

Capitolo 2

Utilizzare oggetti

Obiettivi del capitolo

- **Imparare a utilizzare variabili**
- **Capire i concetti di classe e oggetto**
- **Saper invocare metodi**
- **Usare parametri e valori restituiti dai metodi**
- **Realizzare programmi di collaudo**
- **Essere in grado di consultare la documentazione dell' API di Java**
- **Capire la differenza tra oggetti e riferimenti a oggetti**
- **Scrivere programmi che visualizzano semplici forme grafiche**

Tipi e variabili

- Ogni valore è di un determinato tipo
- Esempi di dichiarazione di variabili:

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

- Variabili
 - Memorizzano valori
 - Possono essere utilizzate al posto degli oggetti che memorizzano

Sintassi 2.1: Definizione di variabile

nomeTipo nomeVariabile = valore;
oppure
nomeTipo nomeVariabile;

Esempio:

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

Serve a:

Definire una nuova variabile di tipo *nomeTipo*
e fornirne eventualmente un *valore* iniziale.

Identificatori

- **Identificatore**: nome di una variabile, di un metodo o di una classe
- Regole per gli identificatori in Java:
 - Possono essere composti di lettere, cifre, caratteri “dollaro” (\$) e segni di sottolineatura (_)
 - non possono iniziare con una cifra
 - non si possono usare altri simboli, come ? o %.
 - gli spazi non sono ammessi all’ interno degli identificatori
 - le parole riservate non possono essere usate come identificatori
 - sono sensibili alla differenza tra lettere maiuscole e minuscole

Continua

Identificatori

- Per convenzione, i nomi delle variabili dovrebbero iniziare con una lettera minuscola.
- Per convenzione, i nomi delle classi dovrebbero iniziare con una lettera maiuscola.

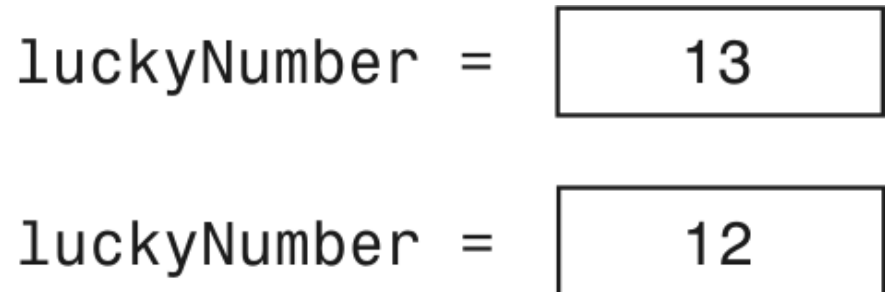
L'operatore di assegnazione

- Operatore di assegnazione (=)
- Non significa “uguaglia” ma “diventa”
- Utilizzato per modificare il valore di una variabile

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

Figura 1

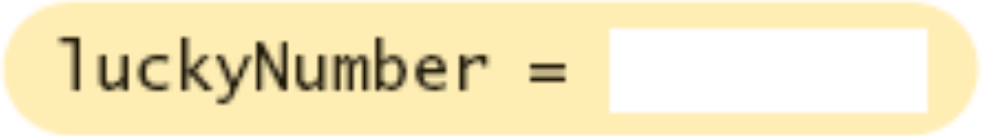
Assegnazione di un nuovo valore a una variabile



Variabile non inizializzata

- Errore:

```
int luckyNumber;  
System.out.println(luckyNumber);  
    // ERRORE - variabile priva di valore
```



luckyNumber =

Figura 2

Una variabile non inizializzata

Sintassi 2.2: Assegnazione

nomeVariabile = valore;

Esempio:

`luckyNumber = 12;`

Serve a:

Assegnare un valore a una variabile definita in precedenza.

Oggetti e classi

- Gli oggetti sono entità di un programma che si possono manipolare invocando metodi.
- Tali oggetti appartengono a diverse classi. Per esempio l'oggetto *System.out* appartiene alla classe *PrintStream*.

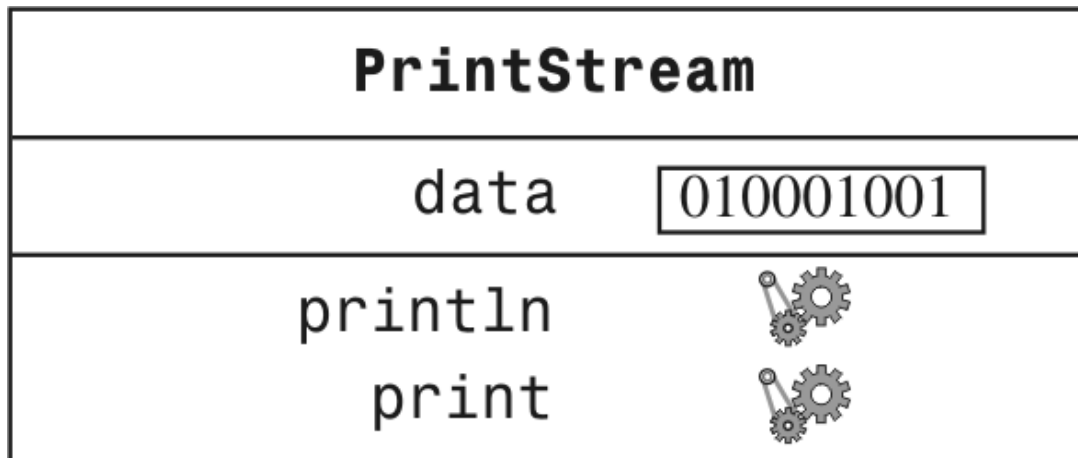


Figura 3:
Rappresentazione
dell'oggetto `System.out`

Metodi

- Metodo: sequenza di istruzioni che accede ai dati di un oggetto
- Gli oggetti possono essere manipolati invocando metodi
- Classe: insieme di oggetti con lo stesso comportamento
- Una classe specifica i metodi che possono essere applicati ai suoi oggetti

```
String greeting = "Hello";  
greeting.println() // Error  
greeting.length() // OK
```

- L'interfaccia pubblica di una classe specifica *cosa* si può fare con i suoi oggetti mentre l'implementazione nascosta descrive *come* si svolgono tali azioni.

Rappresentazione di due oggetti di tipo `String`

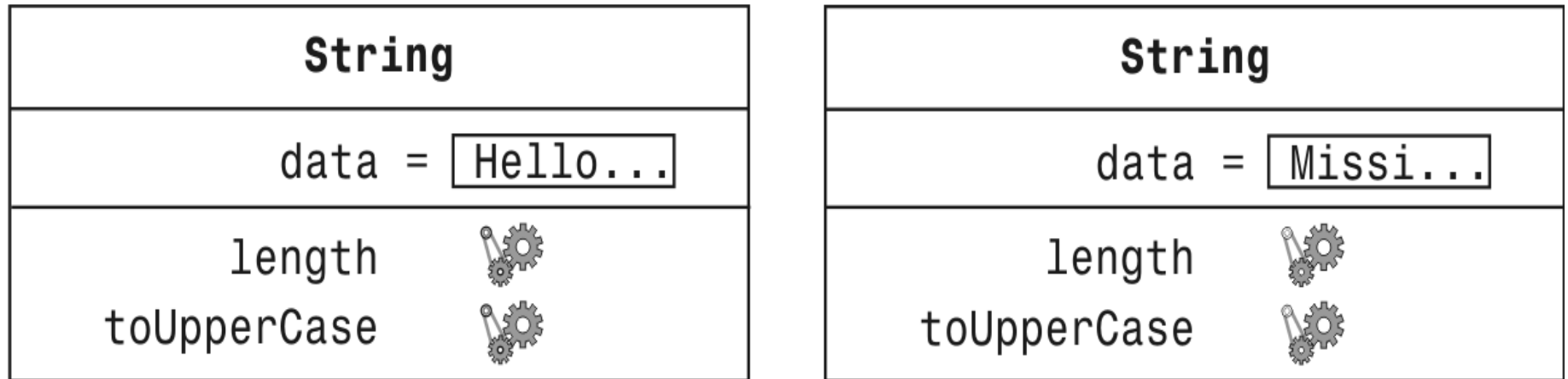


Figura 4

Rappresentazione di due oggetti di tipo `String`

Metodi String

- `length`: conta il numero di caratteri presenti in una stringa.

```
String greeting = "Hello, World!";  
int n = greeting.length(); // assegna a n il numero 13
```

Continua...

Metodi String

- `toUpperCase`: crea un nuovo oggetto di tipo `String` che contiene gli stessi caratteri dell'oggetto originale, con le lettere minuscole convertite in maiuscole.

```
String river = "Mississippi";  
String bigRiver = river.toUpperCase();  
// assegna a bigRiver l'oggetto "MISSISSIPPI"
```

Continua...

Metodi String

- Quando applicate un metodo a un oggetto, dovete essere certi che il metodo sia definito nella classe corrispondente.

```
System.out.length(); // Questa invocazione di metodo è errata
```

Parametri impliciti ed espliciti

- Parametro (parametro esplicito): dati in ingresso a un metodo. Non tutti i metodi necessitano di parametri.

```
System.out.println(greeting)  
greeting.length() // non ha parametri espliciti
```

- Parametro implicito: l'oggetto di cui si invoca un metodo

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

Continua...

