ACCOUNT IN LABORATORI

- compilare e stampare il modulo online
  - http://www.di.unisa.it/dipartimento/strutture?id=37
  - il modulo dovrà essere consegnato dal lunedì al venerdì, dalle ore 10:00 alle ore 12:00 al Sig. Fulvio Marino - staff tecnico del Dipartimento di Informatica: stecca 7, II piano, stanza n. 13;
  - il modulo dovrà essere firmato all'atto della consegna esclusivamente dal richiedente, munito di un documento di identità in corso di validità.

Ancora sulle stringhe

- Metodi della classe String

method	returns	arguments
toUpperCase	ref. String object	none
toLowerCase	ref. String object	none
length	a number	none
trim	ref. String object	none
concat	ref. String object	ref. String object
substring	ref. String object	number
substring	ref. String object	two numbers

Posizioni nelle stringhe

- Le posizioni dei caratteri in una stringa sono numerate a partire da 0

H	a	m	b	u	r	g	e	r
0	1	2	3	4	5	6	7	8

- Stringhe e sottostringhe

```
String big = “hamburger”;
String small = big.substring(3,7);
String medium = big.substring(3);
String bigInCaps = big.toUpperCase();
String order = big.concat(“ with onions”);
```

Esempio

```
import java.io.*;
class Esempio {
        public static void main(String arg[]) {
                String          first = "John";
                String          middle = "Fitzgerald";
                String          last = "Kennedy";
                String          initials;
                String          firstInit, middleInit, lastInit;
                firstInit = first.substring(0,1);
                middleInit = middle.substring(0,1);
                lastInit = last.substring(0,1);
                initials = firstInit.concat(middleInit);
                initials = initials.concat(lastInit);
                System.out.println(initials);
        }
}
```

Proprietà delle stringhe

- **Immutabilità**: una volta creato un oggetto String NON può cambiare
  - Es: l'invio di un messaggio `toUpperCase` comporta la creazione di un nuovo oggetto `String`
- Stringa vuota
  - Lunghezza 0
  - Nessun carattere
  - Reference: “”

Meccanismi

- Dato

```
String w, x, y, z, s;
w = “ab”;
x = “cd”;
y = “ef”;
z = “gh”;
```

- Assegnare ad **s** la concatenazione delle stringhe referenziate da **w**, **x**, **y**, **z**
  - **“abcdefgh”**

Cascata di messaggi

- `s=w.concat(x).concat(y).concat(z);`
- Il messaggio `concat(x)` è inviato a `w`
  - L'espressione `w.concat(x)` si riferisce alla stringa risultante
  - `w.concat(x)` → abcd
- Il messaggio `concat(y)` è inviato alla nuova stringa "abcd"
  - L'espressione `w.concat(x).concat(y)` si riferisce alla stringa risultante
  - `w.concat(x).concat(y)` → abcdef
- Il messaggio `concat(z)` è inviato alla nuova stringa "abcdef"
  - L'espressione `w.concat(x).concat(y).concat(z)` si riferisce alla stringa risultante
  - `w.concat(x).concat(y).concat(z)` → $abcdefgh_{64}$

Cascata di messaggi

- `s=w.concat(x).concat(y).concat(z)`
- E' il processo di invio di un messaggio ad un oggetto per creare un nuovo oggetto, che a sua volta riceve un messaggio per creare un nuovo oggetto, che …

Composizione di messaggi

- `s=w.concat(x.concat(y.concat(z)))`
- Il messaggio `concat(z)` è inviato a `y`
  - `y.concat(z)` si riferisce alla stringa risultante
  - `y.concat(z)` → efgh
- Un messaggio `concat` con tale nuovo oggetto come argomento è inviato a `x`
  - `x.concat(y.concat(z))` si riferisce alla stringa risultante
  - `x.concat(y.concat(z))` → cdefgh
- Un messaggio `concat` con tale nuovo oggetto come argomento è inviato a `w`
  - `w.concat(x.concat(y.concat(z)))` si riferisce alla stringa risultante
  - `w.concat(x.concat(y.concat(z)))` → abcdefgh

