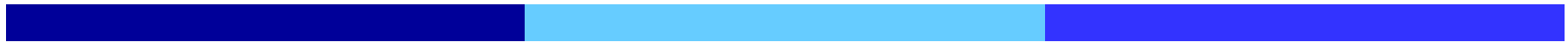


Programmazione II

A.A. 2022-23

Prof. Maria Tortorella



Usare Classi Predefinite

Percorso formativo

- Programmare in Java:
 - **Definire classi**
 - **Istanziare oggetti**
- Imparare ad usare oggetti e classi predefiniti
- Imparare a definire nuove classi



Classi ed oggetti predefiniti

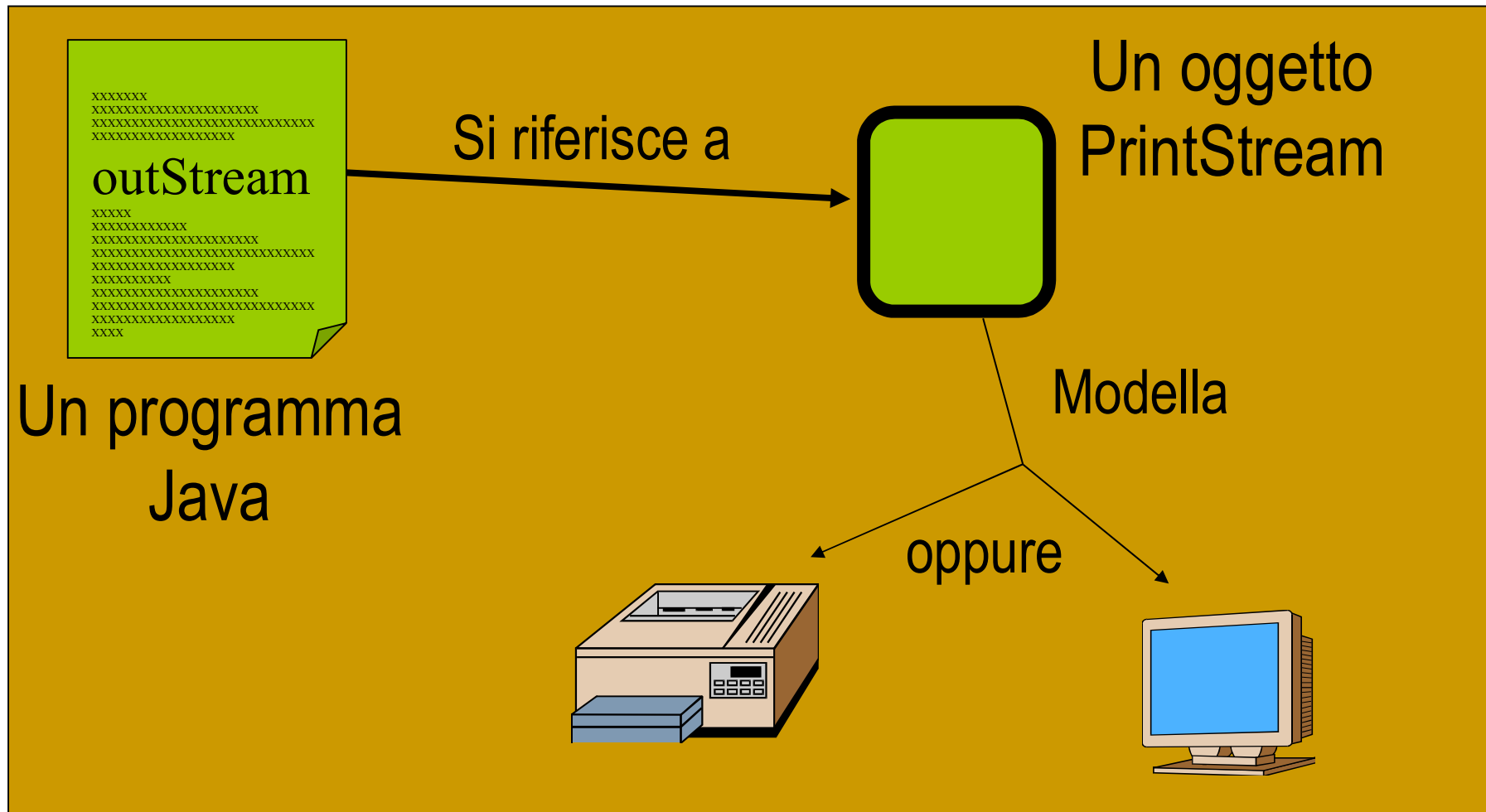
- ❑ Modellano componenti e comportamenti del sistema
- ❑ Modellano l'interfaccia grafica
- ❑ Modellano "oggetti" di uso comune, ad esempio Data e Calendario
- ❑ Un esempio: il monitor
 - accessibile mediante il riferimento: **System.out**

```
import java.io.*;
class Program1 {
    public static void main (String[] arg) {
        System.out.println("Benvenuti al corso");
    }
}
```

PrintStream e System.out

- La classe PrintStream
 - Modella monitor e stampanti
 - Comportamento: visualizzare sequenze di caratteri
- System.out
 - Una reference ad un oggetto predefinito
 - Istanza della classe PrintStream

PrintStream e System.out



Messaggi in Java

- Forma generale
Comportamento-desiderato (altre-informazioni)
- Esempio:
- `println` ("Benvenuti al corso")
 - Comportamento: **println** - stampa una linea
 - Informazione: "Benvenuti al corso"
- contenuto della linea

Invio di un messaggio

- Forma generale:

Riferimento-al-destinatario.messaggio

- Esempio:

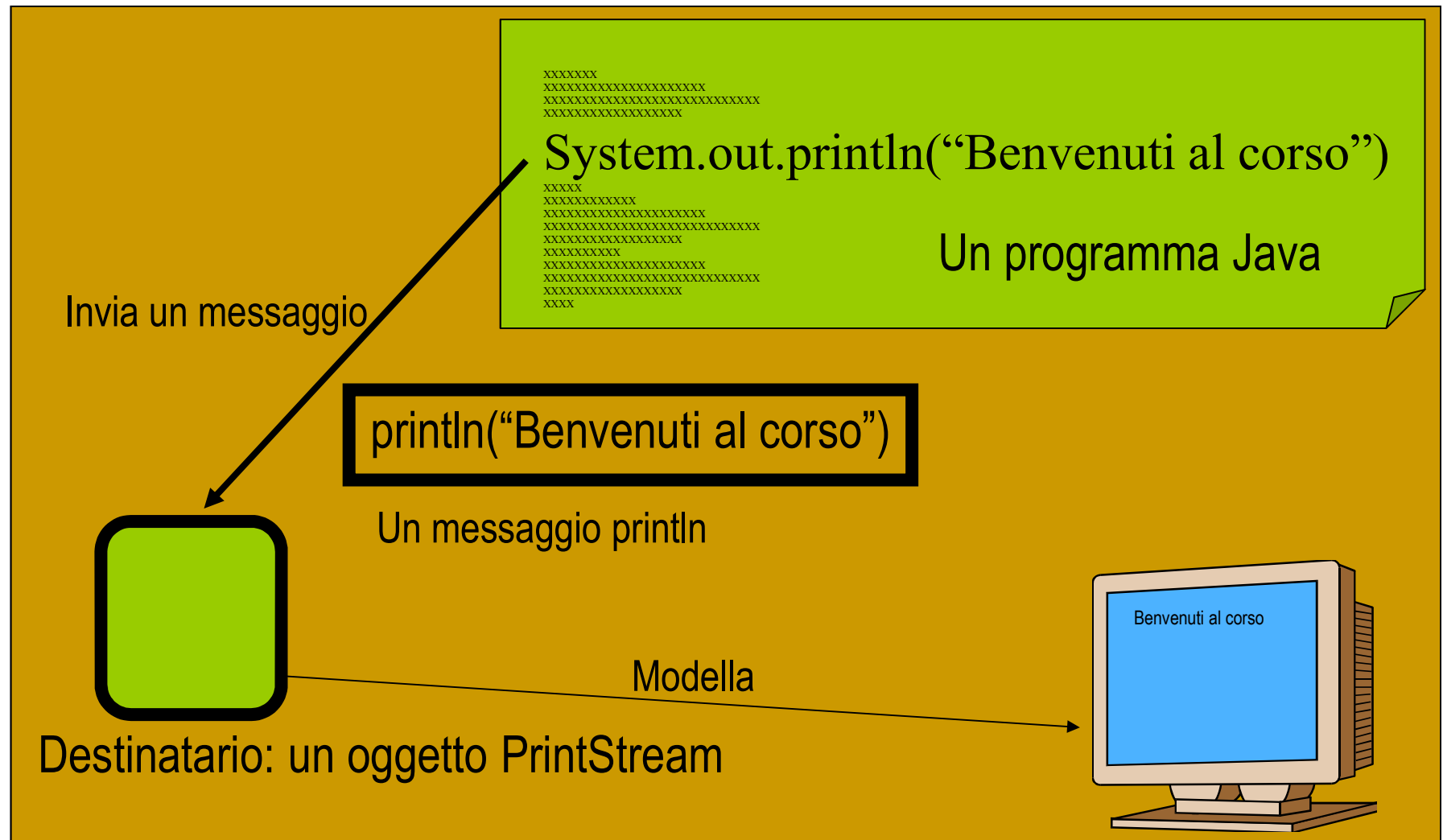
System.out.println ("Benvenuti al corso")

Riferimento

Messaggio

- L'oggetto a cui si riferisce il riferimento
System.out è il destinatario del messaggio
println("Benvenuti al corso")

Invio di un messaggio

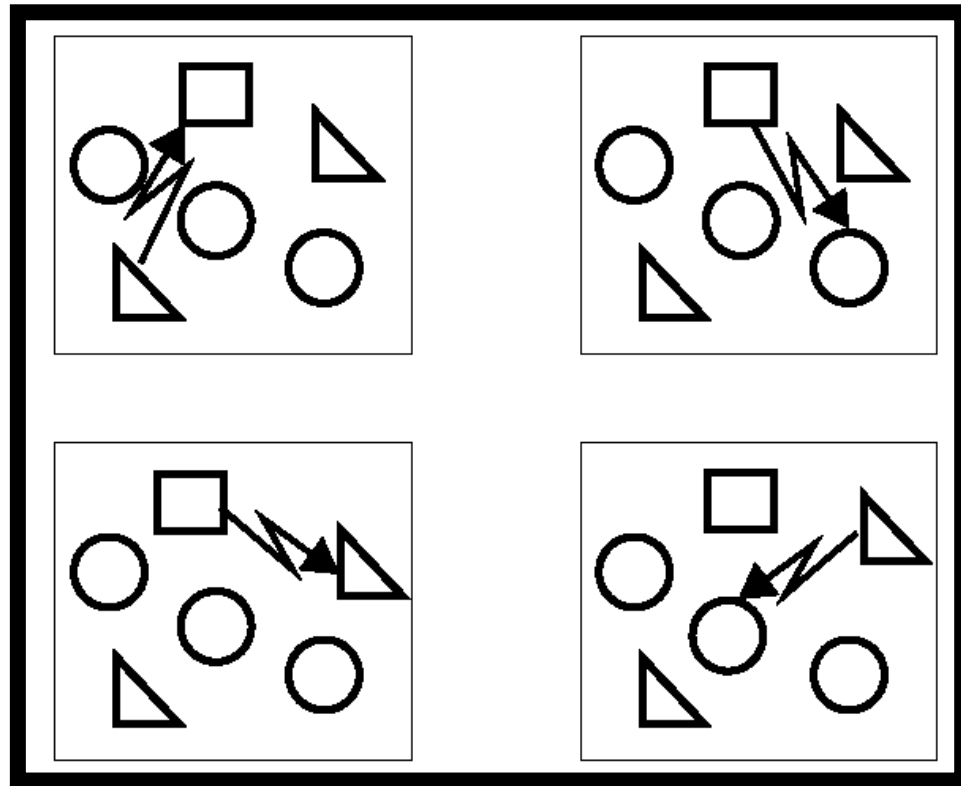


Istruzioni

- Le istruzioni Java
 - Provocano un'azione (es: inviare un messaggio)
 - Devono essere chiusi da punto e virgola ";"
- Esempio
- `System.out.println ("Benvenuti al corso");`
 - L'invio di un messaggio è sempre espresso da una istruzione

Un programma Java

... è un insieme di
oggetti, ognuno
istanza di una
classe, che si
inviano
messaggi ...



