

JAVA

IL LINGUAGGIO

Tipi e variabili

- Ogni valore nel linguaggio ha un tipo
- Ogni variabile deve essere dichiarata ed associata ad un tipo:

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

- Il tipo determina quali sono le operazioni legali sui valori (e le variabili)
- Variabili di un tipo possono solo assumere valori di quel tipo

Controllo dei tipi forte

- **Il compilatore controlla che il programma rispetti strettamente tutte le regole**
 - controlla l'uso dei cast
- **I programmi che superano i controlli del compilatore non causano errori nelle operazioni sui dati**

Oggetti e classi

OGGETTI

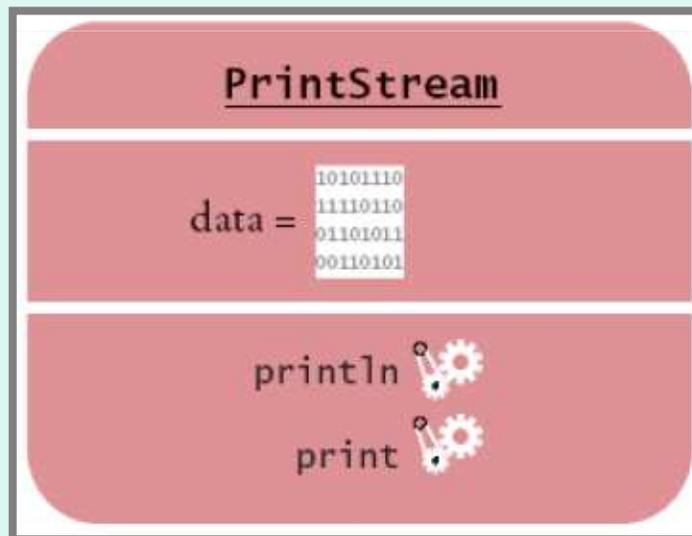
- **Le strutture dati complesse di un programma**
 - in C: structs, unions e arrays
 - In Java: oggetti (arrays inclusi)
- **Generalizzazione delle struct di C**
 - campi: struttura dell'oggetto
 - metodi: operazioni sui campi

CLASSI

- Descrivono la struttura dei campi
- Definiscono l'implementazione dei metodi
- Sono i tipi degli oggetti