Altri metodi della classe String

- String s = “parola”
- length(): ritorna la lunghezza della stringa

```
int len = s.length(); // len == 6
```

- charAt(int i): ritorna il carattere in posizione i-esima

```
char c=s.charAt(0); // c ==‘p’
```

- indexOf(char c): ritorna l’indice della prima occorrenza del carattere indicato

```
int i=s.indexOf(‘o’); // i == 3
```

- lastIndexOf(char c): come sopra ma per l’ultima occorrenza di c

Esercizi

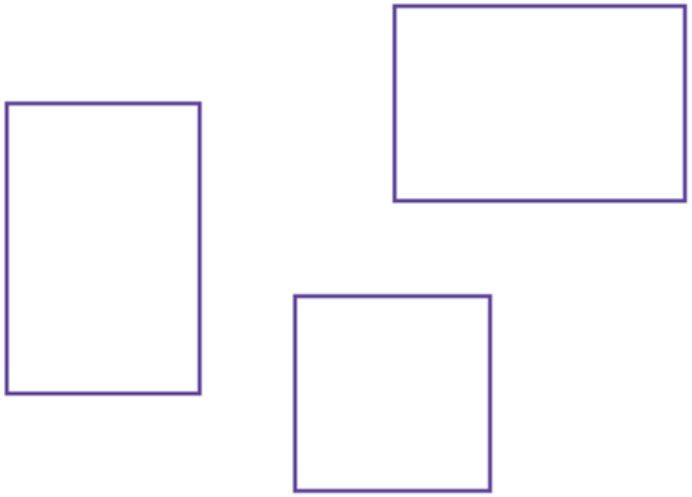
- Scrivere un programma che stampa il carattere centrale di una stringa.

- Scrivere un programma che data una stringa di almeno 2 caratteri, ne costruisce un'altra dove il primo carattere è scambiato con l'ultimo, che viene poi stampata a video.

Esercizi (2)

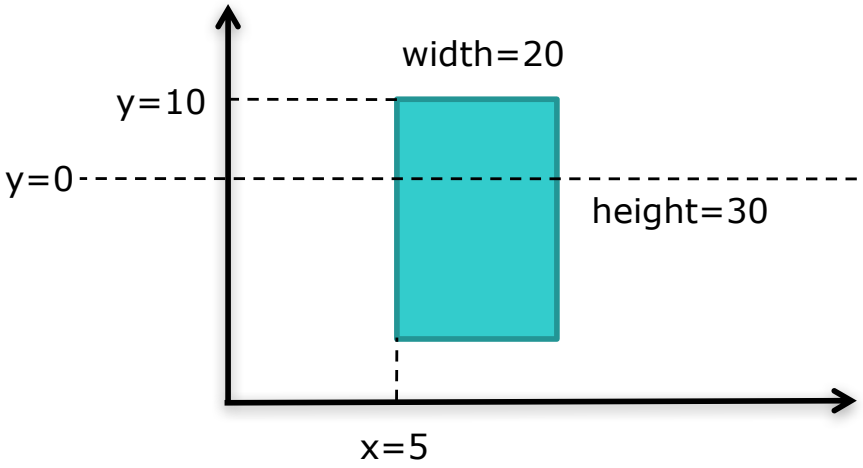
- Contare il numero di caratteri tra la prima occorrenza e l'ultima occorrenza del carattere i nella stringa arrivederci
- Invertire la prima parte con la seconda parte della sequenza di caratteri arrivederci