Variabili e tipi

- In Java ogni valore appartiene ad un tipo
- Le variabili servono ad immagazzinare valori, compresi i riferimenti ad oggetti
- Esempi di dichiarazioni di variabili

```
String greeting = "Hello, World!";  
PrintStream printer = System.out;  
int luckyNumber = 13;
```

Nota sintattica

typeName variableName = value;

typeName variableName;

Esempio:

```
String greeting = "Hello, Dave!";
```

Definisce una nuova variabile di un certo tipo ed eventualmente assegna un valore iniziale

String è una classe

Operatore di assegnazione

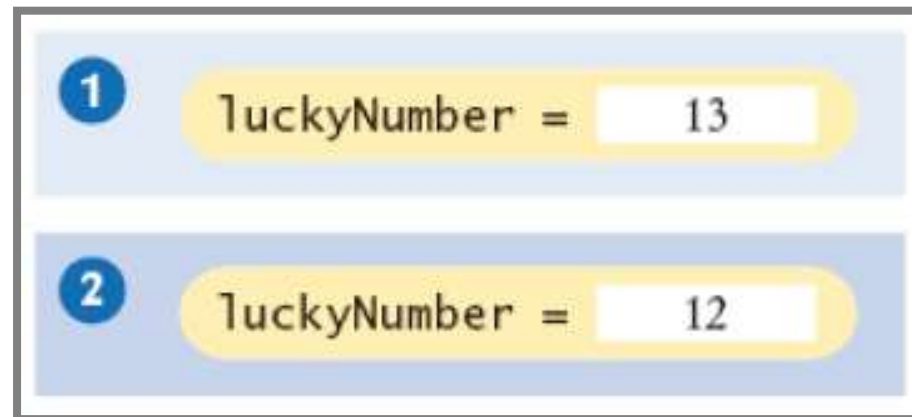
- Operatore =
 - Non un test di eguaglianza per confrontare due valori
 - Cambia il valore di una variabile

```
int luckyNumber = 13;
```

1

```
luckyNumber = 12;
```

2



Uso di variabili non inizializzate

- ❑ Ogni uso di variabile deve essere preceduto da almeno un'assegnazione di valore

```
int luckyNumber;  
System.out.println(luckyNumber);  
// ERROR - uninitialized variable
```

luckyNumber =

Nota sintattica

```
variableName = value;
```

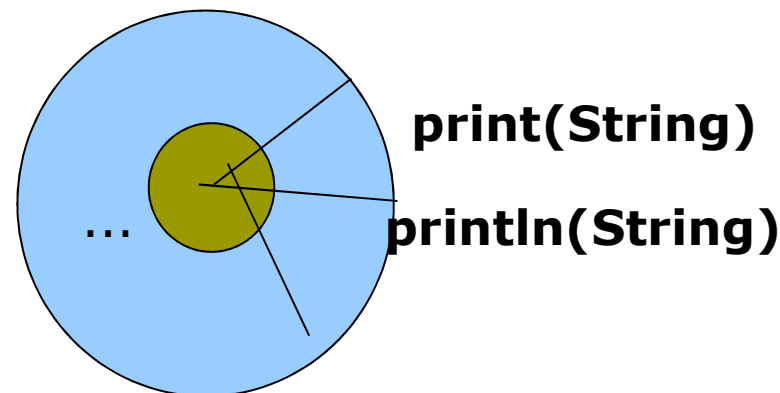
Esempio:

```
luckyNumber = 12;
```

Assegna un nuovo valore ad una variabile definita in precedenza

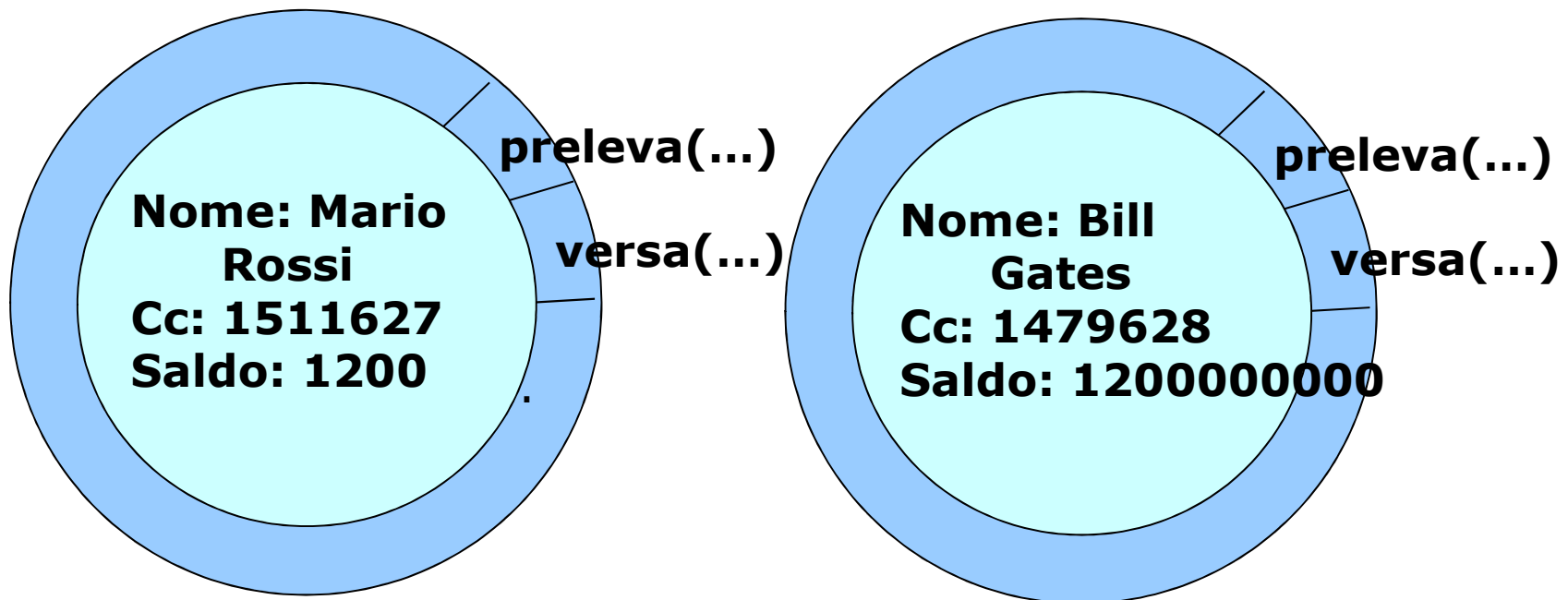
Oggetti e classi

- ❑ Oggetto: astrazione di una entità del dominio del problema
- ❑ Ogni oggetto appartiene ad una classe
 - **Istanza**
- ❑ L'oggetto referenziato da `System.out` appartiene alla classe `PrintStream`



Oggetti e stato

- Ogni oggetto ha un suo stato distinto dagli altri oggetti
 - Lo **stato** è l'insieme dei valori assunti dalle variabili d'istanza



Messaggi e metodi

- Un comportamento di un oggetto è attivato dalla ricezione di un messaggio
- Le classi determinano il comportamento degli oggetti definendo quali sono i messaggi “leciti”
- Le classi determinano i messaggi leciti mediante la definizione di metodi:
 - **Una sezione di codice all’interno di una classe che implementa un particolare comportamento**

Forma di un messaggio

nome-del-metodo(argomenti)

- ▣ Un messaggio deve specificare
 - **Il nome del metodo da invocare**
... il comportamento desiderato
 - **Gli eventuali argomenti**
... altre informazioni

System.out.println ("Benvenuti al corso")

Nome del metodo

Argomenti

I metodi di PrintStream

- ▣ Conoscere una classe equivale a conoscerne i metodi

La classe: PrintStream

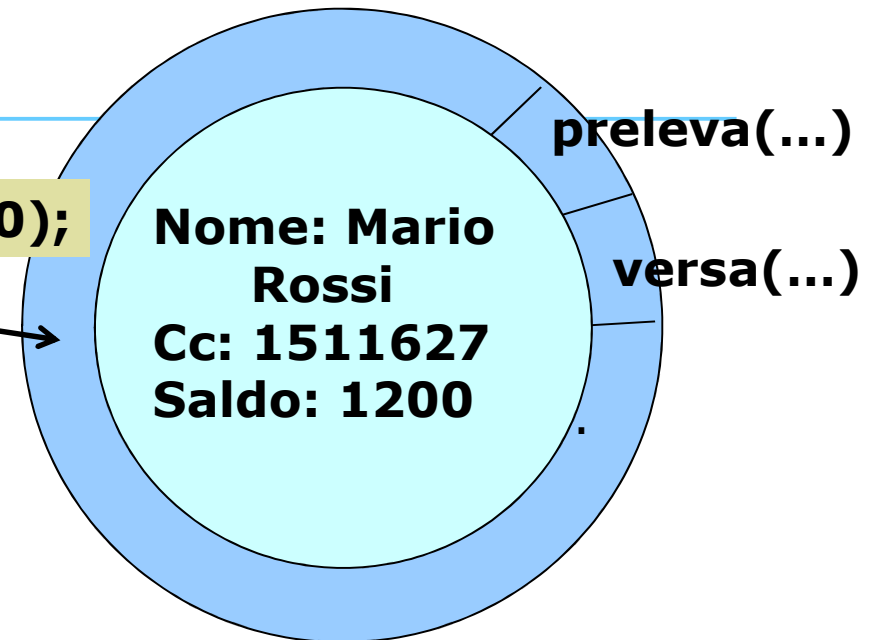
<u>Nome</u>	<u>Argomenti</u>
println	stringa di caratteri
println	nessuno
print	stringa di caratteri