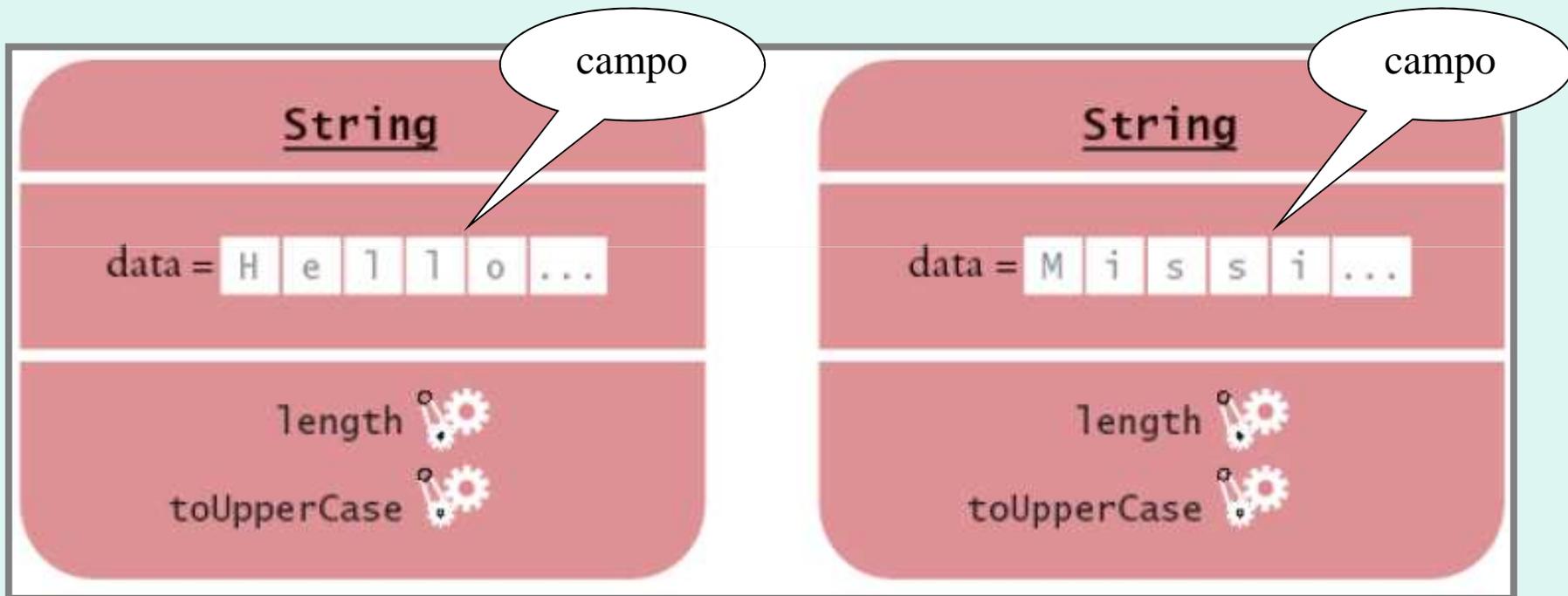
Oggetti e classi

- Ogni oggetto appartiene ad una classe.
- un oggetto di classe `PrintStream`



Due oggetti di tipo `String`

- Ogni oggetto ha una copia privata dei suoi campi



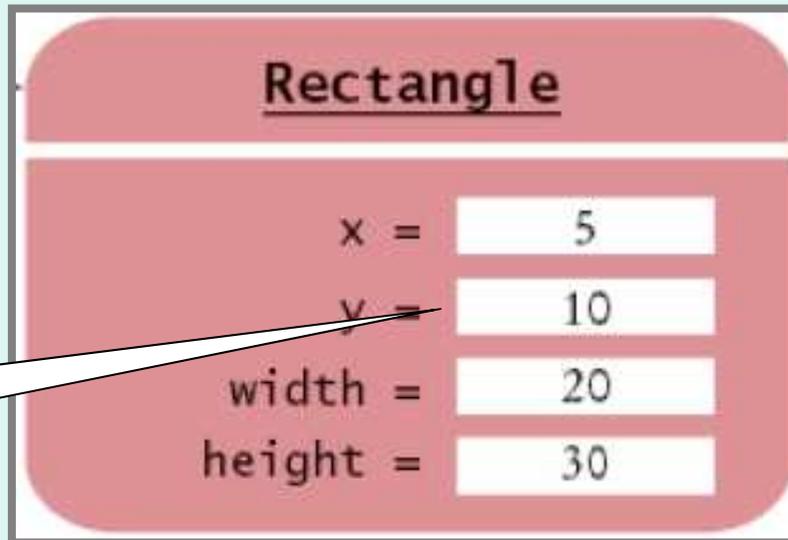
- Tutti gli oggetti di una classe condividono lo stesso codice dei metodi

Oggetti di classe Rectangle

- Classe definita nella libreria `java.awt`
- Descrive la struttura di un rettangolo posizionato sul piano cartesiano

(x,y) = posizione origine:
angolo in alto a sinistra

campi privati
dell'oggetto



Costruttori

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

- **Il costrutto `new` invoca il costruttore, per allocare nuovo oggetto di tipo `Rectangle`**
 - Il nome del costruttore coincide con il nome della classe
 - Usa i parametri (5, 10, 20, e 30) per inizializzare i dati dell'oggetto
- **Restituisce un riferimento all'oggetto**

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

Costruttori

- Una classe può fornire più di un costruttore
- *Overloading*
- Diverse modalità di creazione di oggetti

```
Rectangle box1 = new Rectangle(5, 10, 20, 30)
```

```
Rectangle box2 = new Rectangle()  
    // costruisce un rettangolo con origine (0,0)  
    // larghezza 0, e altezza zero 0
```

Sintassi: new

```
new ClassName(parameters)
```

Esempi:

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

```
new Rectangle()
```

- **Costruisce un nuovo oggetto, inizializza lo stato con i parametri, e restituisce un riferimento all'oggetto costruito.**

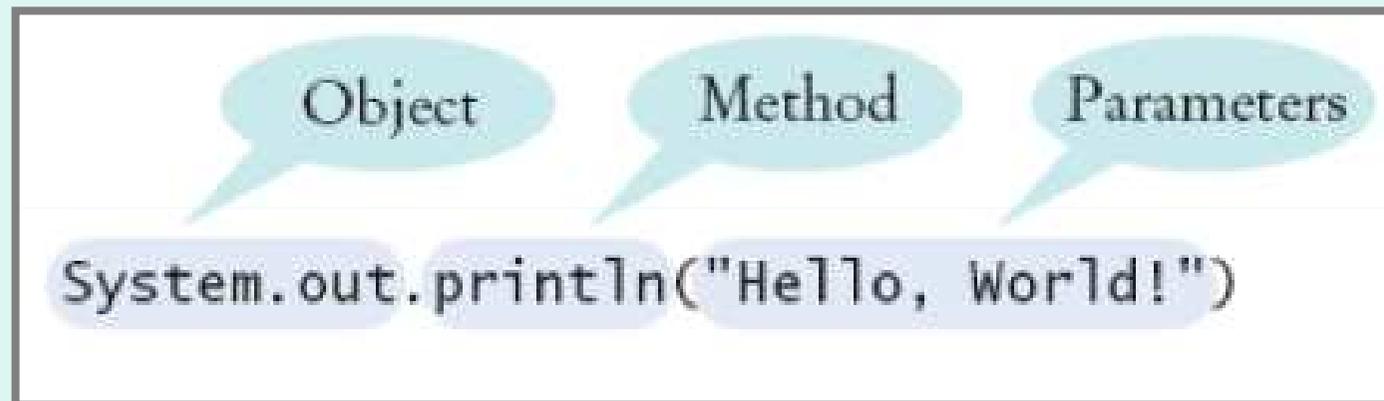
Metodi

- **Realizzano le operazioni supportate dagli oggetti**
- **Definiti dalla classe**
 - Classe definisce l'implementazione
 - Descrive la firma visibile all'esterno
- **Invocazione**
 - Deve indicare l'oggetto "target"

```
oggetto.metodo( argomenti )
```

a...