Parametri impliciti ed espliciti

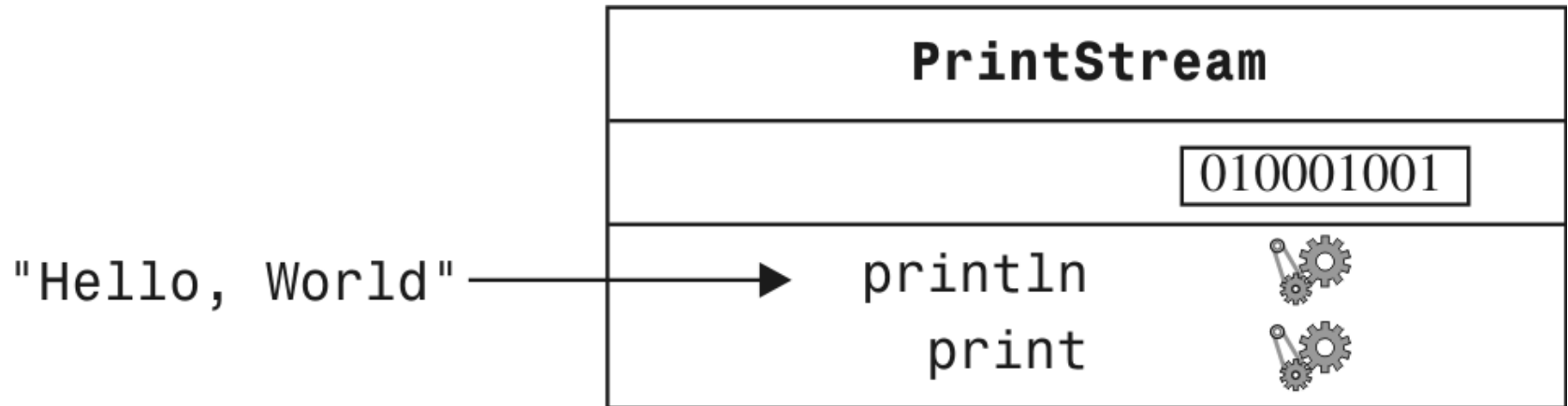


Figura 5

Passaggio di parametro al metodo `println`

Valori restituiti

- Il **valore restituito** da un metodo è il risultato che il metodo ha calcolato perché questo venga utilizzato nel codice che ha invocato il metodo

```
int n = greeting.length(); // restituisce il valore  
                             // memorizzato in n
```

Continua

Valori restituiti

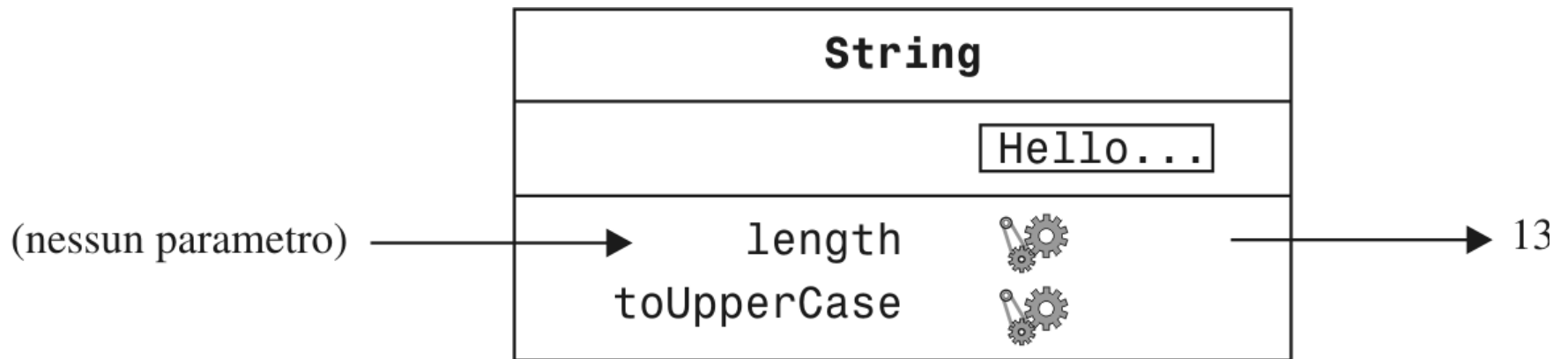


Figura 6 Invocazione del metodo `length` su un oggetto di tipo `String`

Utilizzo dei valori restituiti

- Il valore restituito da un metodo può anche essere utilizzato direttamente come parametro di un altro metodo

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

- Non tutti i metodi restituiscono valori.
Per esempio:

```
println
```

Continua...

Utilizzo dei valori restituiti

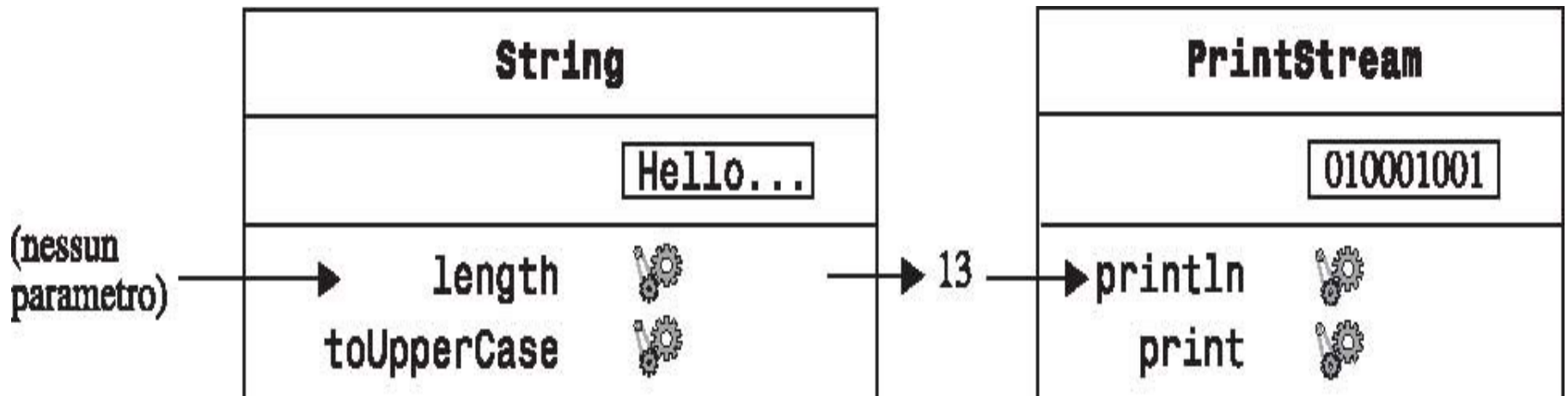


Figura 7

Il valore restituito da un metodo utilizzato come parametro di un altro metodo

Una invocazione più complessa

- Il metodo `replace` esegue metodi di ricerca e sostituzione

```
river.replace("issipp", "our")  
// costruisce una nuova stringa ("Missouri")
```

- Come si vede nella Figura 8, questa invocazione di metodo ha
 - un parametro implicito: la stringa “Mississippi”
 - due parametri espliciti: le stringhe “issipp” e “our”
 - un valore restituito: la stringa “Missouri”

Continua...

Una invocazione più complessa

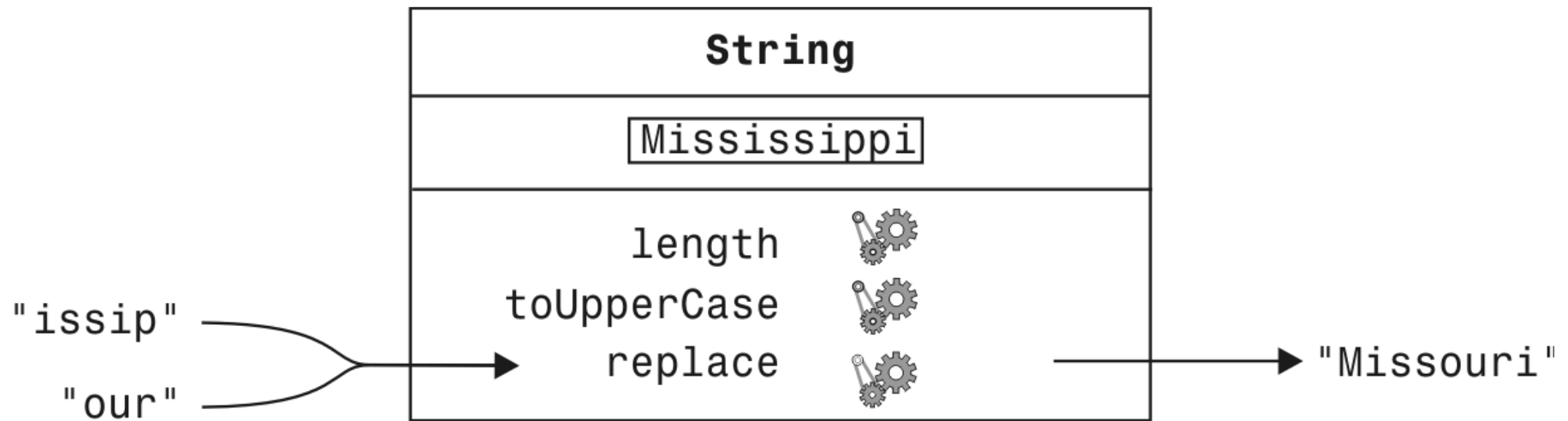


Figura 8 Invocazione del metodo `replace`

Definizioni di metodo

- Quando in una classe si definisce un metodo, vengono specificati i tipi dei parametri espliciti e del valore restituito.
- Il tipo del parametro implicito è la classe in cui è definito il metodo: ciò non viene menzionato nella definizione del metodo, e proprio per questo si parla di parametro “implicito”.

Continua...

Definizioni di metodo

- Esempio: la classe `String` definisce

```
public int length()  
    // restituisce un valore di tipo int  
    // non ha parametri espliciti  
  
public String replace(String target, String replacement)  
    // restituisce un valore di tipo String;  
    // due parametri espliciti di tipo String
```

Continua...