Oggetti e classi

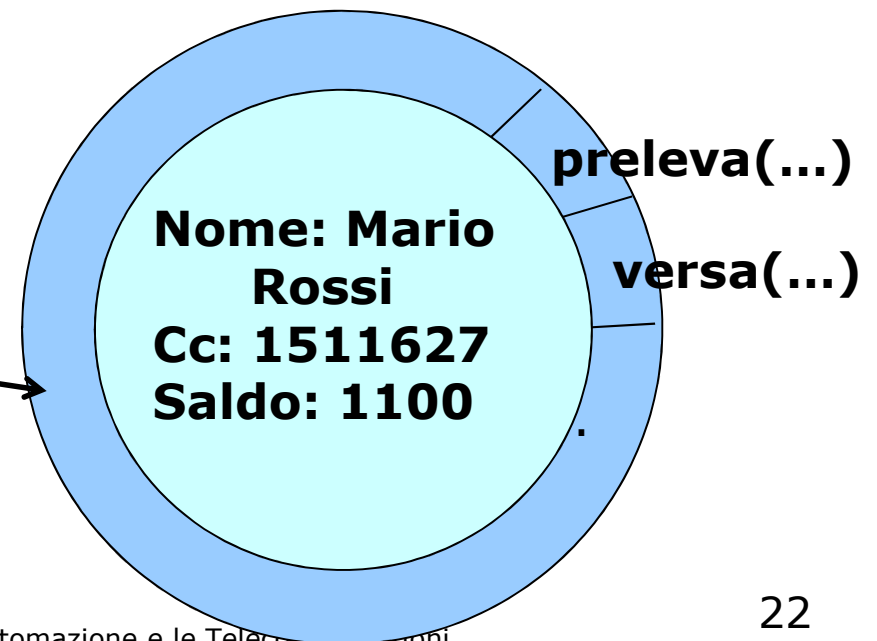
- Classe: insieme di oggetti con lo stesso comportamento
 - **La classe determina i metodi dell'oggetto, e quindi i messaggi cui l'oggetto può rispondere**
- Metodo: frammento di codice che accede lo stato (i dati) di un oggetto
 - **Si manipola un oggetto inviando un messaggio che provoca l'esecuzione di uno dei suoi metodi**
- Interfaccia pubblica: l'insieme dei metodi di un oggetto che possono essere invocati da altri oggetti

Esempio

```
contoCorrenteMarioRossi.preleva(100);
```



contoCorrenteMarioRossi



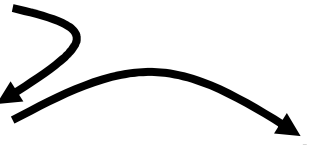
La segnatura di un metodo

- println("salve") e println() sono lo stesso metodo?
- Due metodi differenti (**metodi overloaded**)
 - **Stesso nome**
 - **Argomenti diversi**
 - **Comportamento diverso**
- I metodi sono individuati dalla segnatura, e non solo dal nome

La segnatura (signature) di un metodo:
Il nome del metodo + la descrizione degli argomenti
- **Overloading**: la possibilità di avere una classe che definisca metodi differenti con lo stesso nome
 - println è un metodo *overloaded* della classe PrintStream

Invio di un messaggio (I)

```
statement1;  
statement2;  
referenceToX.methodA();  
statement4;
```



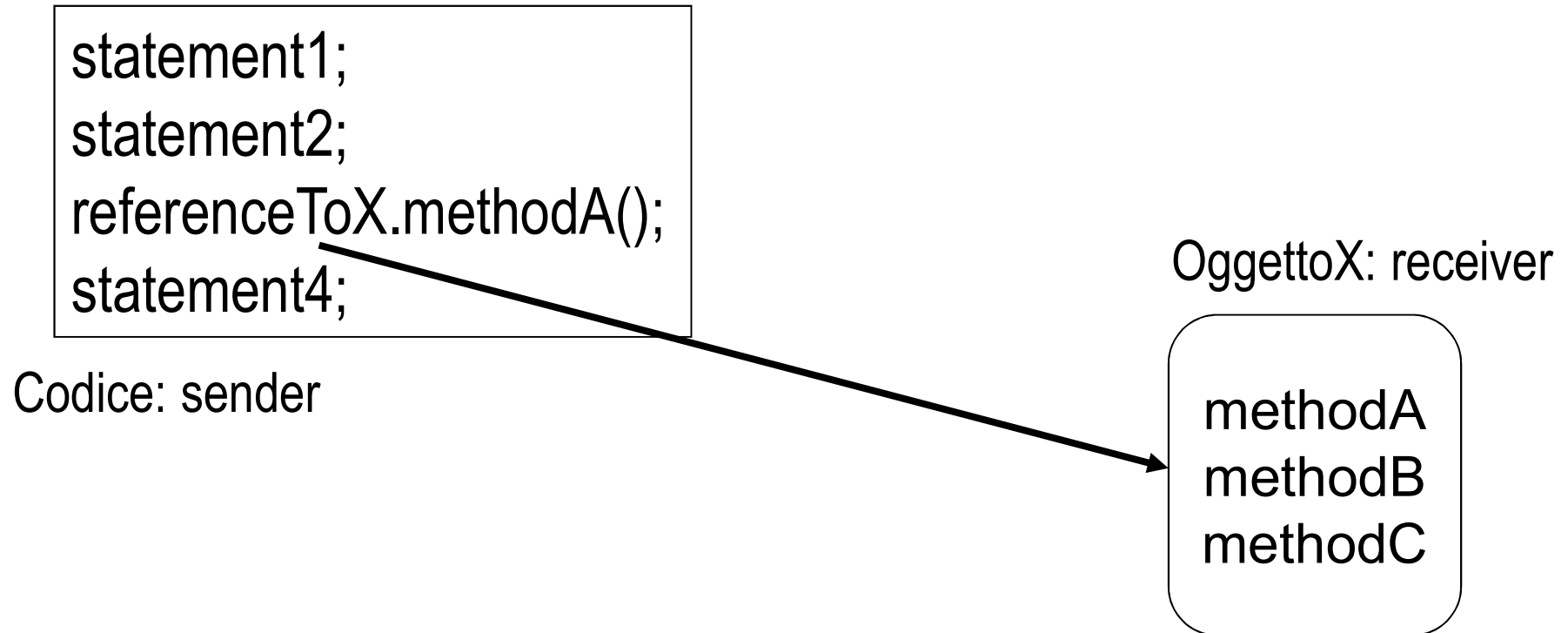
Codice

OggettoX

methodA
methodB
methodC

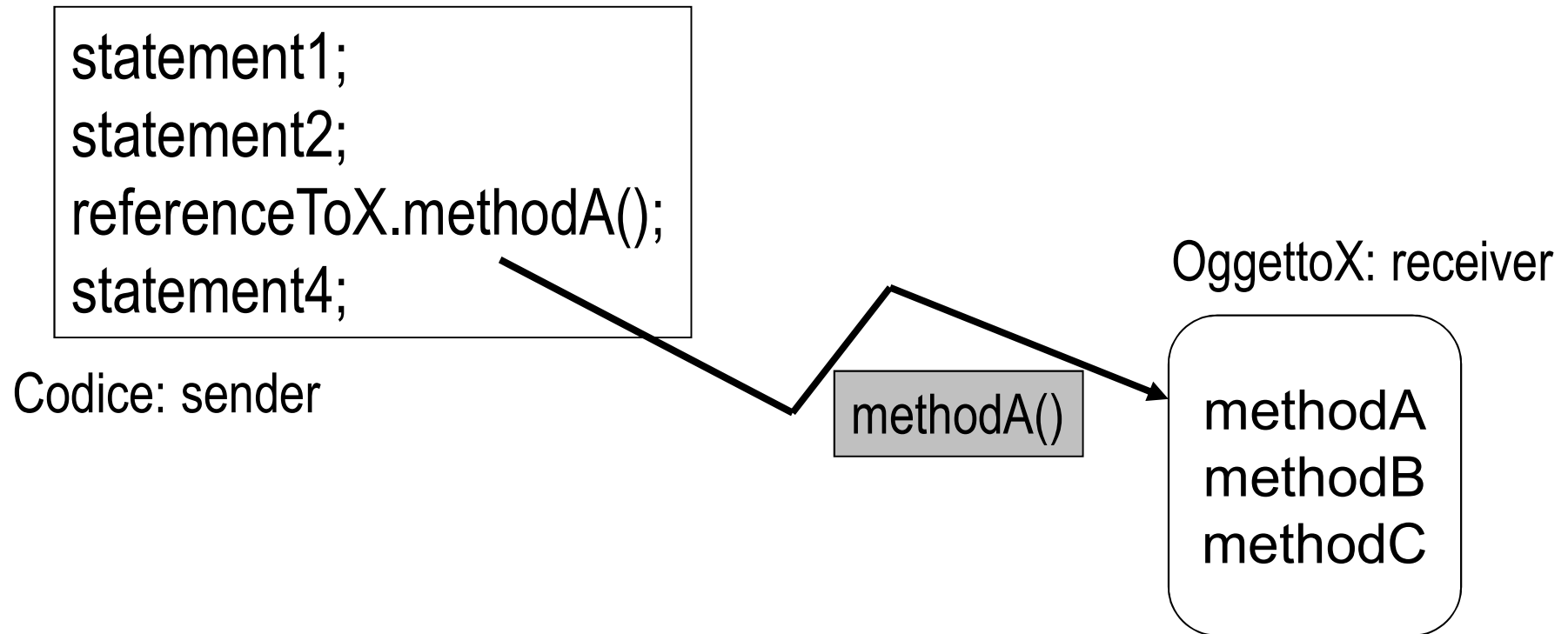
- ▣ Ordine di esecuzione sequenziale, fino a raggiungere una istruzione di invio di un messaggio

Invio di un messaggio (II)



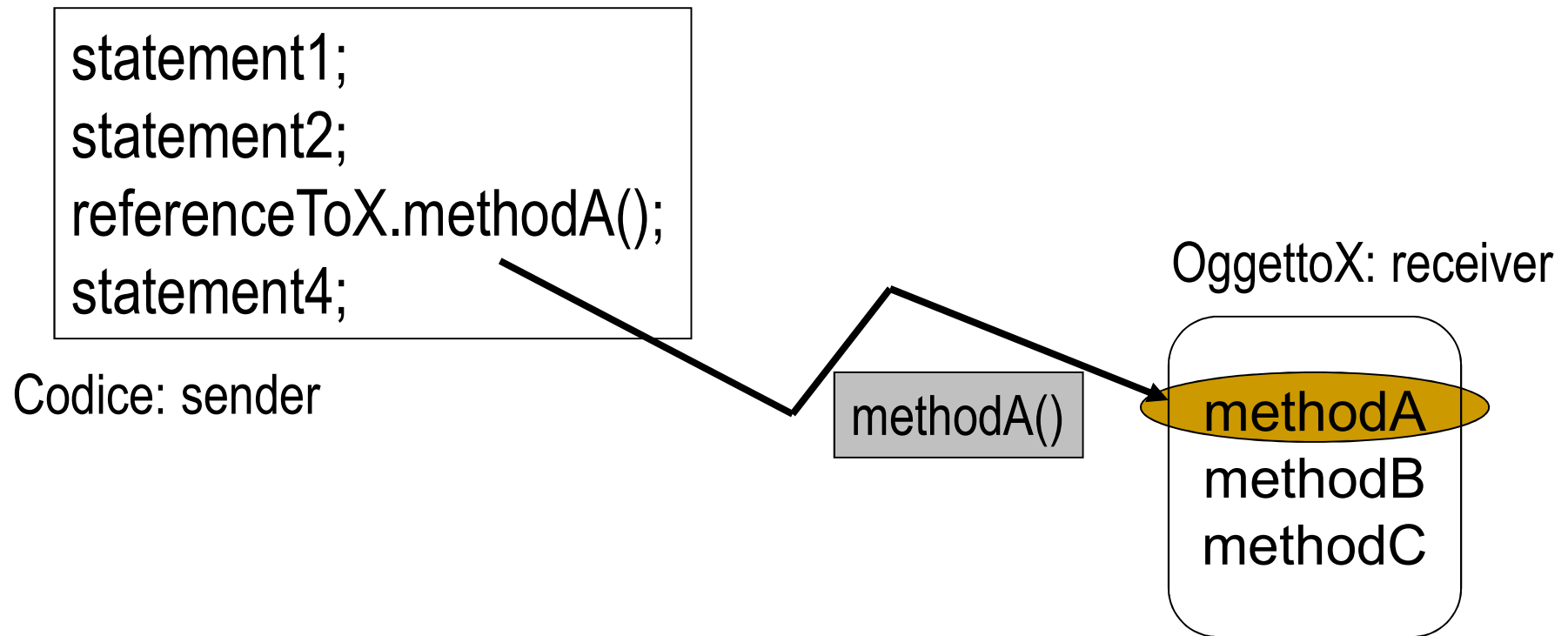
- L'esecuzione del sender è sospesa

Invio di un messaggio (III)



