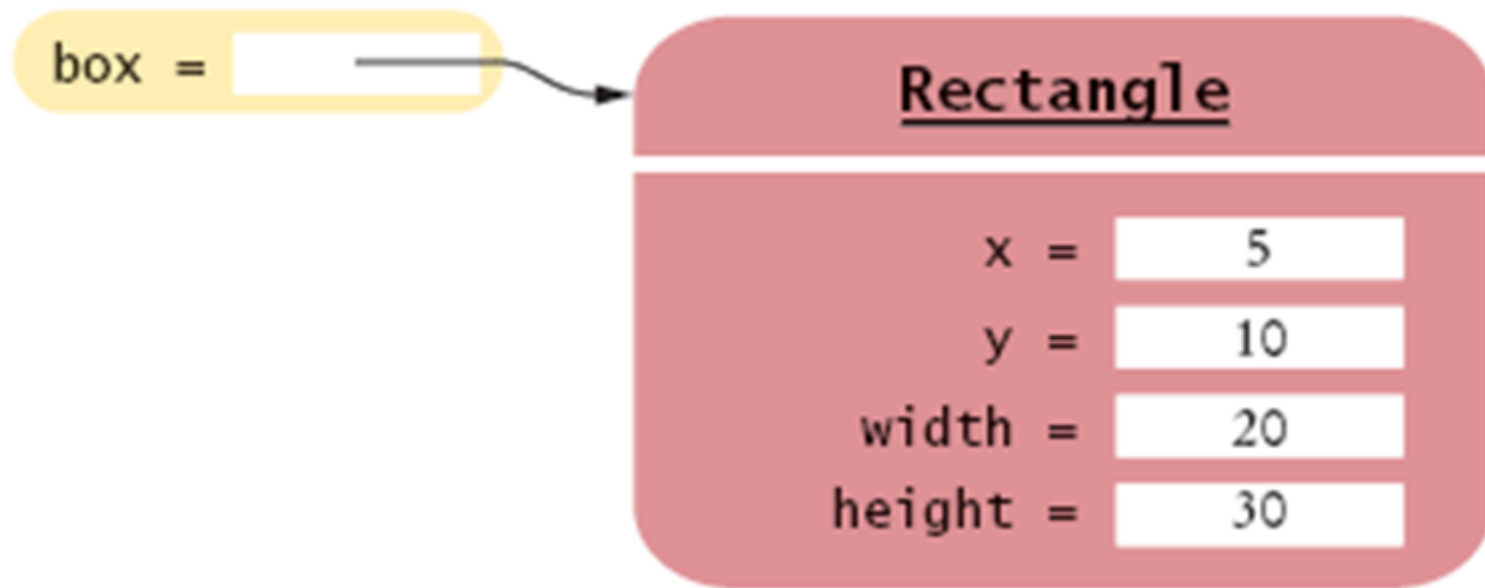
---

```
public Rectangle(int x_init, int y_init,  
                int width_init, int height_init) {  
  
    x=x_init;  
    y=y_init;  
    width=width_init;  
    height=height_init;  
}
```

- Una **classe** può implementare un **metodo** particolare, detto **costruttore**, che serve a inizializzare i nuovi oggetti
- Quando esiste un **costruttore** deve chiamarsi come la **classe**
- Per creare un oggetto di una classe deve essere invocato un costruttore della classe in combinazione con l'operatore **new**
- Se una variabile di istanza non è inizializzata dal costruttore allora è inizializzata automaticamente a **0** se si tratta di un tipo numerico o a **null** se si tratta di un riferimento

# Variabili che contengono oggetti

---



- La variabile **box** contiene un riferimento (indirizzo) ad un oggetto di tipo rettangolo.