Definizioni di metodo

- Se il metodo non restituisce un valore, il tipo di valore restituito viene dichiarato come *void*

```
public void println(String output) // nella classe PrintStream
```

- Il nome di un metodo è sovraccarico se una classe definisce più metodi con lo stesso nome (ma con parametri di tipi diversi).

```
public void println(String output)  
public void println(int output)
```

Tipi numerici

- Numeri interi `short`, `int`, `long`
13

- Numeri in virgola mobile
1.3
0.00013

Continua...

Tipi numerici

- Quando un numero in virgola mobile viene moltiplicato o diviso per 10, si modifica solamente la posizione del separatore decimale, che diviene così “mobile”.

```
1.3E-4      // 1.3 × 10-4 in Java
```

- In Java, i numeri non sono oggetti e i tipi numerici non sono classi; i tipi numerici sono tipi primitivi, non classi.

Operazioni aritmetiche

- Operatori: + - *

```
10 + n  
n - 1  
10 * n          // 10 × n
```

Come avviene in matematica, l'operatore * ha la precedenza rispetto all'operatore +

```
x + y * 2      // rappresenta la somma di x e y * 2  
(x + y) * 2    // moltiplica la somma di x e y per 2
```

Forme rettangolari e oggetti Rectangle

- Gli oggetti di tipo `Rectangle` *descrivono* forme rettangolari

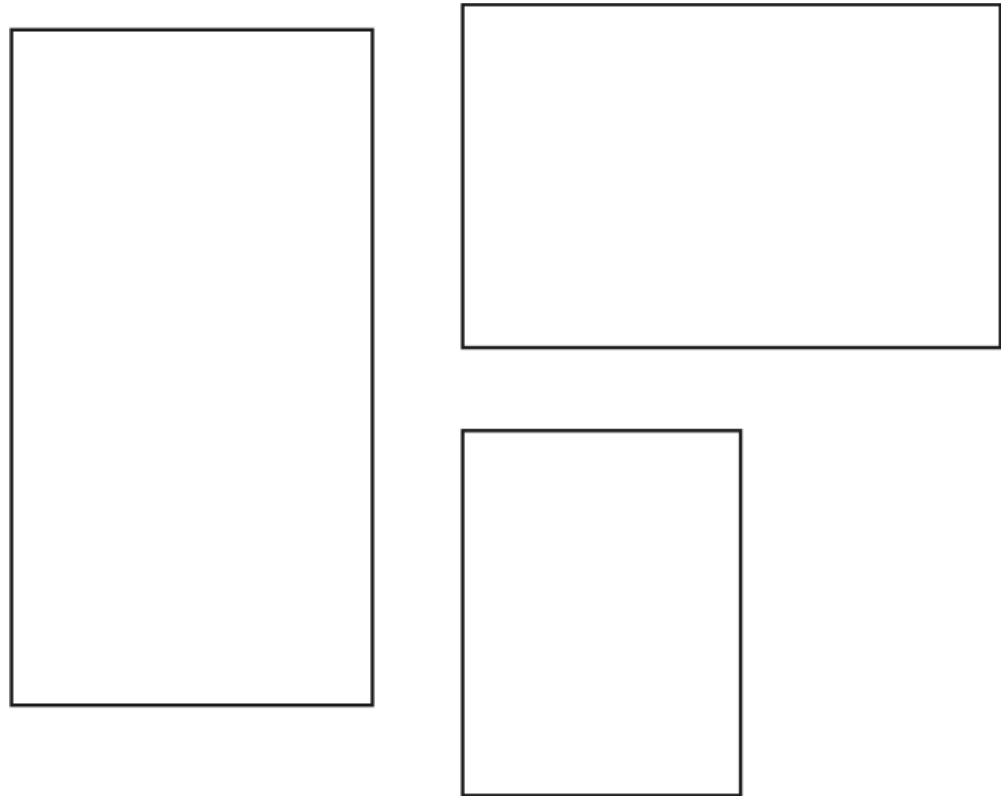


Figura 9
Forme rettangolari

Forme rettangolari e oggetti Rectangle

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

Rectangle	
x =	5
y =	10
width =	20
height =	30

Rectangle	
x =	35
y =	30
width =	20
height =	20

Rectangle	
x =	45
y =	0
width =	30
height =	20

Figura 10 Oggetti di tipo `Rectangle`

Costruzione di oggetti

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

- Dettaglio:
 1. L'operatore `new` costruisce un oggetto di tipo `Rectangle`.
 2. Nel fare ciò, usa i parametri ricevuti (in questo caso, 5, 10, 20 e 30) per assegnare valori iniziali ai dati dell'oggetto.
 3. Restituisce l'oggetto.
- Solitamente l'oggetto creato dall'operatore `new` viene memorizzato in una variabile, in questo modo:

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


Costruzione di oggetti

- Il processo che crea un nuovo oggetto è detto *costruzione*.
- I quattro valori 5, 10, 20 e 30 rappresentano i *parametri di costruzione*.
- Alcune classi permettono di costruire oggetti in più modi diversi.

```
new Rectangle()  
    // costruisce un rettangolo con il vertice superiore  
    // sinistro posizionato all'origine (0, 0),  
    // con larghezza 0, e altezza 0
```

Sintassi 2.3: costruzione di oggetti

```
new NomeClasse(parametri)
```

Esempio:

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

```
new Rectangle()
```

Serve a:

Costruire un nuovo oggetto, inizializzarlo tramite i parametri di costruzione e restituire un riferimento all'oggetto costruito.

Metodi di accesso e metodi modificatori

- Un metodo che accede a un oggetto e restituisce alcune informazioni a esso relative, senza modificare l'oggetto stesso, viene chiamato metodo *d'accesso*.

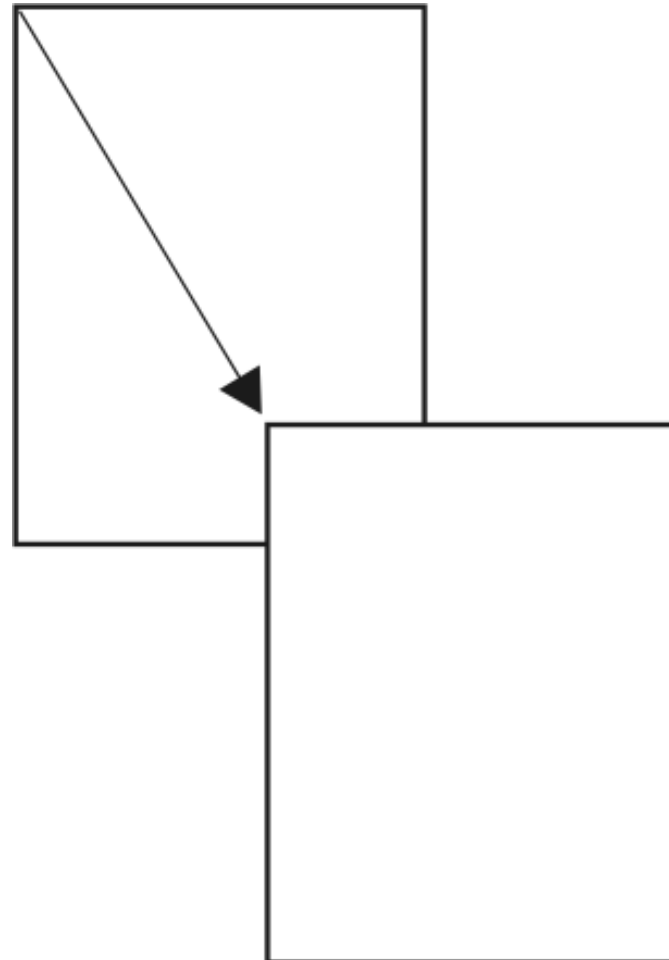
```
double width = box.getWidth();
```

- Un metodo che abbia lo scopo di modificare lo stato di un oggetto viene chiamato metodo *modificatore*.

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

Metodi di accesso e metodi modificatori

Figura 11
Uso del metodo `translate`
per spostare un rettangolo



Realizzare un programma di collaudo

Il programma di collaudo esegue i seguenti passi:

- Definisce una nuova classe.
- Definisce in essa il metodo `main`.
- Costruisce uno o più oggetti all'interno del metodo `main`.
- Applica metodi agli oggetti.
- Visualizza i risultati delle invocazioni dei metodi.
- Visualizza i valori previsti.

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:         // sposta il rettangolo
10:         box.translate(15, 25);
11:
12:         // visualizza informazioni sul rettangolo traslato
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: }
```

ch02/rectangle/MoveTester.java (cont.)

Visualizza:

x: 20

Expected: 20

y: 35

Expected: 35

Importare "pacchetti"

Ricordarsi di importare i pacchetti appropriati:

- Tutte le classi della libreria standard sono contenute all'interno di pacchetti (packages)
- Importate le classi della libreria standard specificando il nome del pacchetto e della classe:

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

- le classi `System` e `String` si trovano nel pacchetto `java.lang`, le cui classi vengono importate automaticamente, in modo che non ci sia mai bisogno di importarle in modo esplicito.

Sintassi 2.4: Importazione di una classe da un pacchetto