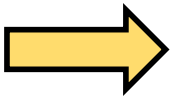
Classe Rectangle

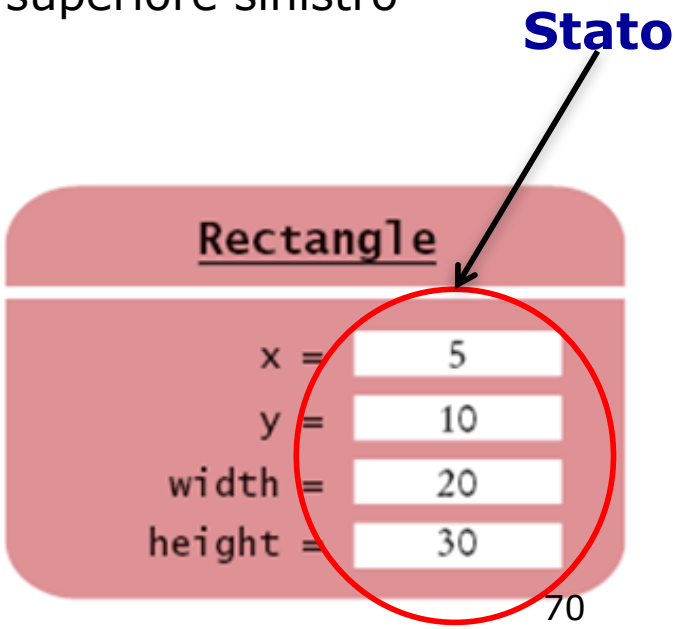


- Oggetti di tipo rettangolo:

- 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)





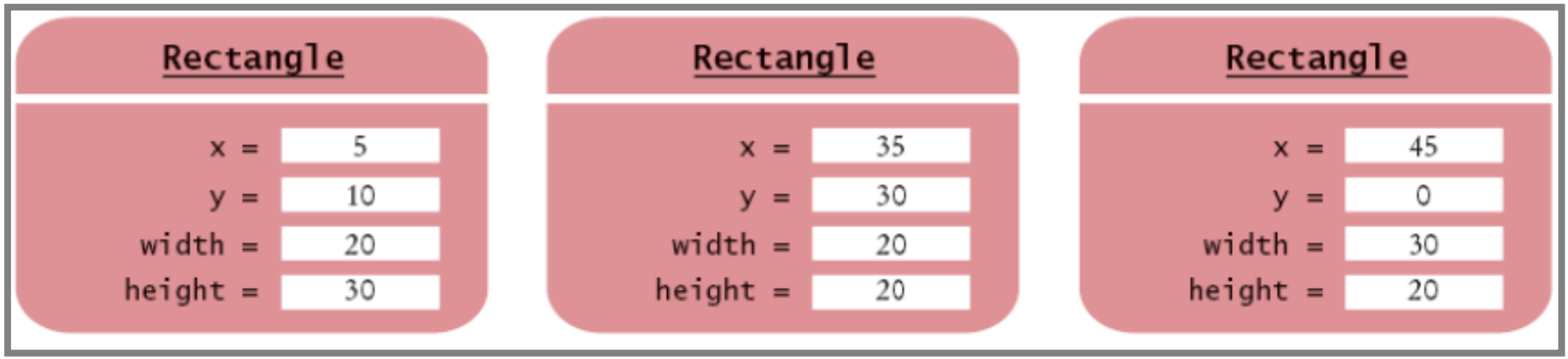


Classe Rectangle cont.