- Il messaggio è inviato al receiver

Invio di un messaggio (IV)



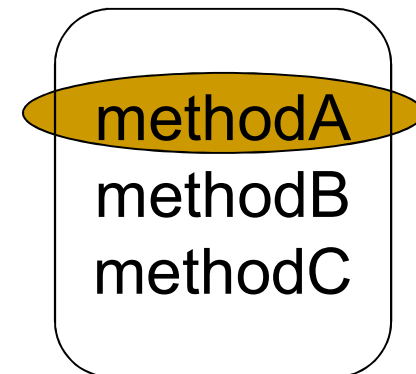
- L'arrivo del messaggio provoca l'invocazione di uno dei metodi del receiver

Invio di un messaggio (V)

```
statement1;  
statement2;  
referenceToX.methodA();  
statement4;
```

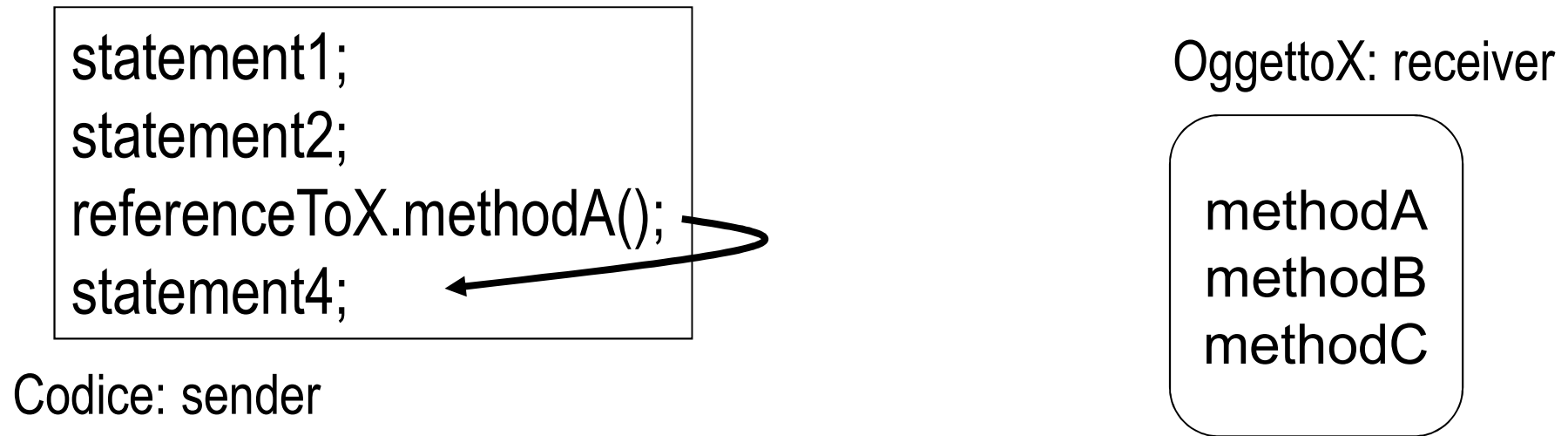
Codice: sender

OggettoX: receiver



- Il codice relativo al metodo invocato viene eseguito; questo può eventualmente provocare l'invio di altri messaggi ad altri oggetti

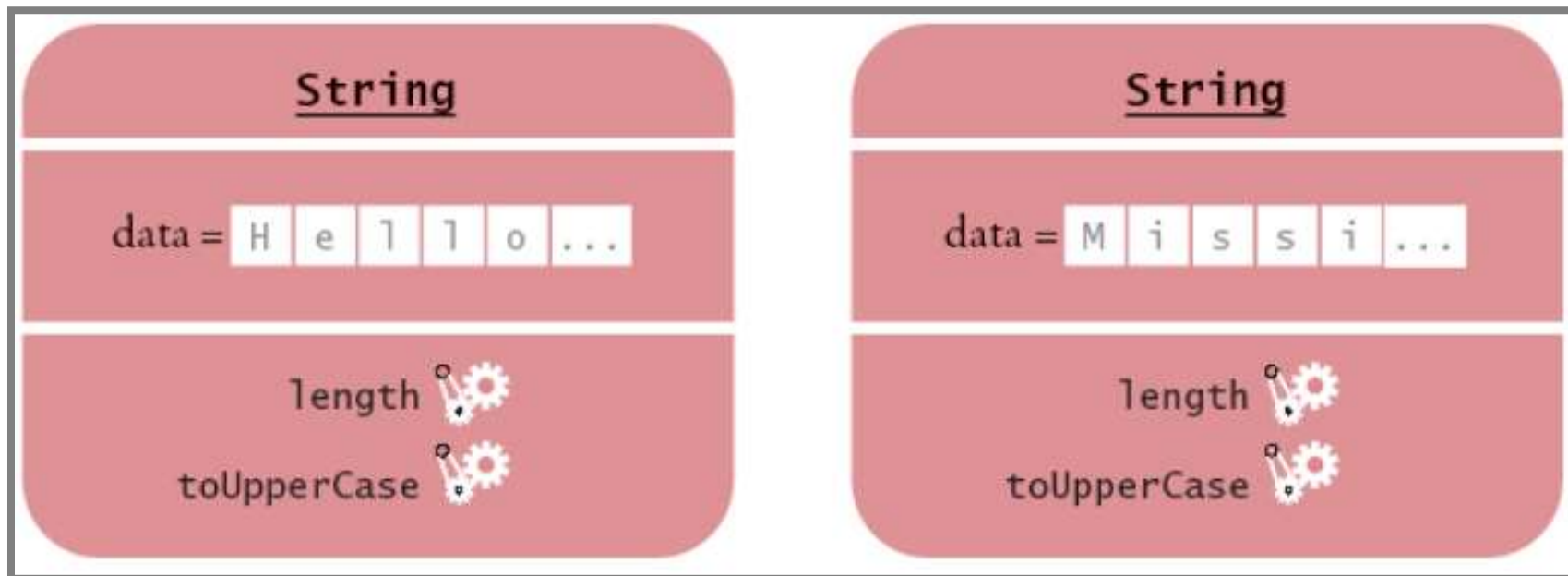
Invio di un messaggio (VI)



- Quando l'esecuzione del metodo invocato termina
 - Il controllo (ed eventuali informazioni aggiuntive) vengono restituite al sender (return)
 - Riprende l'ordine sequenziale

La classe String

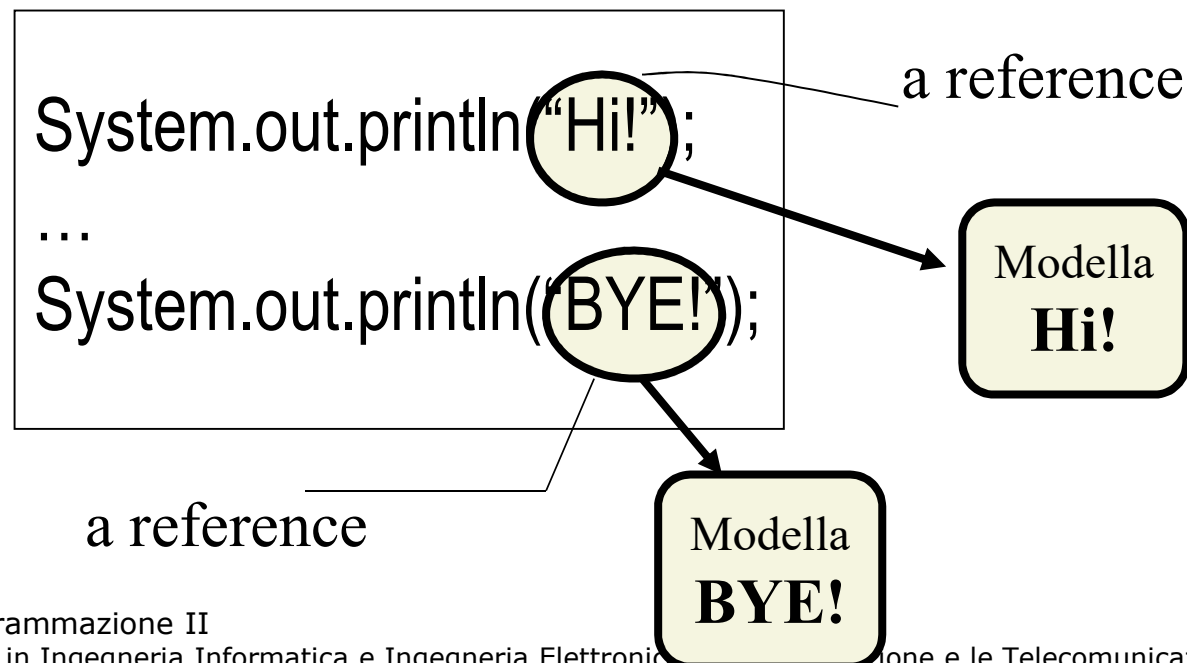
- Una classe predefinita
 - [`java.lang.String`](#)
- Modella una qualunque sequenza di caratteri



String: referenze ed oggetti

- **Referenze oggetti “costante” String**
 - **Sequenze di caratteri fra doppi apici**

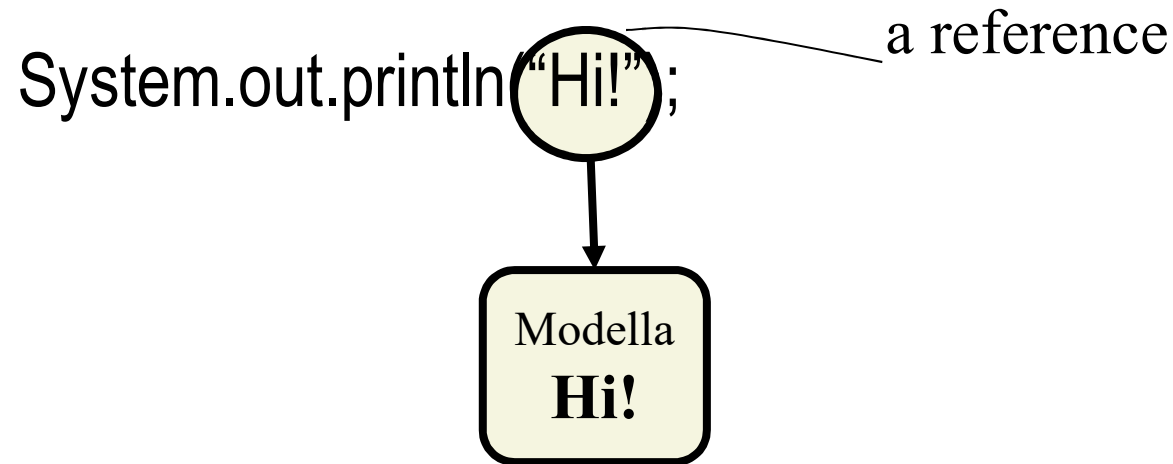
“Hi!” e “BYE!” sono due riferimenti a oggetti String che modellano le sequenze di caratteri Hi! e BYE!