```
import nomePacchetto.NomeClasse;
```

Esempio:

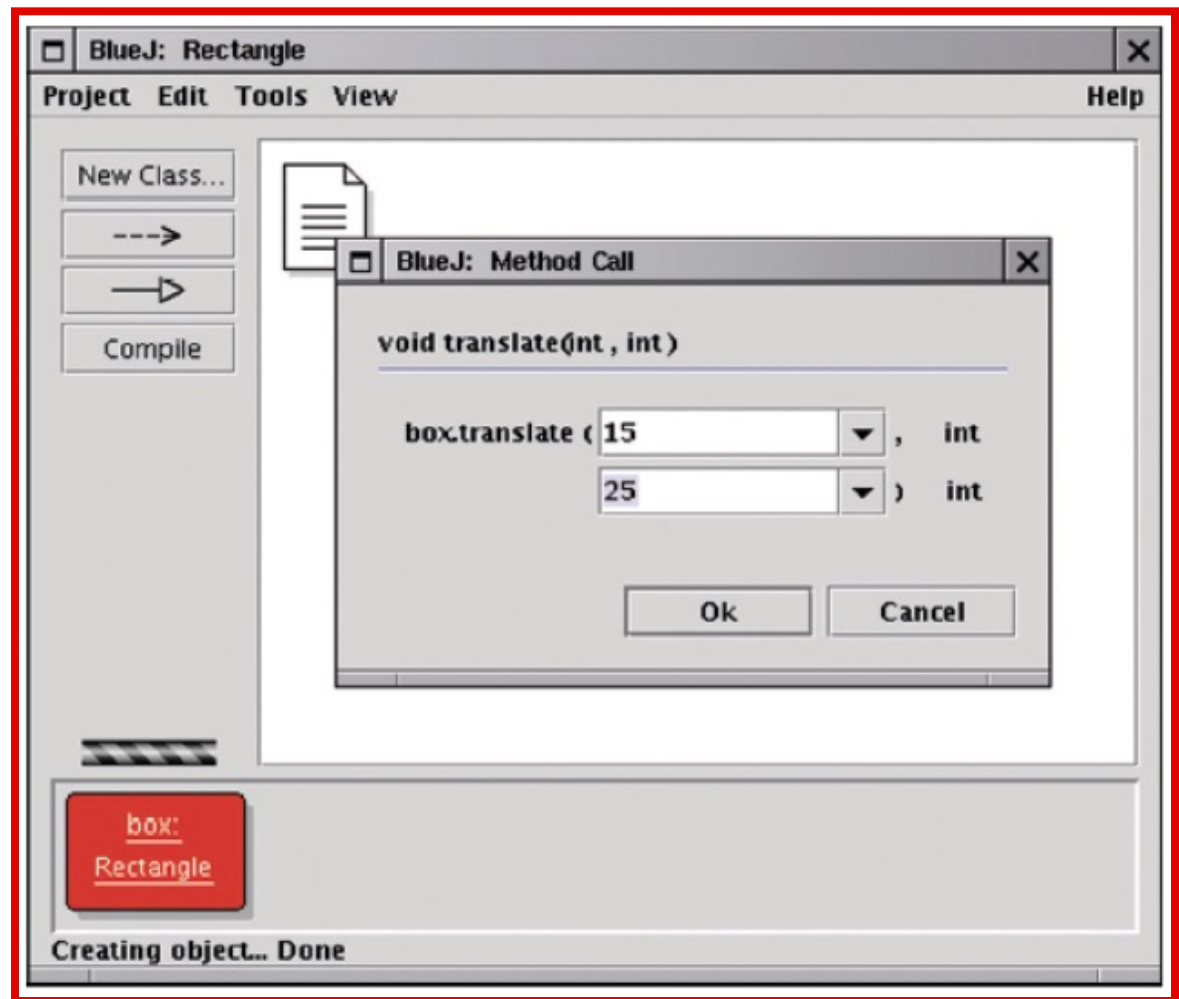
```
import java.awt.Rectangle;
```

Serve a:

Importare una classe da un pacchetto per utilizzarla in un programma.

Collaudare una classe in un ambiente interattivo

Figura 12
Collaudo
dell'invocazione
di un metodo con BlueJ



La documentazione API

- API: Application Programming Interface
(interfaccia per la pubblicazione di applicazioni)
- Elenca le classi e i metodi della libreria Java
- <http://java.sun.com/javase/6/docs/api/index.html>

La documentazione API per la libreria standard di Java

The screenshot shows a Microsoft Internet Explorer window titled "Overview (Java 2 Platform SE 5.0) - Microsoft Internet Explorer". The address bar shows the URL "http://java.sun.com/j2se/1.5.0/docs/api/index.html". The page content includes a navigation bar with links: Overview, Package, Class, Use, Tree, Deprecated, Index, Help. Below this, the main heading is "Java™ 2 Platform Standard Edition 5.0 API Specification". A paragraph states: "This document is the API specification for the Java 2 Platform Standard Edition 5.0." Below this, a section titled "See:" contains a link to "Description". A table titled "Java 2 Platform Packages" lists several packages and their descriptions.

Java 2 Platform Packages	
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between applications.

La documentazione API per la classe Rectangle

The screenshot shows a web browser window titled "Rectangle (Java 2 Platform SE 5.0) - Microsoft Internet Explorer". The address bar shows the URL "http://java.sun.com/j2se/1.5.0/docs/api/index.html". The browser's menu bar includes "File", "Modifica", "Visualizza", "Preferiti", "Strumenti", and "?".

On the left side, there is a "Packages" list with links to various Java packages, including [java.applet](#), [java.awt](#), [java.awt.color](#), [java.awt.datatransfer](#), [java.awt.dnd](#), [java.awt.event](#), [java.awt.font](#), and [java.awt.geom](#). Below this, a list of classes is shown, with [Rectangle](#) highlighted.

The main content area displays the "Class Rectangle" page. At the top, there are navigation tabs: "Overview", "Package", "Class" (selected), "Use", "Tree", "Deprecated", "Index", and "Help". Below these tabs, there are links for "PREV CLASS", "NEXT CLASS", "FRAMES", and "NO FRAMES". The "SUMMARY" section lists "NESTED", "FIELD", "CONSTR", and "METHOD". The "DETAIL" section lists "FIELD", "CONSTR", and "METHOD".

The class hierarchy is shown as follows:

```
java.lang.Object
├── java.awt.geom.RectangularShape
│   └── java.awt.geom.Rectangle2D
│       └── java.awt.Rectangle
```

Below the hierarchy, the "All Implemented Interfaces:" section lists [Shape](#), [Serializable](#), and [Cloneable](#). The "Direct Known Subclasses:" section lists [DefaultCaret](#).

The class declaration is shown as:

```
public class Rectangle
extends Rectangle2D
implements Shape, Serializable
```

The bottom of the browser window shows the "Internet" icon and the "Vai" button.

L'elenco riassuntivo dei metodi della classe Rectangle

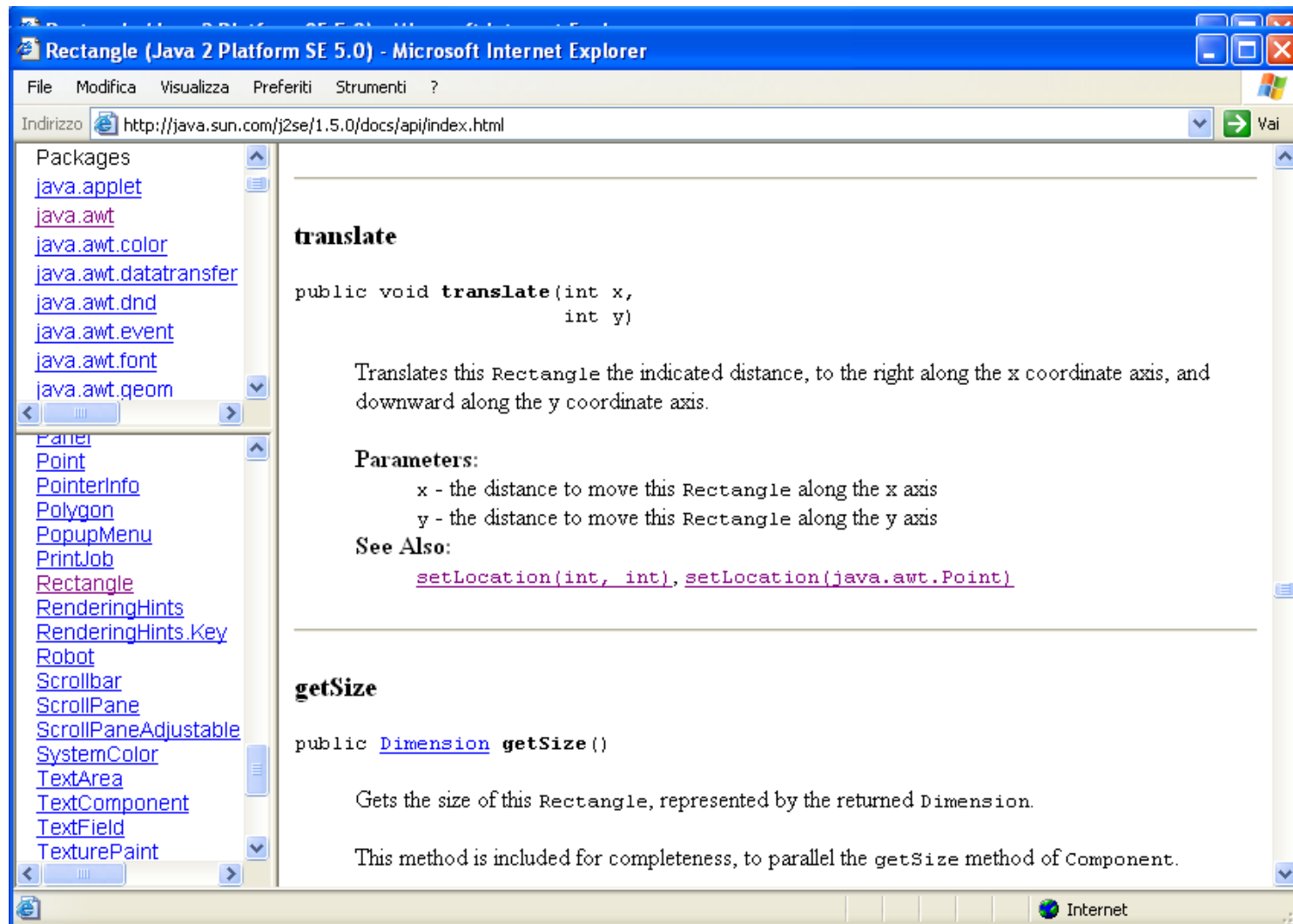
The screenshot shows a web browser window titled "Rectangle (Java 2 Platform SE 5.0) - Microsoft Internet Explorer". The address bar shows the URL "http://java.sun.com/j2se/1.5.0/docs/api/index.html". The left sidebar contains a list of packages and classes, with "Rectangle" selected. The main content area displays the "Method Summary" for the Rectangle class.

Constructs a new Rectangle, initialized to match the values of the specified Rectangle.

Method Summary

void	add (int newx, int newy)	Adds a point, specified by the integer arguments newx and newy, to this Rectangle.
void	add (Point pt)	Adds the specified Point to this Rectangle.
void	add (Rectangle r)	Adds a Rectangle to this Rectangle.
boolean	contains (int x, int y)	Checks whether or not this Rectangle contains the point at the specified location (x,y).
boolean	contains (int X, int Y, int W, int H)	Checks whether this Rectangle entirely contains the Rectangle at the specified location (X, Y) with the specified dimensions (W, H).
boolean	contains (Point p)	Checks whether or not this Rectangle contains the specified Point.
boolean	contains (Rectangle r)	Checks whether or not this Rectangle entirely contains the specified Rectangle.
Rectangle2D	createIntersection (Rectangle2D r)	

La documentazione API del metodo `translate`



Riferimenti a oggetti

- Un riferimento a un oggetto descrive la posizione dell'oggetto in memoria.
- L'operatore `new` restituisce un riferimento a un nuovo oggetto