- 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

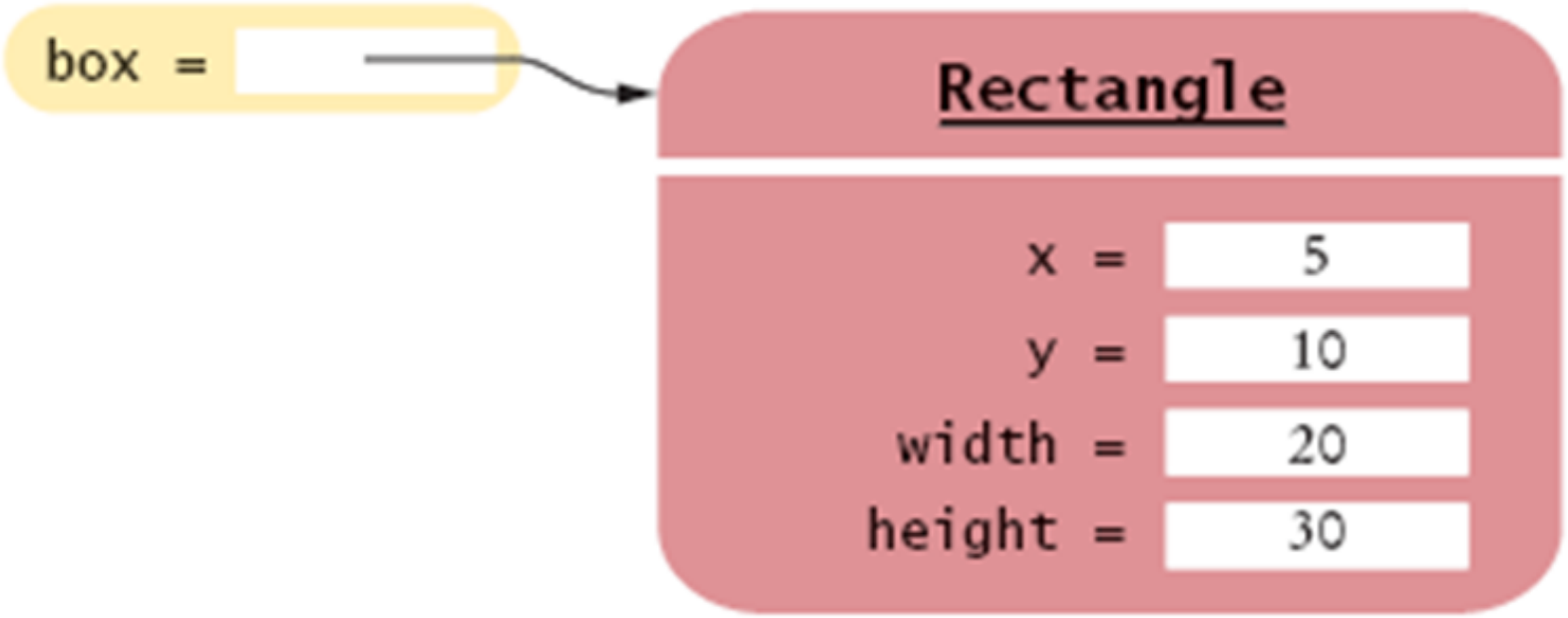


Il metodo costruttore

```
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)

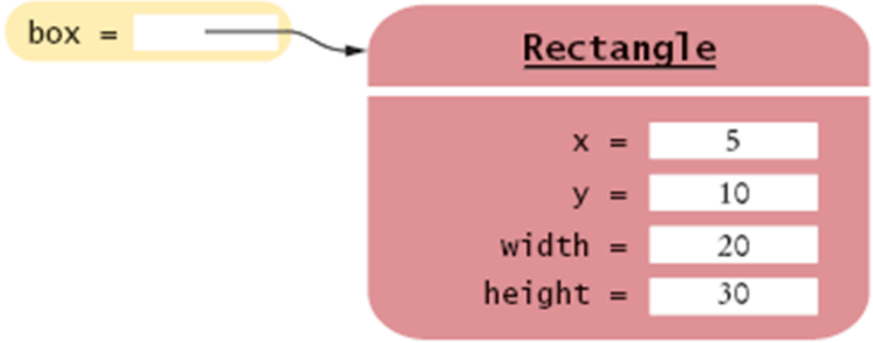
```
Rectangle box;
```

- Definisce una variabile oggetto rettangolo non inizializzata