# Creazione di oggetti (1)

`Rectangle box;`

- Definisce una variabile oggetto rettangolo non inizializzata

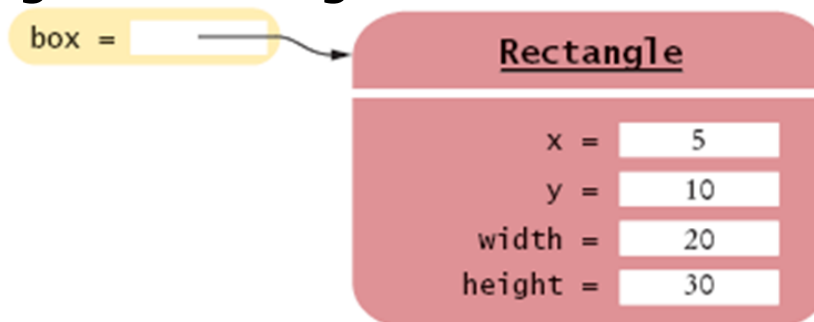
`box =`

- E' buona norma inizializzare sempre le variabili oggetto

`box = new Rectangle(5, 10, 20, 30);`

- Crea un rettangolo e assegna il suo indirizzo a `box`

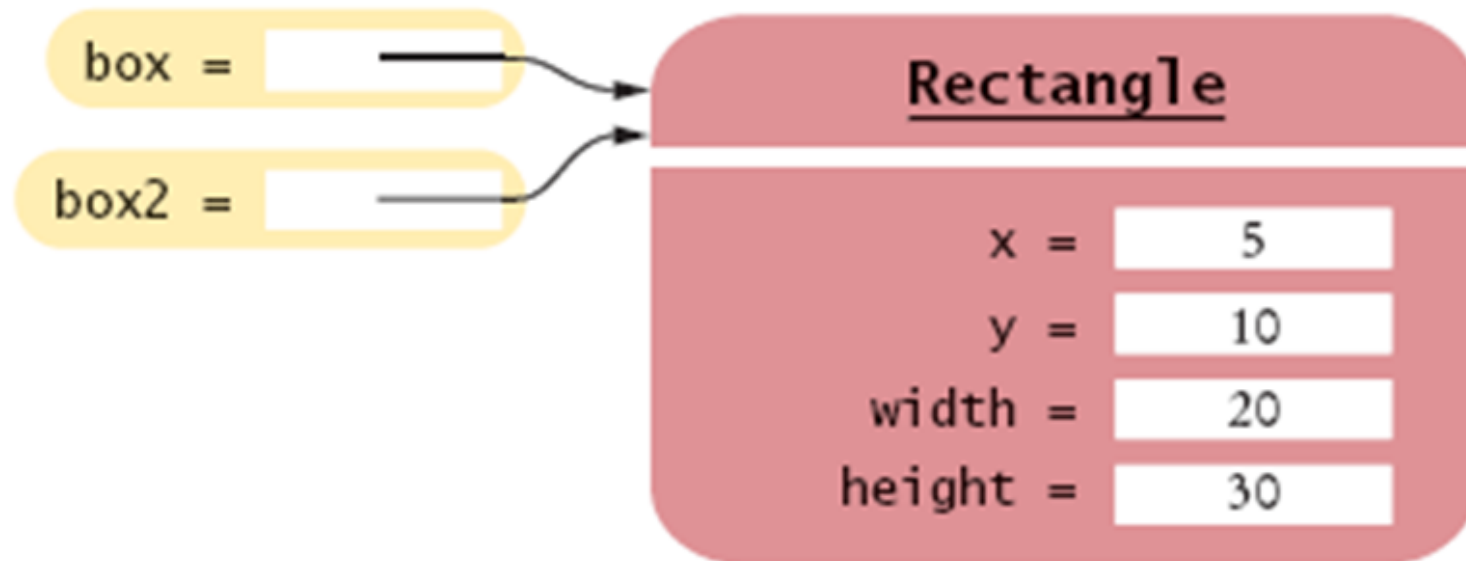
`box =`



## Creazione di oggetti (2)

```
Rectangle box = new Rectangle(5, 10, 20, 30);  
Rectangle box2 = box; // (assegnamento a variabile)
```