```
Rectangle box = new Rectangle();
```

- Più variabili oggetto possono contenere riferimenti al medesimo oggetto.

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

- Le variabili numeriche memorizzano numeri, mentre le variabili oggetto memorizzano riferimenti.

Variabili oggetto e variabili numeriche

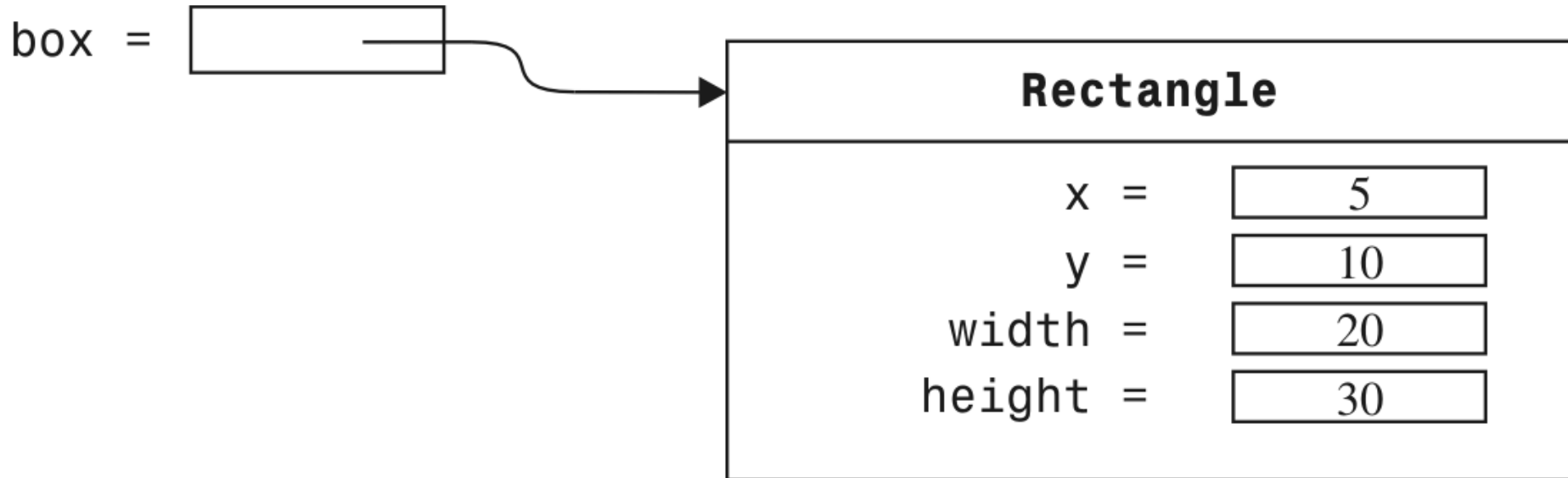


Figure 17

Una variabile oggetto contenente un riferimento a un oggetto

Variabili oggetto e variabili numeriche

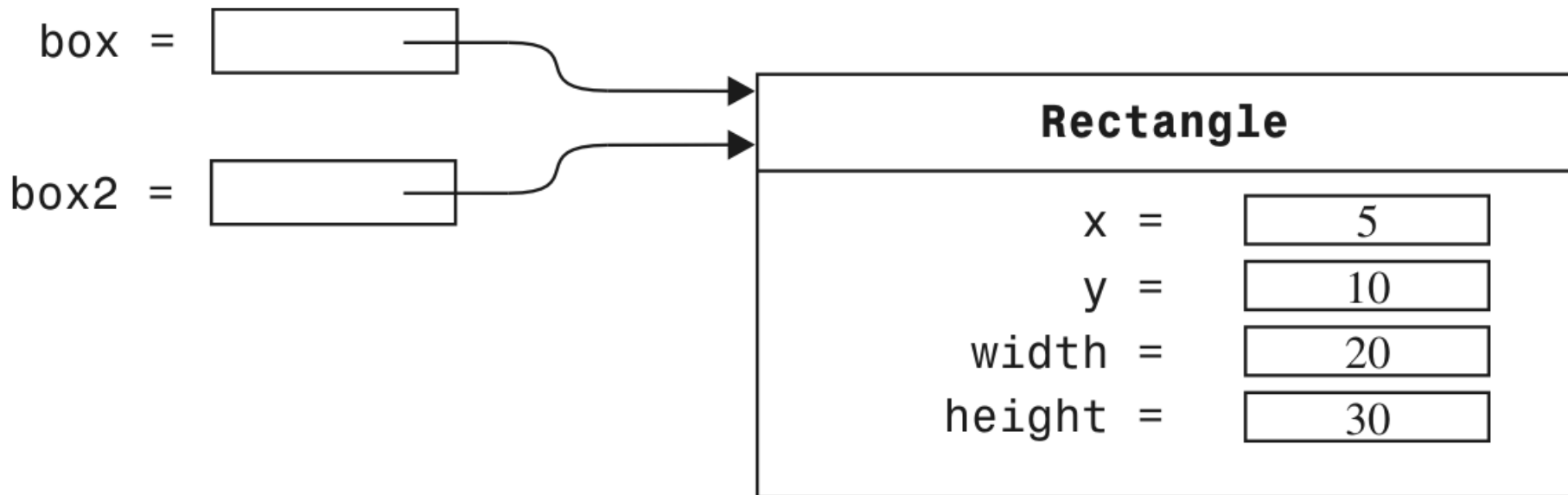


Figura 18

Due variabili oggetto che fanno riferimento al medesimo oggetto

Variabili oggetto e variabili numeriche


luckyNumber = 

Figura 19

Una variabile di tipo numerico memorizza un numero

Copiatura di numeri

■

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

luckyNumber =

13

luckyNumber =

13

luckyNumber2 =

13



luckyNumber =

13

luckyNumber2 =

12

Figura 20

Copiatura di numeri

Copiatura di riferimenti a oggetti

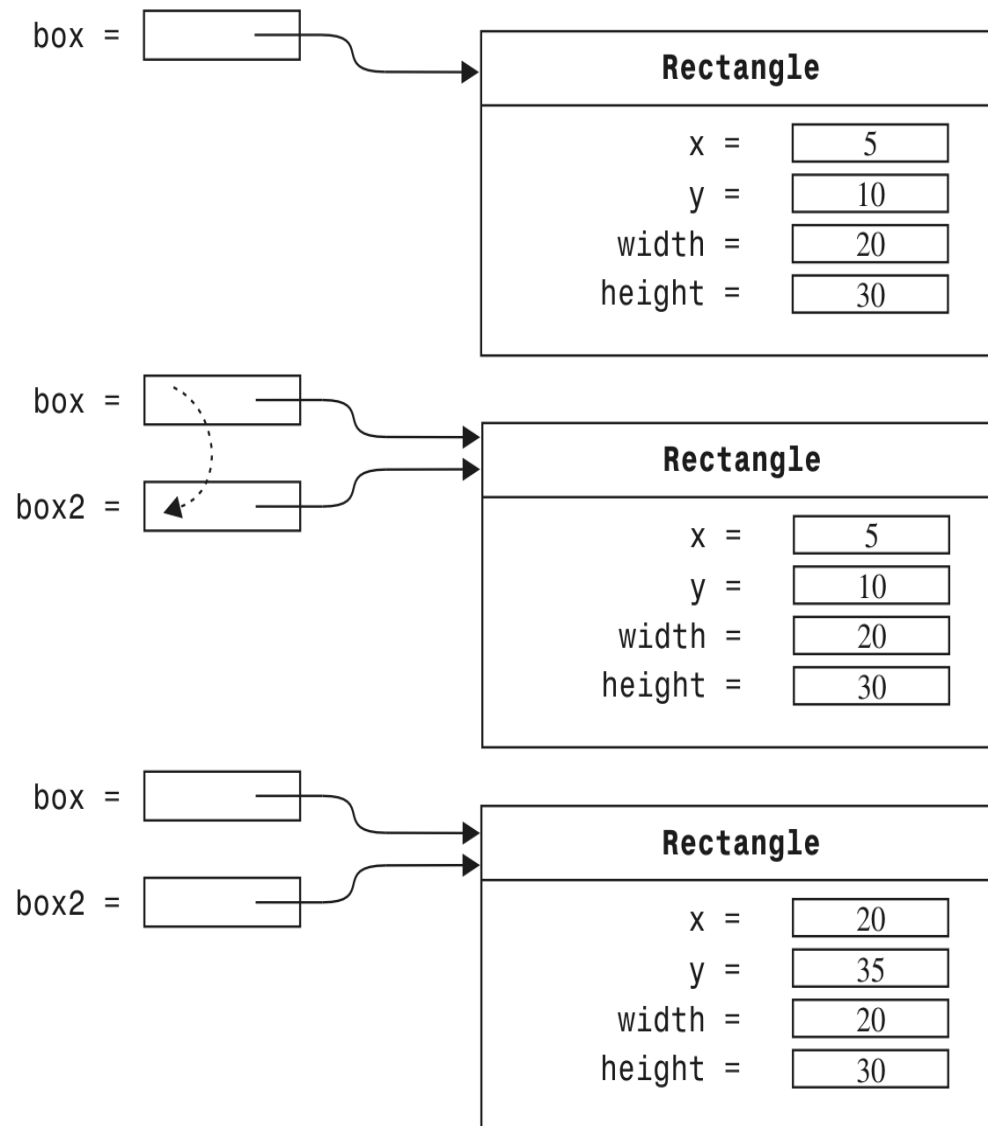