- E' buona norma inizializzare sempre le variabili oggetto

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

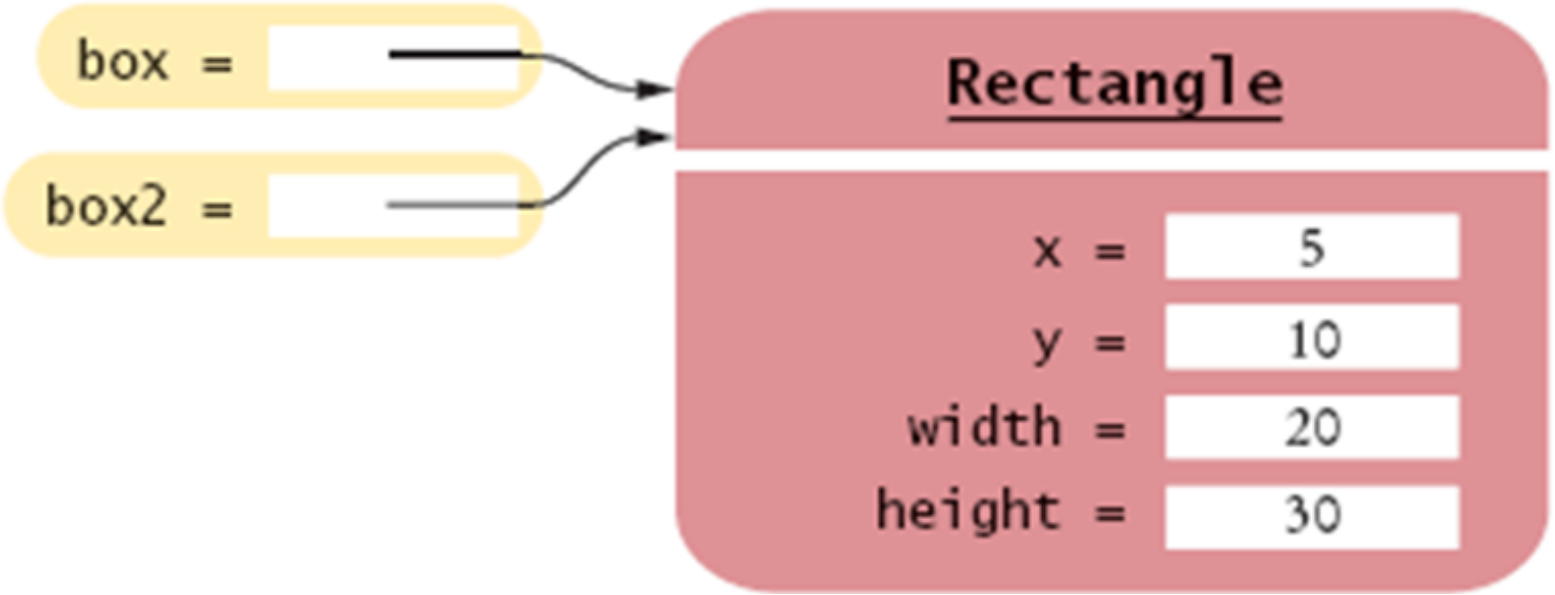
- Crea un rettangolo e assegna il suo indirizzo a **box**



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

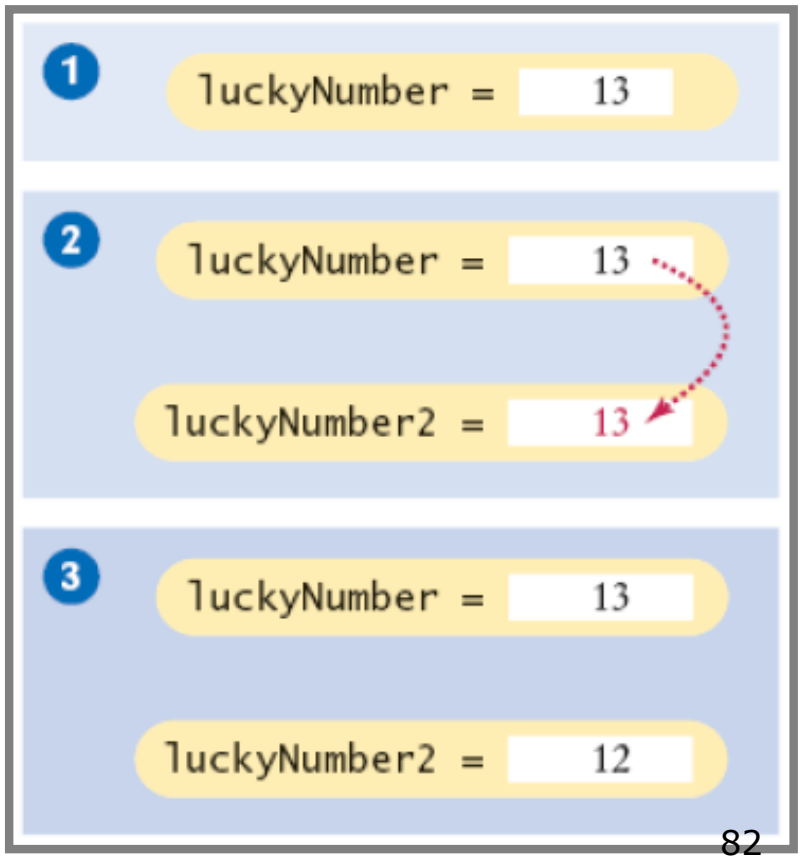
- ```
  new Rectangle(90, 90, 20, 20)
  ```
- 0
  ```