Riferimenti a stringhe

esempi di utilizzo

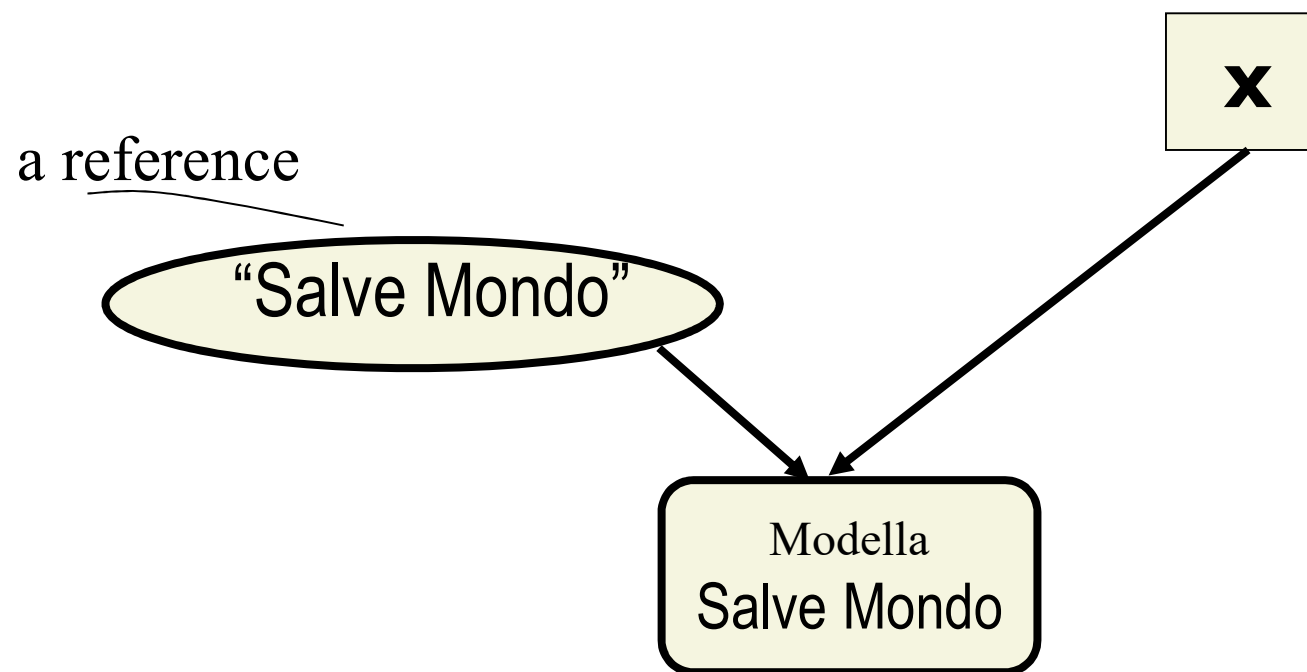
- Come argomento di un messaggio
 - Uno dei metodi println di PrintStream ha un argomento che è un riferimento ad un oggetto stringa
 - println(riferimento-ad-un-oggetto-String)



Riferimenti a stringhe

esempi di utilizzo

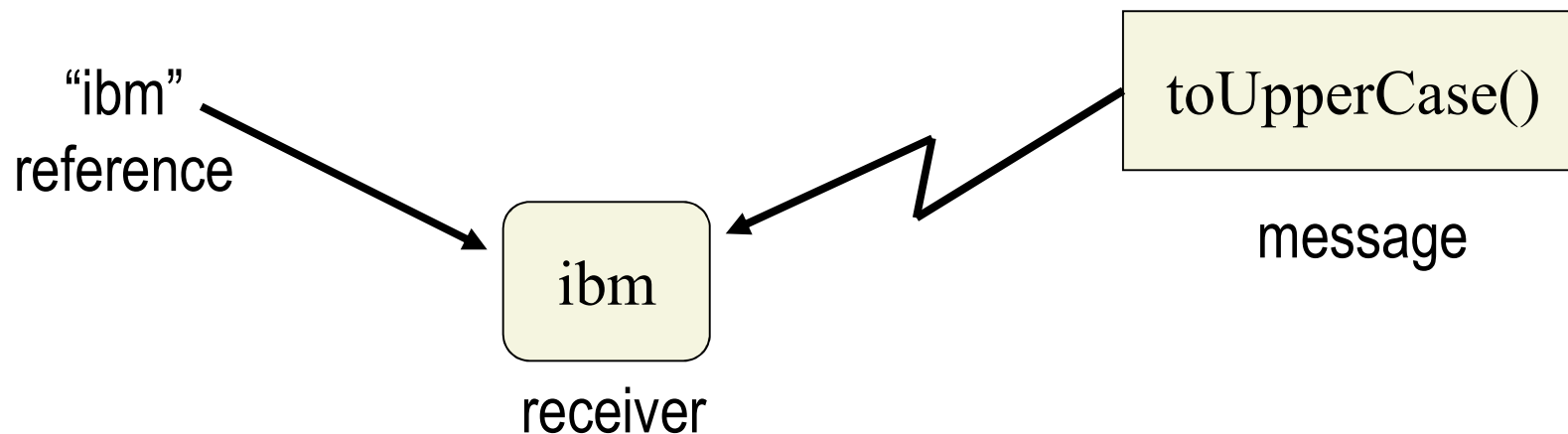
- ▣ Avvalorare una variabile
String x = "Salve Mondo"



Riferimenti a stringhe

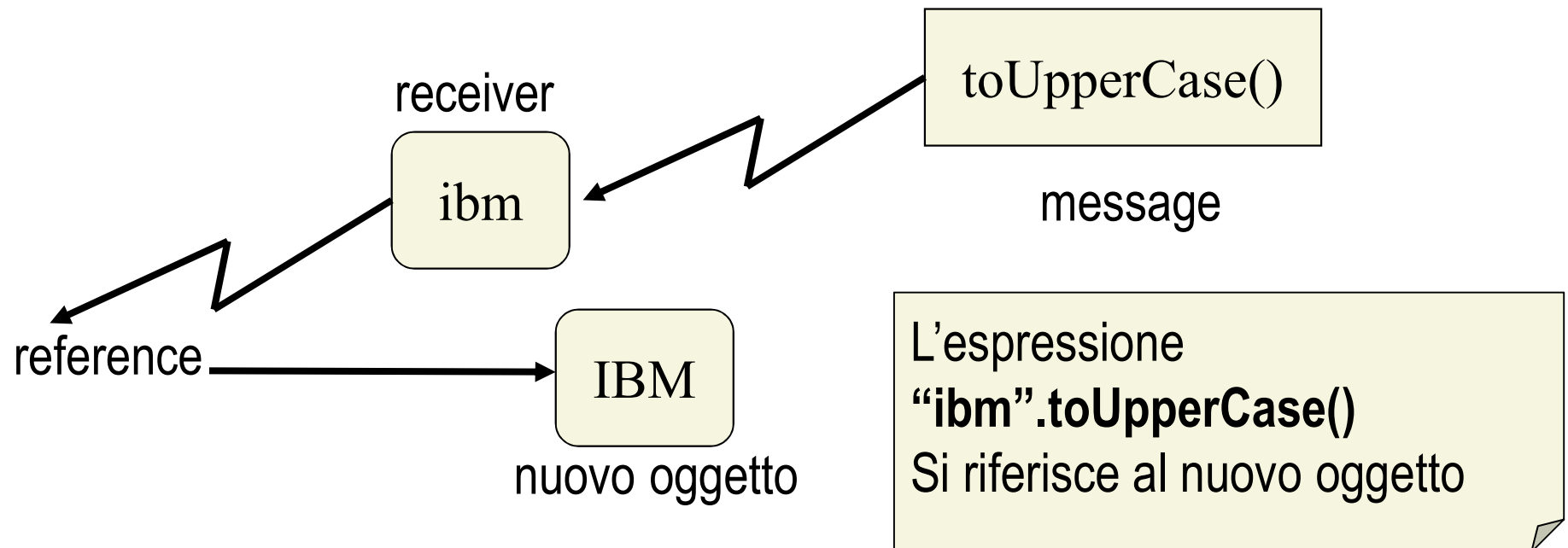
esempi di utilizzo

- ▣ Invio di un messaggio ad una stringa
- ▣ La classe String offre molti metodi
- ▣ Un esempio: toUpperCase()
"ibm".toUpperCase()



Il metodo toUpperCase

- ▣ Crea un nuovo oggetto String
- ▣ Tutti i caratteri sono in maiuscolo
- ▣ Restituisce (returns) un riferimento (reference) al nuovo oggetto



Parametri espliciti ed impliciti

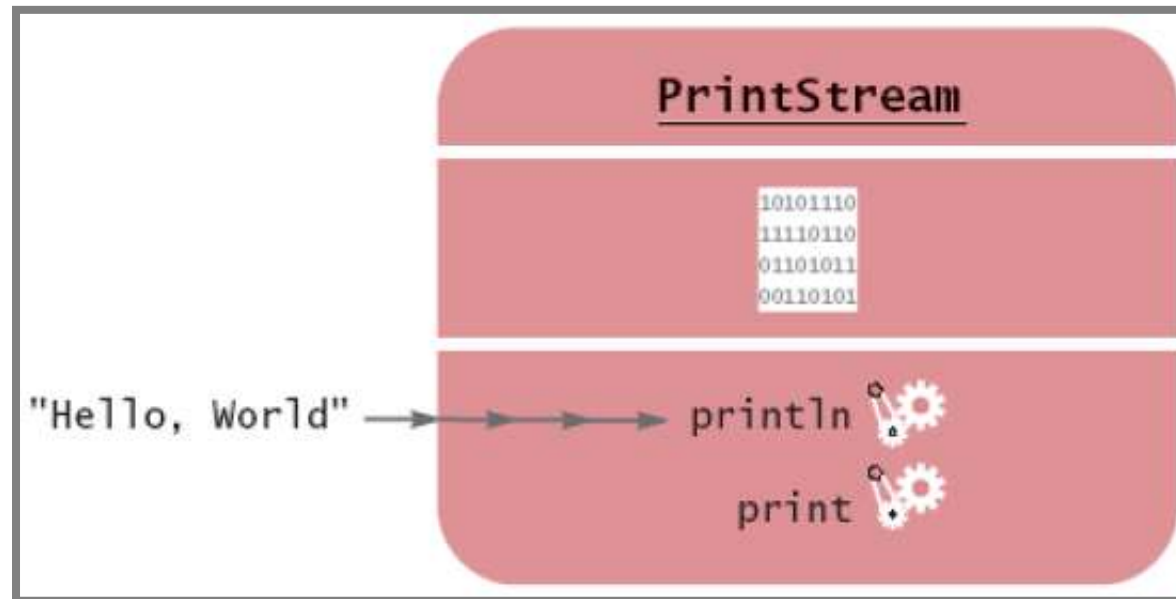
- Parametri espliciti: i dati esplicitamente passati ad un metodo
 - non necessariamente presenti

```
System.out.println(greeting)  
greeting.length() // has no explicit parameter
```

- Parametro implicito: receiver del messaggio
 - l'oggetto su cui si applica il metodo