```
Rectangle box = new Rectangle(5, 10, 20, 30);  
Rectangle box2 = box;  
// situazione rappresentata nella figura 21  
box2.translate(15, 25);
```

Continua...

Copiatura di riferimenti a oggetti

Figura 21
Copiatura di riferimenti a oggetti



I mainframe: quando i dinosauri dominavano la terra

Figura 22
Un computer mainframe



Applicazioni grafiche e finestre

Per visualizzare una finestra frame occorre:

1. Costruire un esemplare della classe JFrame:

```
JFrame frame = new JFrame();
```

2. Impostare la dimensione del frame:

```
frame.setSize(300, 400);
```

3. Se lo preferite, assegnare un titolo al frame:

```
frame.setTitle("An Empty Frame");
```

4. Impostare l' "operazione di chiusura predefinita":

```
frame.setDefaultCloseOperation(  
    JFrame.EXIT_ON_CLOSE);
```

5. Rendere visibile il frame:

```
frame.setVisible(true);
```


Una finestra di tipo frame

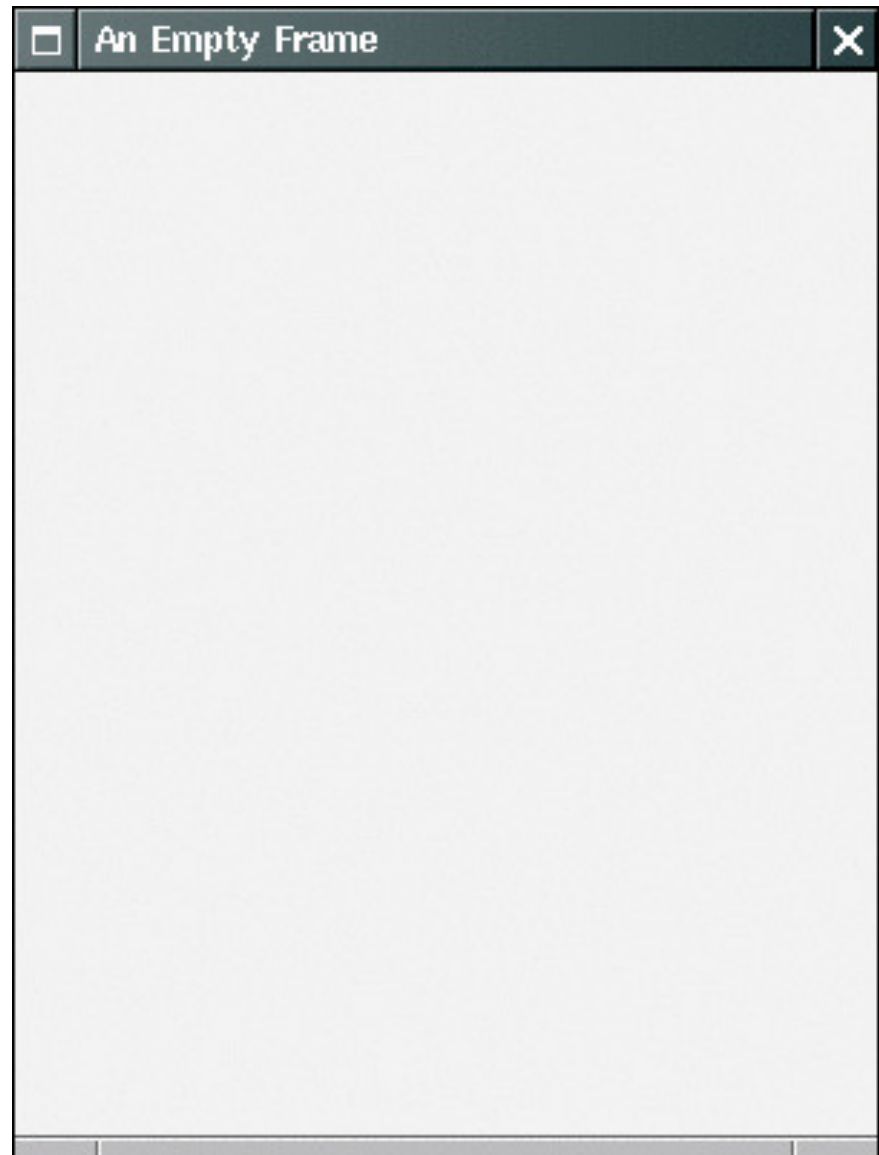


Figura 23:
Una finestra di tipo frame

ch02/emptyframe/EmptyFrameViewer.java

```
01: import javax.swing.JFrame;
02:
03: public class EmptyFrameViewer
04: {
05:     public static void main(String[] args)
06:     {
07:         JFrame frame = new JFrame();
08:
09:         frame.setSize(300, 400);
10:         frame.setTitle("An Empty Frame");
11:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:
13:         frame.setVisible(true);
14:     }
15: }
```

Disegnare in un componente

- Per visualizzare qualcosa in un frame, occorre definire una classe che estenda la classe *JComponent*.
- Inserite le istruzioni di disegno all'interno del metodo *paintComponent*, che viene invocato ogni volta che il componente deve essere ridisegnato.

```
public class RectangleComponent extends JComponent
{
    public void paintComponent(Graphics g)
    {
        Istruzioni per disegnare il componente
    }
}
```

Classi *Graphics* e *Graphics2D*

- La classe *Graphics* ci permette di manipolare lo stato grafico (come il colore attuale).
- La classe *Graphics2D* fornisce metodi che consentono di disegnare forme grafiche.
- Nel metodo *paintComponent*, usate un cast per recuperare l'oggetto *Graphics2D* a partire dal parametro di tipo *Graphics*:

```
public class RectangleComponent extends JComponent
{
    public void paintComponent(Graphics g)
    {
        // Recupera Graphics2D
        Graphics2D g2 = (Graphics2D) g;
        . . .
    }
}
```

Classi *Graphics* e *Graphics2D*

- Il metodo *draw* della classe *Graphics2D* è in grado di disegnare forme come rettangoli, ellissi, segmenti di retta, poligoni e archi.

```
public class RectangleComponent extends JComponent
{
    public void paintComponent(Graphics g)
    {
        . . .
        Rectangle box = new Rectangle(5, 10, 20, 30);
        g2.draw(box) ;
        . . .
    }
}
```

Disegnare rettangoli

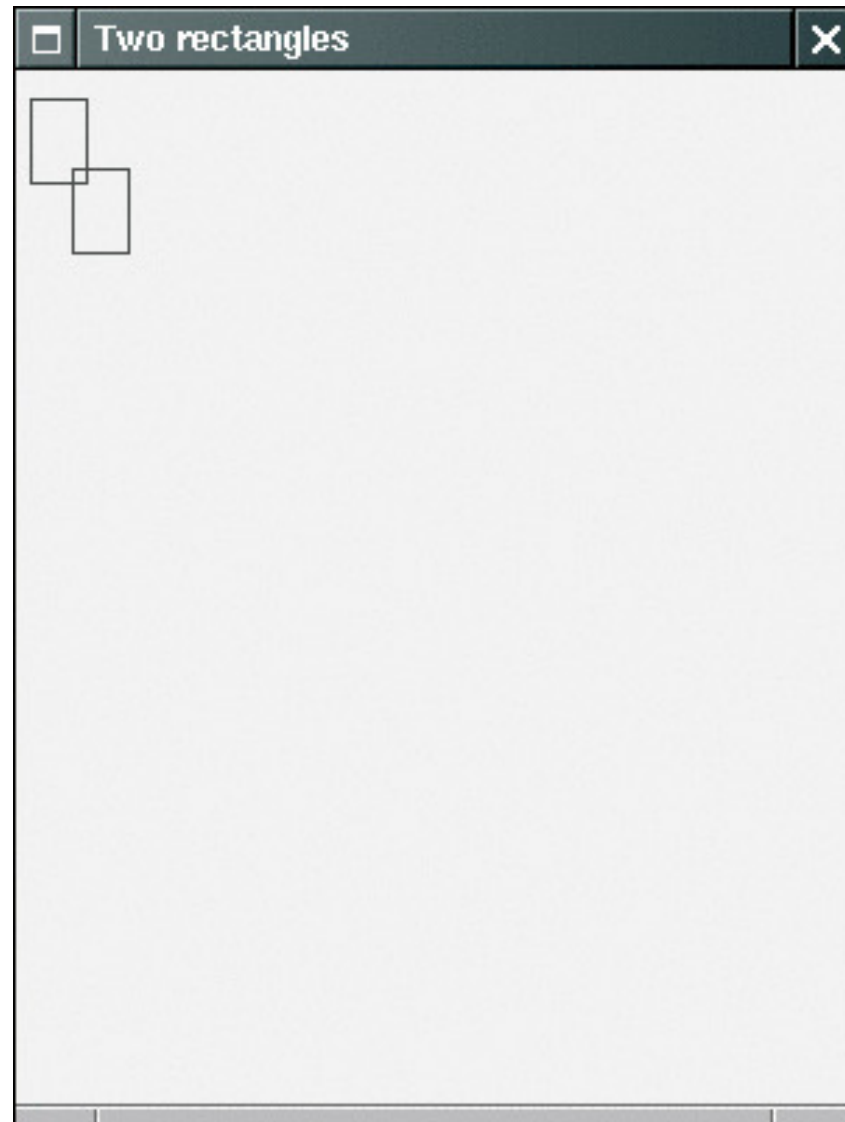


Figura 24:
Disegnare rettangoli

File RectangleComponent.java