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!");
```

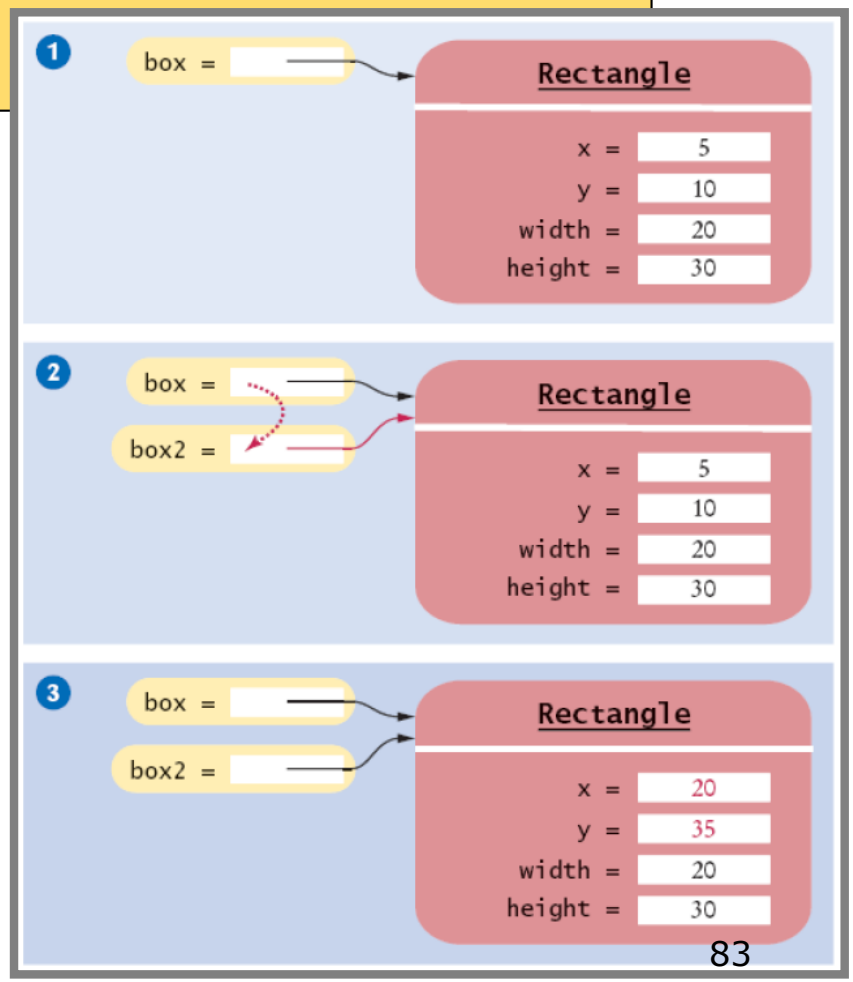
Copiare Numeri

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



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`?

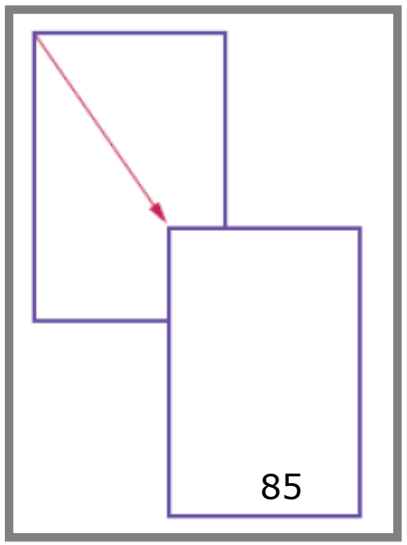
Classificazione metodi

- **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` cosi 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

Importare 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: }
```