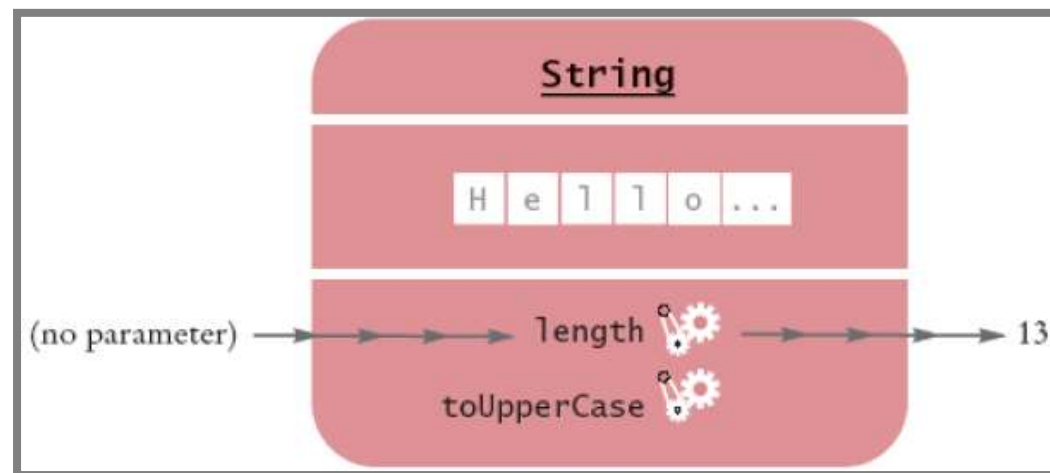
```
System.out.println(greeting)
```

Parametri



Valore di restituzione

- Il risultato dell'invocazione di un metodo, messo a disposizione del chiamante
 - non tutti i metodi restituiscono un valore

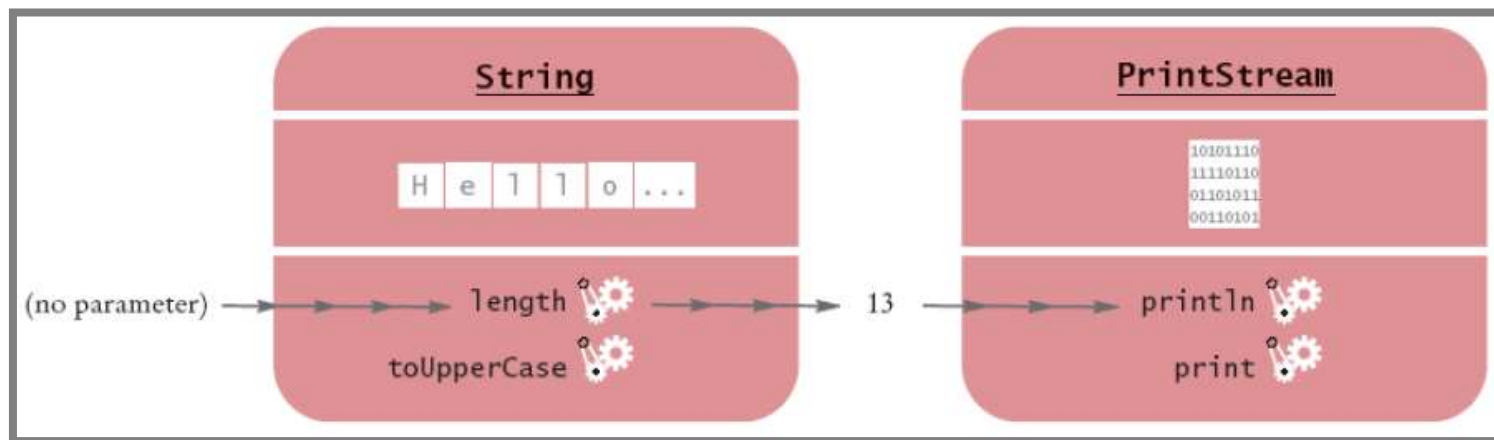


```
int n = greeting.length();  
// return value stored in n
```

Valore di restituzione

- Può essere utilizzato come parametro in un messaggio

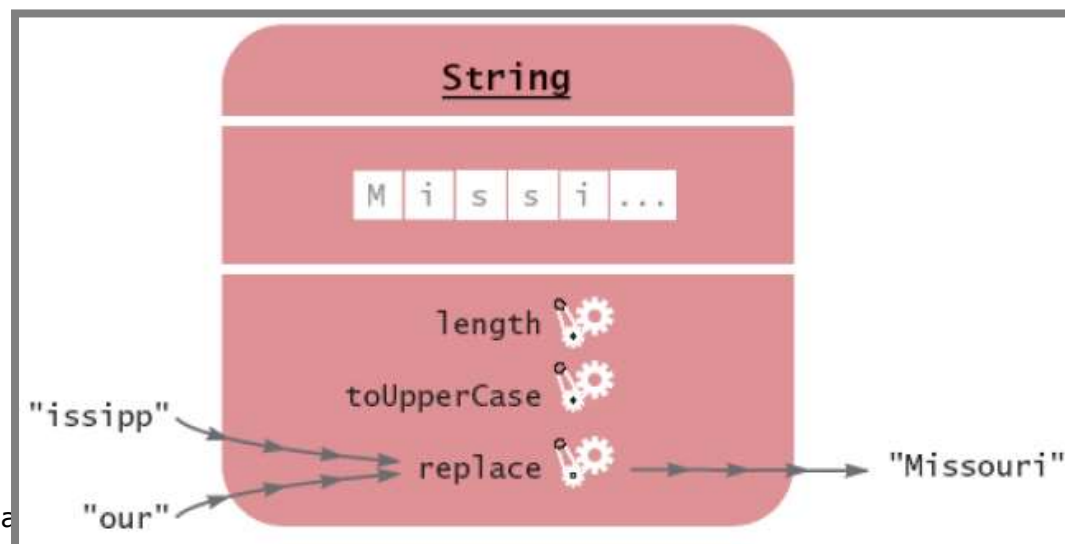
```
System.out.println(greeting.length());
```



Un esempio complesso

- ❑ Il metodo replace compie operazioni di ricerca e sostituzione
 - un parametro implicito
 - due parametri espliciti
 - un valore di restituzione

```
river.replace("issipp", "our")  
// constructs a new string ("Missouri")
```



Definizione di un metodo

- La definizione di un metodo specifica i tipi dei parametri espliciti e del valore di restituzione
 - il parametro implicito è l'oggetto di invocazione, o receiver del messaggio
 - Esempio: `Class String`

```
public int length()  
    // return type: int  
    // no explicit parameter  
public String replace(String target, String replacement)  
    // return type: String;  
    // two explicit parameters of type String
```

- In assenza del parametro di restituzione il metodo è void

```
public void println(String output) // in class PrintStream
```

Segnatura e prototipo

<u>class</u>	<u>method</u>	<u>returns</u>	<u>arguments</u>
PrintStream	println	nothing	none
PrintStream	println	nothing	ref. to String object
PrintStream	print	nothing	ref. to String object
String	toUpperCase	ref. to String object	none

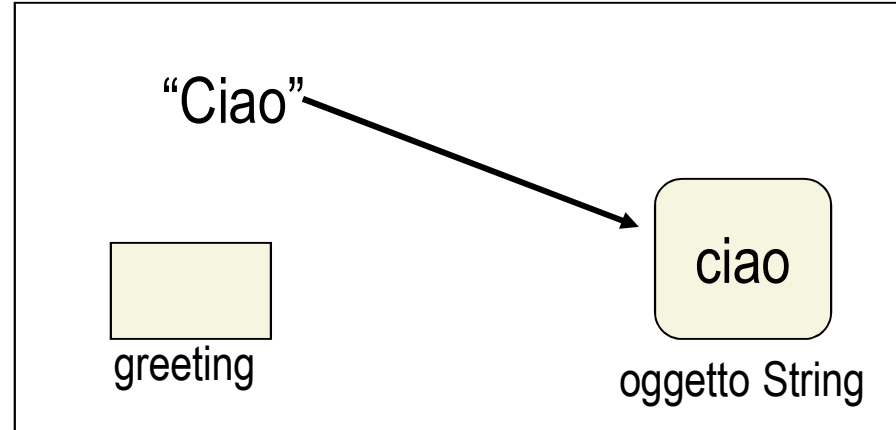
- ▣ Segnatura: nome + argomenti ricevuti
- ▣ Prototipo: segnatura + valore restituito

Variabili di riferimento

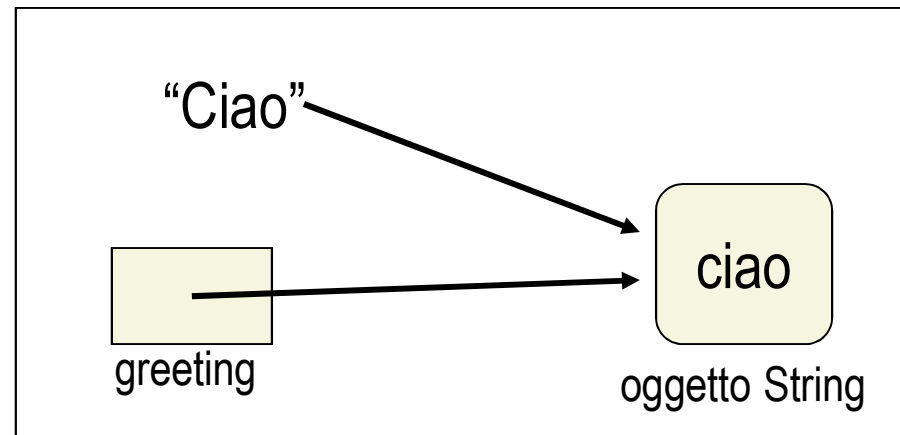
- Una variabile il cui valore è un riferimento ad un oggetto
- Dichiarazioni
`String greeting;`
`PrintStream output;`
- E' necessario assegnare un valore ad una variabile di riferimento prima di poterla utilizzare
`greeting = "Ciao";`
`greeting = System.out; // ??`
- Il tipo del valore deve combaciare con il tipo con cui si è dichiarata una variabile (type matching)
- In generale:
`variabile = valore;`
Il valore è copiato nella variabile

Dichiarazione ed assegnamento

```
String greeting;  
greeting = "ciao";
```



```
String greeting;  
greeting = "ciao";
```



Esempo (I)

```
String s1, s2;  
PrintStream ps1, ps2;  
s1 = "hello";  
s2 = "goodbye";  
s1 = s2;  
ps2 = System.out;  
ps1 = ps2;  
ps1.println(s1);           // cosa succede ?
```

Esempio (II)

```
String greeting;  
greeting = "hey!";  
String bigGreeting;  
bigGreeting = greeting.toUpperCase();  
System.out.println(bigGreeting);  
System.out.println(bigGreeting);  
System.out.println(bigGreeting);
```

... analogamente ...

```
System.out.println(greeting.toUpperCase());  
System.out.println(greeting.toUpperCase());  
System.out.println(greeting.toUpperCase());
```

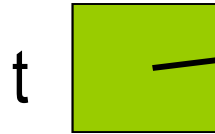
L'assegnamento non è uguaglianza

```
String t;  
t = "Cash";  
t = "Credit";
```

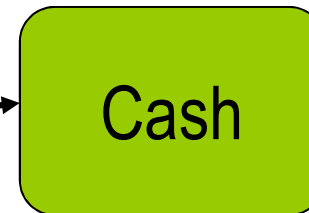
String t;



t = "Cash";