```
01: import java.awt.Graphics;
02: import java.awt.Graphics2D;
03: import java.awt.Rectangle;
04: import javax.swing.JComponent;
05:
06: /**
07:     Un componente che disegna due rettangoli.
08: */
09: public class RectangleComponent extends JComponent
10: {
11:     public void paintComponent(Graphics g)
12:     {
13:         // Recupera Graphics2D
14:         Graphics2D g2 = (Graphics2D) g;
15:
16:         // Costruisce un rettangolo e lo disegna
17:         Rectangle box = new Rectangle(5, 10, 20, 30);
18:         g2.draw(box);
19:
```

continua

File RectangleComponent.java

```
20:         // Sposta il rettangolo di 15 unità verso destra e di
21:         // 25 unità verso il basso
22:         box.translate(15, 25);
23:
24:         // Disegna il rettangolo nella nuova posizione
25:         g2.draw(box);
26:     }
27: }
```


Usare un componente

- Costruite un frame
- Costruite un esemplare della vostra classe che descriva un componente

```
RectangleComponent component = new RectangleComponent();
```

- Aggiungete il componente al frame

```
frame.add(component);
```

- Rendete visibile il frame

File rectangleViewer.java

```
01: import javax.swing.JFrame;
02:
03: public class RectangleViewer
04: {
05:     public static void main(String[] args)
06:     {
07:         JFrame frame = new JFrame();
08:
09:         frame.setSize(300, 400);
10:         frame.setTitle("Two rectangles");
11:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12:
13:         RectangleComponent component = new RectangleComponent();
14:         frame.add(component);
15:
16:         frame.setVisible(true);
17:     }
18: }
```

Applet

- Gli applet sono programmi che vengono eseguiti all'interno di un browser web.
- Per realizzare un applet dovete usare codice che segua questo schema:

```
public class MyApplet extends JApplet
{
    public void paint(Graphics g)
    {
        // Recupera il riferimento a Graphics2D
        Graphics2D g2 = (Graphics2D) g;
        // Istruzioni per disegnare
        . . .
    }
}
```

Applet

- Lo schema è molto simile a quello di un componente, con due differenze di poco conto:
 1. Si deve estendere JApplet e non JComponent
 2. Le istruzioni che tracciano il disegno devono essere inserite nel metodo *paint* e non nel metodo *paintComponent*
- Per eseguire un applet occorre un file HTML che contenga un marcatore *applet*
- Un file HTML può anche contenere più applet: basta aggiungere un diverso marcatore applet per ogni applet
- Gli applet possono essere visualizzati con un apposito visualizzatore o con un browser abilitato al linguaggio Java
`appletviewer RectangleApplet.html`

File RectangleApplet.java

```
01: import java.awt.Graphics;
02: import java.awt.Graphics2D;
03: import java.awt.Rectangle;
04: import javax.swing.JApplet;
05:
06: /**
07:     Un applet che disegna due rettangoli.
08: */
09: public class RectangleApplet extends JApplet
10: {
11:     public void paint(Graphics g)
12:     {
13:         // Recupera Graphics2D
14:         Graphics2D g2 = (Graphics2D) g;
15:
16:         // Costruisce un rettangolo e lo disegna
17:         Rectangle box = new Rectangle(5, 10, 20, 30);
18:         g2.draw(box);
19:
```

continua

File RectangleApplet.java

```
20:         // Sposta il rettangolo di 15 unità verso destra e di
21:         // 25 unità verso il basso
22:         box.translate(15, 25);
23:
24:         // Disegna il rettangolo nella nuova posizione
25:         g2.draw(box);
26:     }
27: }
```

File RectangleApplet.html

1: `<applet code="RectangleApplet.class" width="300" height="400">`

2: `</applet>`

File RectangleAppletExplained.html

```
01: <html>
02:   <head>
03:     <title>Two rectangles</title>
04:   </head>
05:   <body>
06:     <p>Here is my <i>first applet</i>:</p>
07:     <applet code="RectangleApplet.class" width="300" height="400">
08:       </applet>
09:   </body>
10: </html>
```


Applet

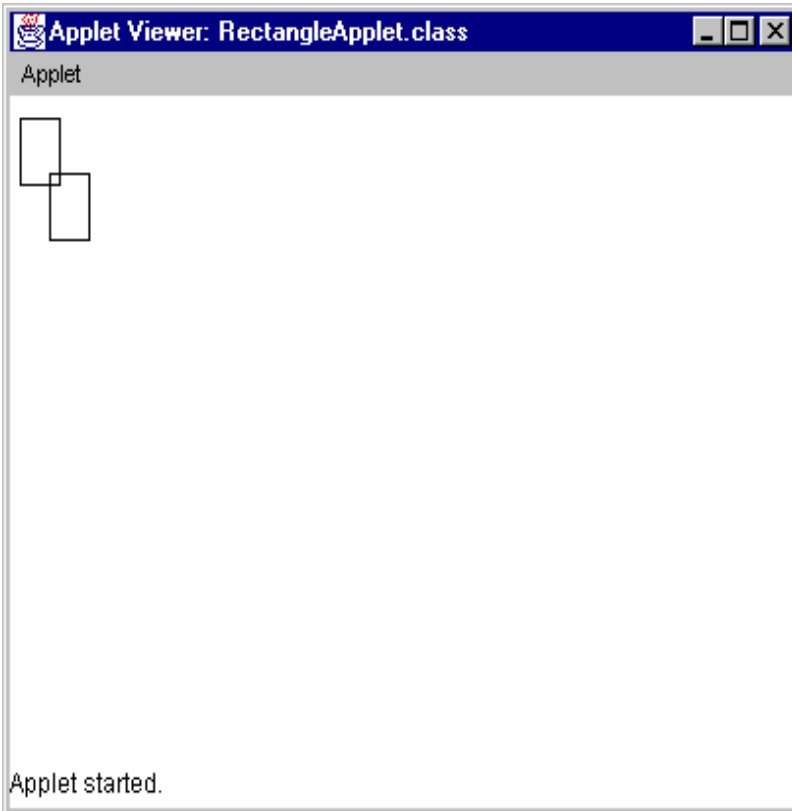


Fig.25: Un applet nel visualizzatore di applet

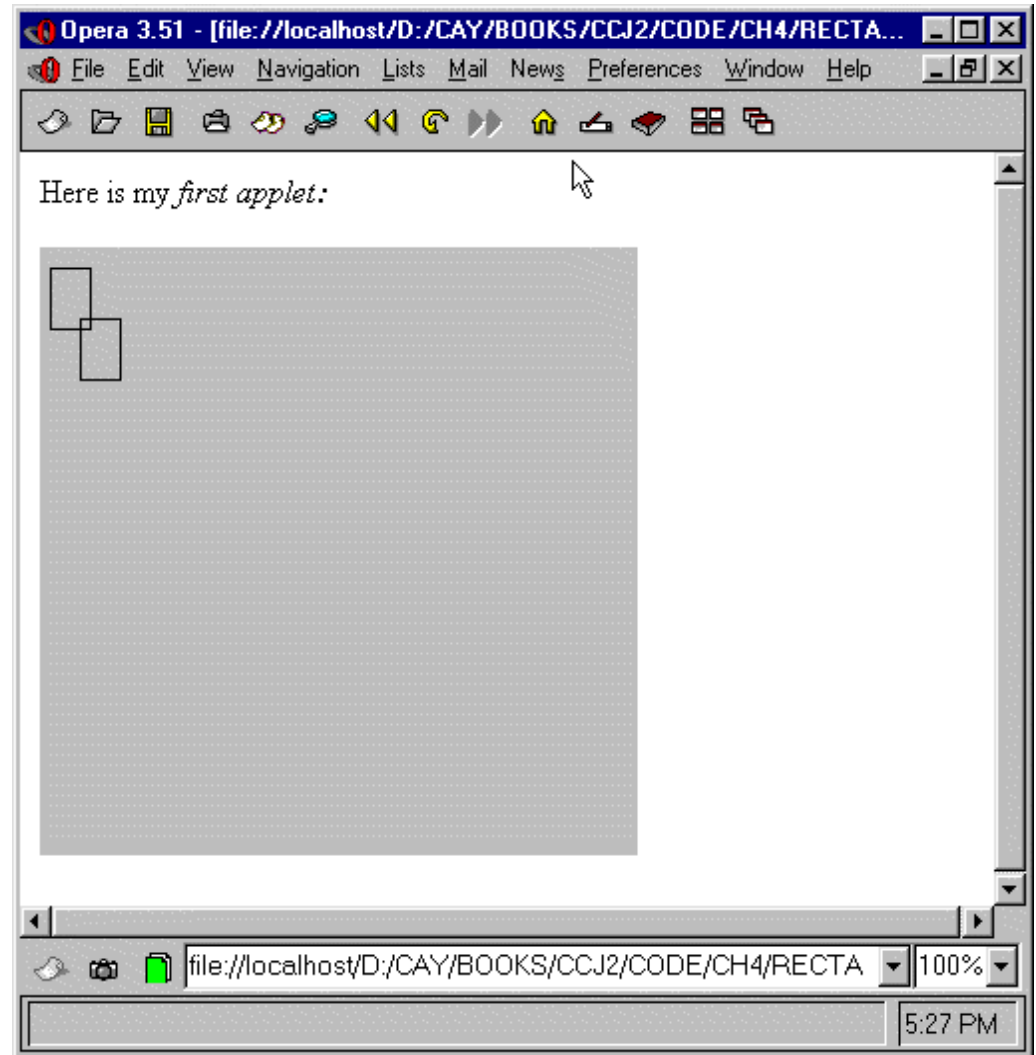


Fig.26: Un applet in un browser web

Ellissi

- `Ellipse2D.Double` **descrive un'ellisse**
- **Non useremo la classe** `Ellipse2D.Float`
- `Ellipse2D.Double` **è una classe interna: ciò non ci deve preoccupare se non per l'enunciato**
`import:`
`import java.awt.geom.Ellipse2D; // no .Double`
- **Disegnare un' ellisse è facile: usate lo stesso metodo**
`draw` della classe `Graphics2D` usato per disegnare
rettangoli (`g2.draw(ellipse);`):