- Si ottengono due variabili oggetto che si riferiscono allo stesso oggetto





## Creazione di oggetti (3)

---

- Quando viene creato l'**oggetto** box con  
`Rectangle box = new Rectangle(5, 10, 20, 30);`  
viene allocato uno spazio di memoria in cui sono conservati
  - i valori di `x`, `y`, `width` e `height`
    - quindi ciascuna istanza di `Rectangle` ha le proprie copie delle variabili `x`, `y`, `width` e `height`
  - gli indirizzi dei metodi `Rectangle`, `translate`, etc.
    - quindi i metodi di tutte le istanze di `Rectangle` sono implementati con lo stesso codice





## Creazione di oggetti (4)

---

- `Rectangle box = new Rectangle(5, 10, 20, 30);`
- `Rectangle r = new Rectangle(5, 10, 20, 30);`
- Si definiscono due variabili inizializzate con due oggetti distinti di tipo **Rectangle**
- **r** e **box** si riferiscono a due oggetti che sono indistinguibili rispetto allo stato (stesso stato) e al comportamento, ma hanno identità differenti



## Self Check

---

- Come costruisci un quadrato con centro (100, 100) e lunghezza del lato 20?
- Cosa stampa la seguente istruzione?

```
System.out.println(new Rectangle().getWidth());
```



# Risposte

---

○

```
new Rectangle(90, 90, 20, 20)
```

○

0



## Self Check - Identificare gli errori

---


- `Rectangle r = (5, 10, 15, 20);`
- `double width = Rectangle(5, 10, 15, 20).getWidth();`
- `Rectangle r;`  
`r.translate(15,25);`
- `r = new Rectangle();`  
`r.translate("far, far away!");`

# Copiare Numeri

```
int luckyNumber = 13;  
int luckyNumber2 = luckyNumber;  
luckyNumber2 = 12;
```

1 luckyNumber = 13

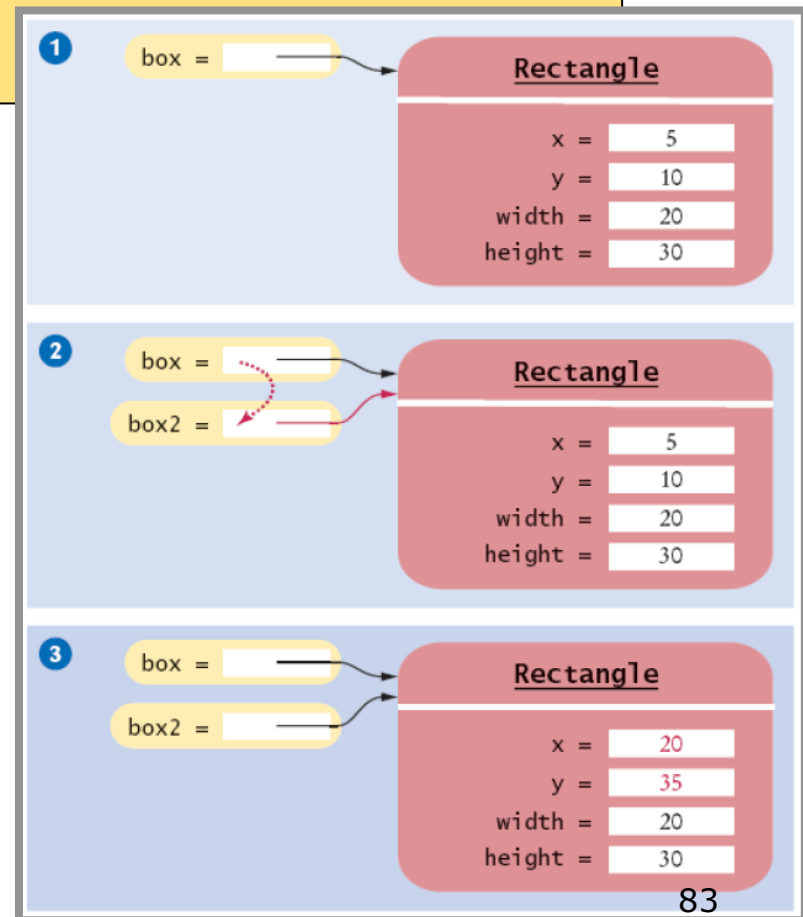
2 luckyNumber = 13  
luckyNumber2 = 13



3 luckyNumber = 13  
luckyNumber2 = 12

# Copiare Riferimenti ad Oggetti

```
Rectangle box = new Rectangle(5, 10, 20, 30);  
Rectangle box2 = box;  
box2.translate(15, 25);
```