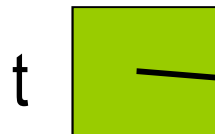


si riferisce a

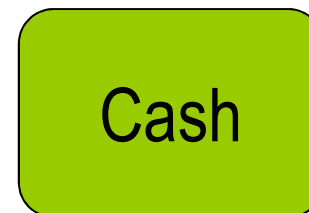


oggetto String

t = "Credit";



si riferisce a



oggetto String

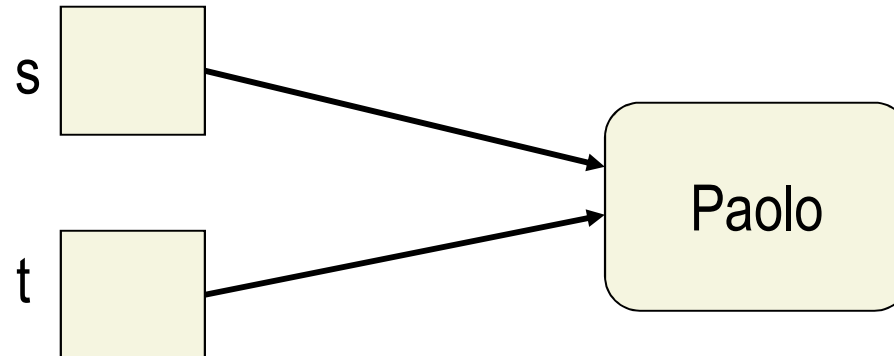


oggetto String

Variabili e oggetti

- Una variabile di riferimento si riferisce ad un solo oggetto alla volta
- Un oggetto può essere referenziato da più variabili simultaneamente

```
String s, t;  
s="Paolo";  
t=s;
```



Esempio (I)

```
class Esempio {  
    public static void main(String arg[]) {  
        String          greeting;  
        String          bigGreeting;  
        greeting = "Yo, World";  
        bigGreeting = greeting.toUpperCase();  
        System.out.println(greeting);  
        System.out.println(bigGreeting);  
    }  
}
```

Esempio (II)

```
class Esempio {  
    public static void main(String arg[]) {  
        String          greeting;  
        greeting = "Yo, World";  
        String          bigGreeting;  
        bigGreeting = greeting.toUpperCase();  
        System.out.println(greeting);  
        System.out.println(bigGreeting);  
    }  
}
```

Esempio (III)

```
class Esempio {  
    public static void main(String arg[]) {  
        String greeting = "Yo, World";  
        String bigGreeting = greeting.toUpperCase();  
        System.out.println(greeting);  
        System.out.println(bigGreeting);  
    }  
}
```

Ancora sulle stringhe

▣ Prototipi di metodi della classe String

<u>method</u>	<u>returns</u>	<u>arguments</u>
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

```
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
 - Referenza: ""

Oggetti intelligenti ed utili

- Notare che per trovare una sottostringa di una data stringa inviamo un messaggio alla stringa
 - E' la stringa a trovare la sottostringa per noi
- Progettare le classi in modo da rendere gli oggetti *utili*, o *intelligenti*, in modo che possa su richiesta eseguire operazioni
 - I metodi toUpperCase e substring dimostrano che i progettisti Java hanno seguito tale principio

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

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`

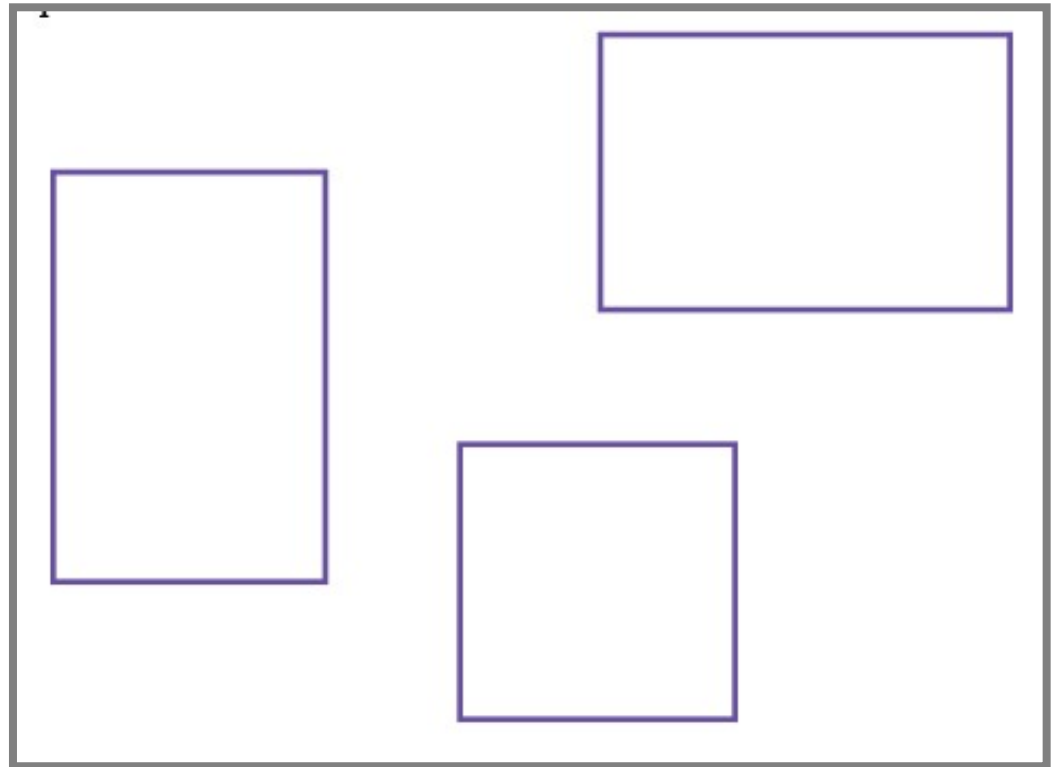
■ **`w.concat(x.concat(y.concat(z)))` si riferisce alla stringa risultante**

■ **`w.concat(x.concat(y.concat(z)))`**

abcdefgh

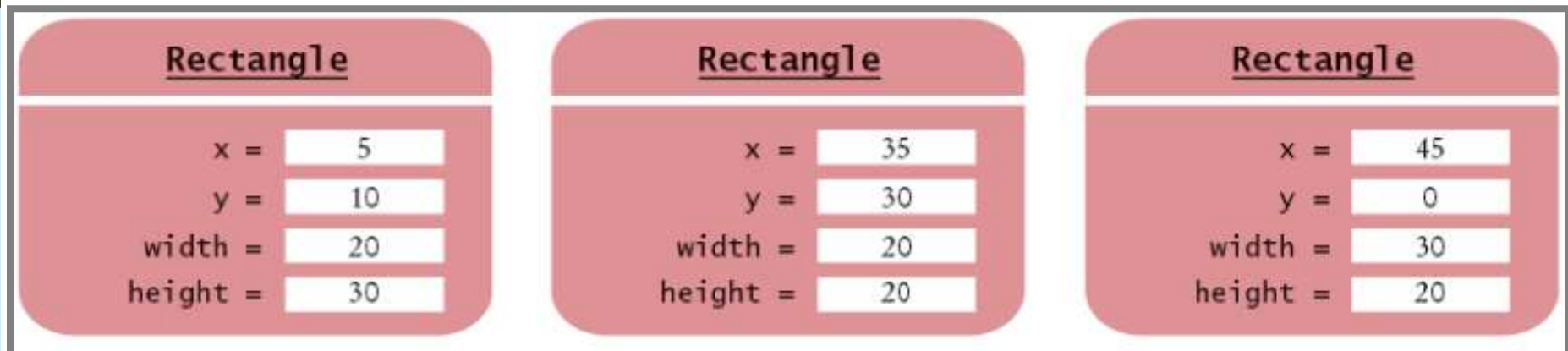
La classe Rectangle

- ▣ Descrive figure rettangolari
- ▣ `java.awt`

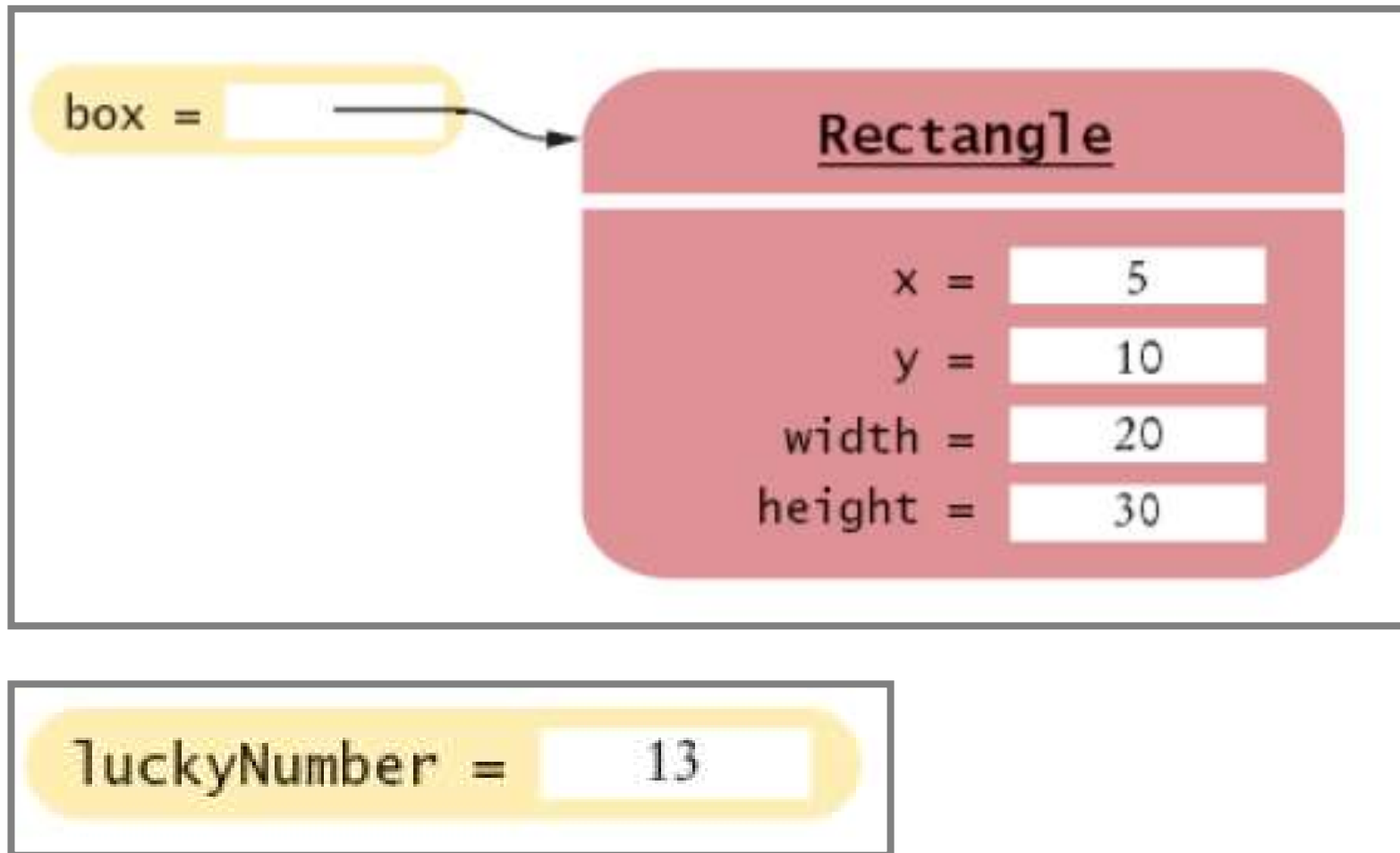


La classe Rectangle

▣ Rappresentazione interna

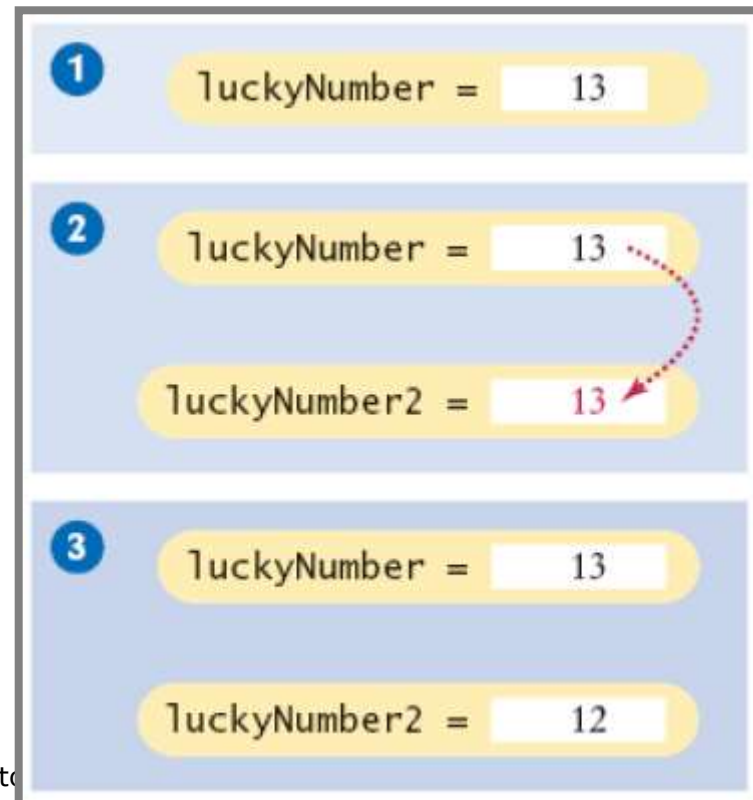


Variabili di riferimento e variabili “primitive”



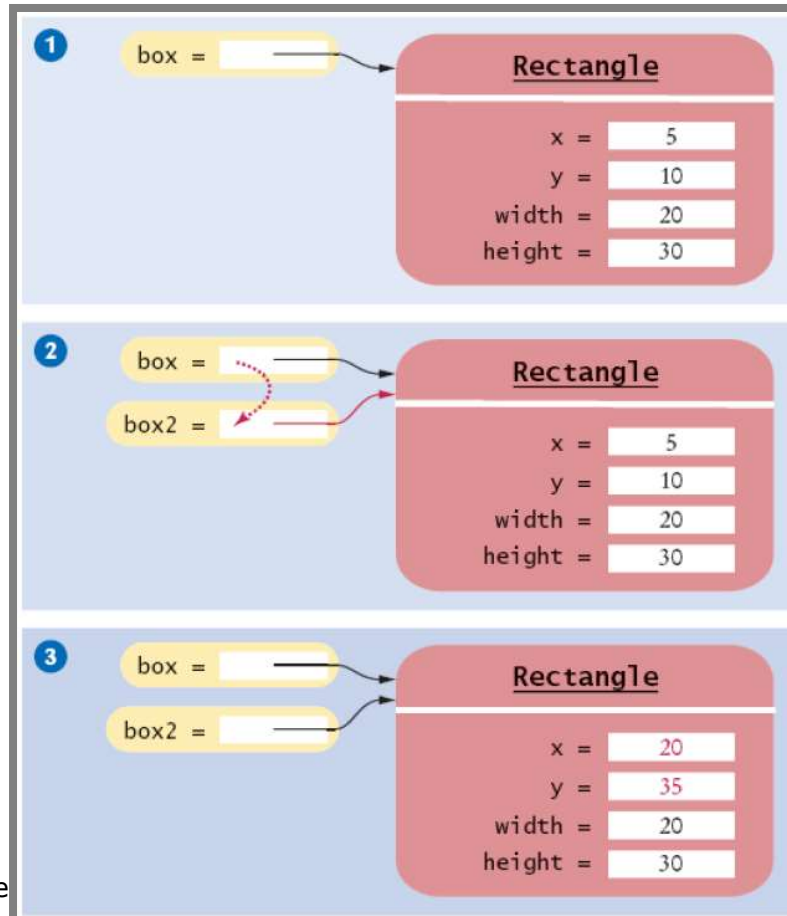
Copiare un numero

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



Copiare un riferimento ad oggetto

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



Creiamo un rettangolo