```
Ellipse2D.Double ellipse  
    = new Ellipse2D.Double(x, y, width, height);  
g2.draw(ellipse);
```

Un'ellisse

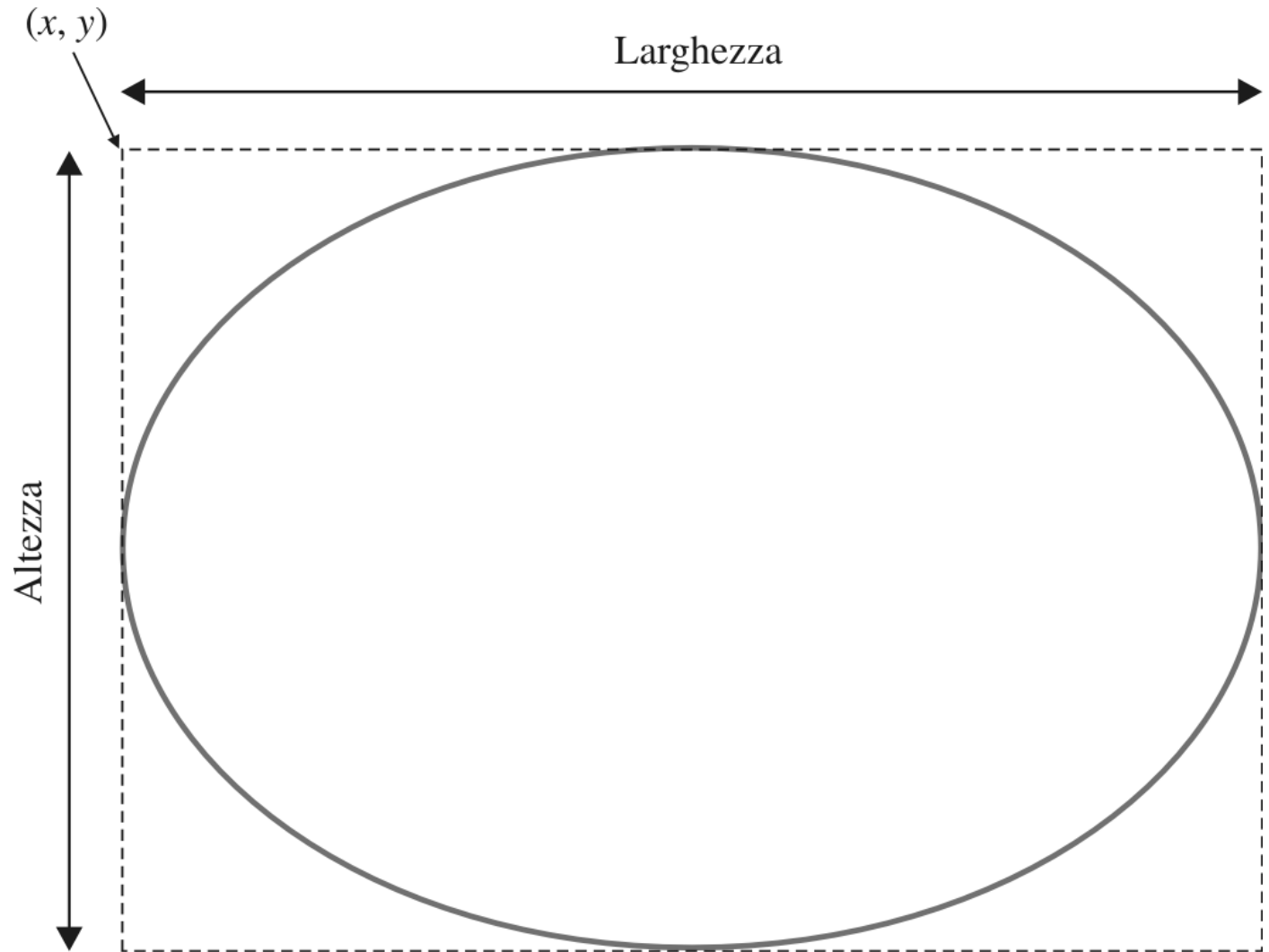


Figura 28:
Un'ellisse e il
suo rettangolo
di delimitazione

Disegnare segmenti

Per tracciare un segmento:

```
Line2D.Double segment  
    = new Line2D.Double(x1, y1, x2, y2);  
g2.draw(segment);
```

Oppure:

```
Point2D.Double from = new Point2D.Double(x1, y1);  
Point2D.Double to = new Point2D.Double(x2, y2);  
Line2D.Double segment  
    = new Line2D.Double(from, to);  
g2.draw(segment);
```

Disegnare testo

```
g2.drawString("Applet", 50, 100);
```

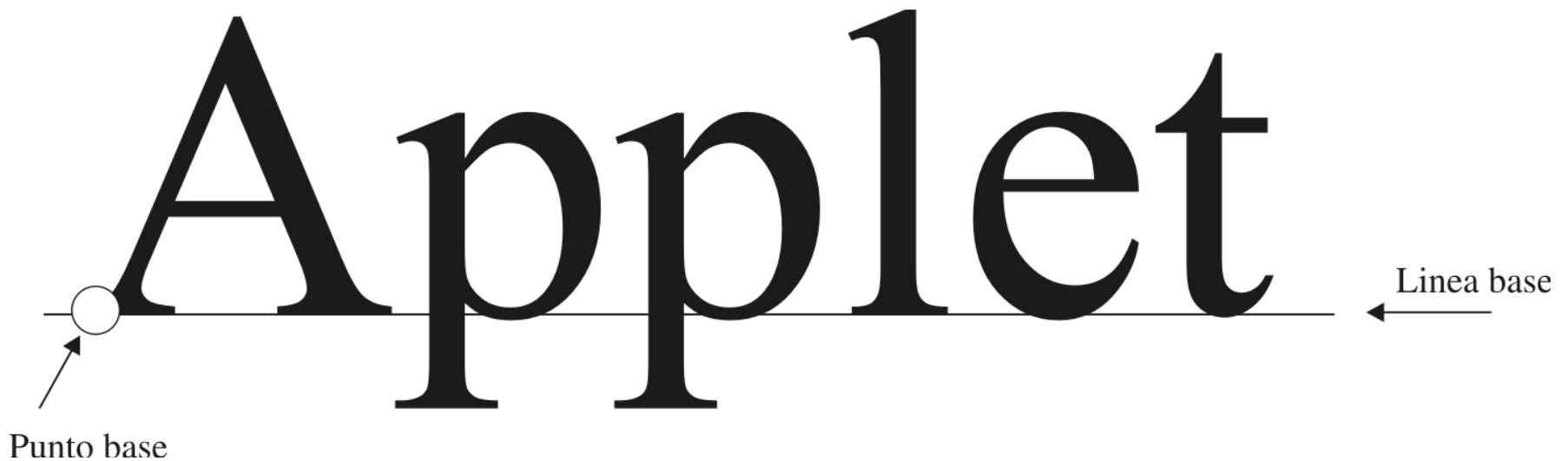


Figura 29: Il punto base e la linea base

Colori

- **Colori predefiniti:** `Color.BLUE`, `Color.RED`, `Color.PINK` **ecc.**
- **Specificare rosso, verde e blu tra 0 e 255:**

```
Color magenta = new Color(255, 0, 255);
```

- **Impostare il colore nell'oggetto di tipo `Graphics2D`:**

```
g2.setColor(magenta);
```

- **Il colore si usa quando si vuole disegnare o riempire una figura:**

```
g2.fill(rectangle);
```

Colori predefiniti e relativi valori RGB

Colore	Descrizione	Valore RGB
Color.BLACK	NERO	0, 0, 0
Color.BLUE	BLU	0, 0, 255
Color.CYAN	AZZURRO	0, 255, 255
Color.GRAY	GRIGIO	128, 128, 128
Color.DARK_GRAY	GRIGIO SCURO	64, 64, 64
Color.LIGHT_GRAY	GRIGIO CHIARO	192, 192, 192
Color.GREEN	VERDE	0, 255, 0
Color.MAGENTA	MAGENTA	255, 0, 255
Color.ORANGE	ARANCIONE	255, 200, 0
Color.PINK	ROSA	255, 175, 175
Color.RED	ROSSO	255, 0, 0
Color.WHITE	BIANCO	255, 255, 255
Color.YELLOW	GIALLO	255, 255, 0