## Self Check

---

- Qual' è l' effetto dell' assegnamento `greeting2 = greeting`?
- Dopo l' invocazione di `greeting2.toUpperCase()`, quali sono i contenuti di `greeting` e `greeting2`?

# Classificazione metodi

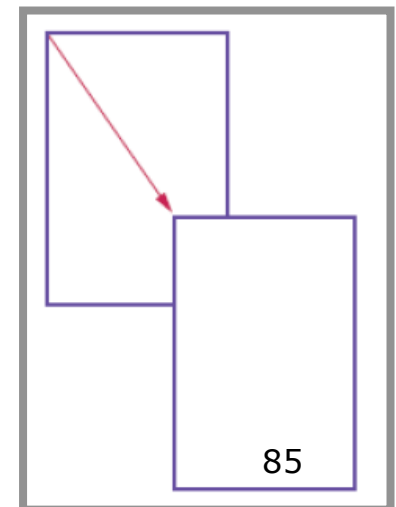
---

- **Metodi di accesso:** non cambiano lo stato del suo parametro implicito

```
double width = box.getWidth();
```

- **Metodi modificatori:** cambiano lo stato del suo parametro implicito

```
box.translate(15, 25);
```







## Self Check

---

- Il metodo `toUpperCase` della classe `String` è un metodo di accesso o un modificatore?
- Quale chiamata a `translate` è necessaria per muovere il rettangolo `box` così che il punto in alto a sinistra è l'origine (0, 0)?

```
Rectangle box = new Rectangle(5, 10, 20, 30);  
box.translate(-5, -10)
```



# Garbage Collection

---

- Quando un oggetto non ha più un riferimento non può più essere referenziato da alcun programma
- L'oggetto diventa inaccessibile e quindi inutile
  - si chiama *garbage* (spazzatura)
- Java effettua una raccolta automatica e periodica degli oggetti inutili (*garbage collection*)
  - per rendere nuovamente utilizzabile per usi futuri lo spazio di memoria che l'oggetto occupava
  - Per ridurre lo spazio occupato dallo *heap*
- In altri linguaggi è responsabilità del programmatore effettuare il rilascio della memoria occupata da oggetti non referenziati e quindi inaccessibili



# Implementare un programma test

---

- Scrivi una nuova classe con il metodo **main**
- All'interno del metodo **main** costruisci uno o più oggetti
- Applica i metodi agli oggetti
- Visualizza i risultati delle chiamate ai metodi



# Importazione pacchetti

---

- Per usare le classi delle librerie o riutilizzare codice proprio può essere necessario importare delle classi:
  - le classi Java sono raggruppate in pacchetti
  - le classi di libreria si importano specificandone il pacchetto e il nome della classe

```
import java.awt.Rectangle;
```

- Non è necessario importare le classi del pacchetto `java.lang` come `String` e `System`



# File MoveTester.java

```
01: import java.awt.Rectangle;
02:
03: public class MoveTester
04: {
05:     public static void main(String[] args)
06:     {
07:         Rectangle box = new Rectangle(5, 10, 20, 30);
08:
09:         // Move the rectangle
10:         box.translate(15, 25);
11:
12:         // Print information about the moved rectangle
13:         System.out.print("x: ");
14:         System.out.println(box.getX());
15:         System.out.println("Expected: 20");
16:
17:         System.out.print("y: ");
18:         System.out.println(box.getY());
19:         System.out.println("Expected: 35");
20:     }
21: }
```