```
new Rectangle(5, 10, 20, 30)
```

- ❑ L'operatore new costruisce un nuovo oggetto di tipo Rectangle
- ❑ I parametri servono ad inizializzare lo stato
- ❑ Restituisce un riferimento all'oggetto creato
 - tipicamente memorizzato in una variabile

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

- ❑ Il processo di creazione di un nuovo oggetto è detto **costruzione**

Costruzione dell'oggetto

- La costruzione di un oggetto comporta l'invocazione di un particolare metodo della classe, detto **costruttore**

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

- i costruttori possono essere overloaded

```
Rectangle box1 = new Rectangle(20, 30);
```

- Costruisce un Rectangle di larghezza uguale a 20 ed altezza uguale a 30

```
Rectangle nullBox = new Rectangle();
```

- Costruisce un Rectangle di larghezza uguale a 0 ed altezza uguale a 0 ed il cui punto in alto a sinistra ha coordinate (0,0)

Creare oggetti: costruttori

□ Costruttori

- Metodi speciali presenti in ogni classe
- Ogni classe ha uno o più costruttori
- Hanno lo stesso nome della classe
- Sono invocati per creare un nuovo oggetto utilizzando l'operatore new
- Restituiscono una referenza al nuovo oggetto creato

- la creazione di un oggetto della classe X comporta l'invocazione di uno dei costruttori di X

Nota sintattica

`new ClassName(parameters)`

Esempio:

`new Rectangle(5, 10, 20, 30)`

`new Rectangle()`

Costruisce un nuovo oggetto e ne inizializza lo stato, restituendo un riferimento all'oggetto

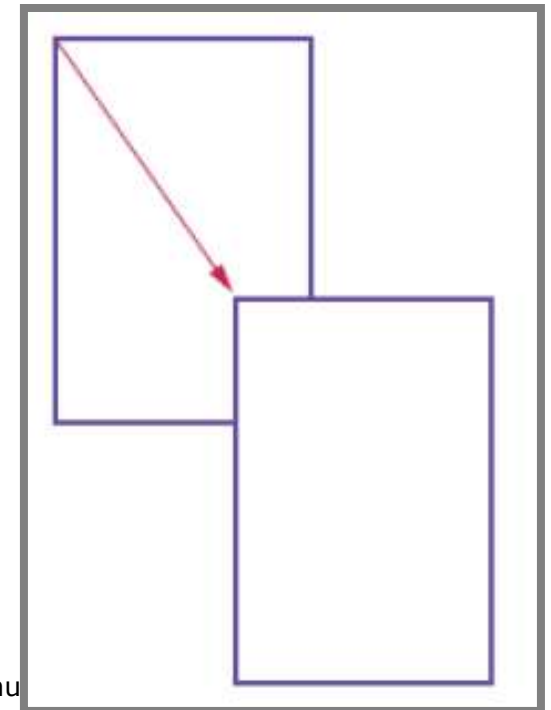
Metodi di accesso e mutatori

- Un metodo *Accessorio* non modifica lo stato del parametro implicito:

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

- Un metodo *Mutatore* modifica lo stato del suo parametro implicito

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



Importare le classi in un package

- Un package è un gruppo di classi Java

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

- Non è necessario importare del classi del package java.lang

```
import packageName.ClassName;
```

Esempio:

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

Importa una classe da un package

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.println("After moving, ")
14:         System.out.println("the top-left corner is:");
15:         System.out.println(box.getX());
16:         System.out.println(box.getY());
17:     }
18: }
```


Esercizio 1

- Scrivere un programma che crea due oggetti della classe Rectangle, e stabilisce quale dei due ha l'area maggiore, quale il perimetro maggiore e qual è posizionato più in alto all'interno di un sistema di assi cartesiani ortogonali

Input

- ▣ Descrive una sequenza di caratteri entranti
 - java.util

```
Scanner in = new Scanner(System.in);  
System.out.print("Enter quantity: ");  
int quantity = in.nextInt();
```

La classe Scanner (da Java 5)

- ▣ `next` legge un sequenza di caratteri
- ▣ `nextDouble` legge un double
- ▣ `nextLine` legge una linea (fino all'Enter)
- ▣ `nextWord` legge una parola (fino ad uno spazio)

Esercizio 2

- Scrivere un programma che crea una sequenza di oggetti della classe Rectangle, leggendo dimensione e posizione da tastiera, ed individua l'oggetto della classe con l'area maggiore, quello con il perimetro maggiore e quello posizionato più in alto su un sistema di assi cartesiani ortogonali.

Vengono creati oggetti della classe Rectangle fintanto che non vengano dati in input i valori relativi ad un rettangolo avente l'area uguale a 0 o posizionato nell'origine del sistema di assi.

Esempio 1

- ▣ Descrivere cosa fa il seguente programma; compilare ed eseguire.

```
import javax.swing.JOptionPane;

public class Ese1
{
    public static void main(String[] args)
    {
        JOptionPane.showMessageDialog(null, "Salve, mondo!");
        System.exit(0);
    }
}
```

Esempio 2

- ▣ Descrivere cosa fa il seguente programma; compilare ed eseguire.

```
import javax.swing.JOptionPane;

public class Ese2
{
    public static void main(String[] args)
    {
        String name =
            JOptionPane.showInputDialog("Come ti chiami?");
        System.out.print("Salve, ");
        System.out.print(name);
        System.out.println("!");
        System.exit(0);
    }
}
```

Esercizi

- Scrivere un programma che visualizzi in sequenza due panel per chiedere il nome di una persona e darle il benvenuto
- Scrivere un programma che calcola l'intersezione di due rettangoli
- Dopo aver analizzato la classe Random, scrivere un programma che simuli il lancio di una moneta. Scrivere poi un programma per simulare il lancio di un dado
- Dopo aver analizzato la classe StringTokenizer scrivere un programma che conti i token in una frase e li stampi uno per riga

La classe Date

- Definita nel package `java.util`
- Modella una data includendo:
 - anno, mese, giorno, ora, minuti, secondi
 - non tutti i dati servono e sono presenti nella String che rappresenta un oggetto Date
 - Esempio: `16/02/2016`
- Bisogna definire il formato della Date
 - Class `SimpleDateFormat` del package `java.util`

La classe SimpleDateFormat

- Modella il formato di una data
 - Appartiene al package java.util
 - Un costruttore con un argomento String.
 - L'argomento rappresenta il formato dell'oggetto String che rappresenta una data. Esempio: "dd/MM/yy"
 - Per definire un oggetto Date, bisogna definire un oggetto che ne descrive il formato:

```
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
```

- La costruzione dell'oggetto Date usa l'oggetto referenziato da sdf:

```
Date data = sdf.format("16/02/2016");
```