Metodi della classe `PrintStream`



Metodi della classe `String`

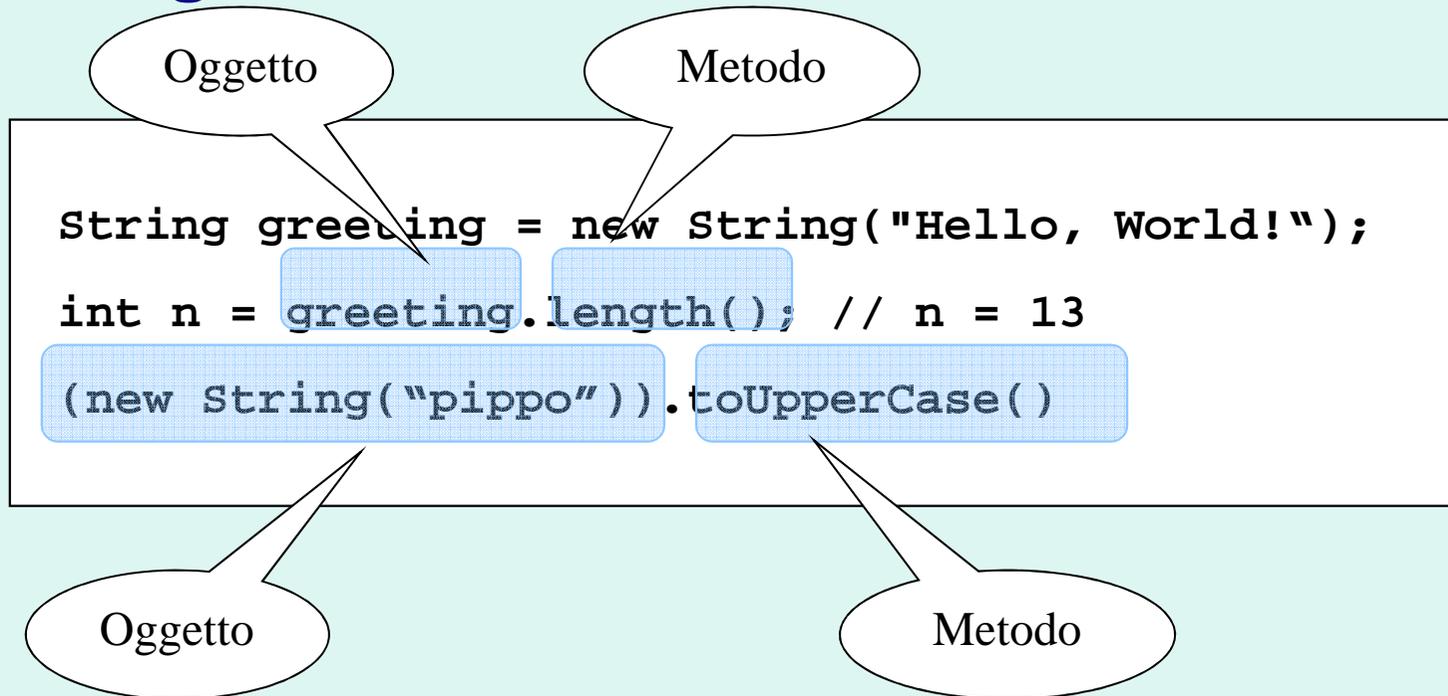
- `toUpperCase`: crea un altro oggetto di tipo `String` che contiene i caratteri della stringa originaria, ma convertiti in maiuscolo

```
String fiume = "Mississippi";  
String fiumeInUpperCase = fiume.toUpperCase();  
// fiumeInUpperCase = "MISSISSIPPI"
```

Continua...

Metodi della classe `String`

- `length()`: conta il numero di caratteri della stringa



Domande

- Quale è la sequenza di istruzioni per calcolare la lunghezza della stringa "Arcobaleno"?
- Quale è la sequenza di istruzioni per stampare la versione uppercase della stringa "Hello, World!"?
- È legale la seguente sequenza di istruzioni?

```
String fiume = "Mississippi";  
fiume.println();
```

Perché o perché no?

Risposte

- ```
new String("Arcobaleno").length()
```
- ```
System.out.println("Hello World".toUpperCase());
```
- **Non è legale: la variabile `fiume` ha tipo `String` e la classe `String` non definisce il metodo `println()`**

Controllo di tipi forte

- La classe di un oggetto definisce quali metodi si possono invocare sugli oggetti della classe
- Il compilatore controlla che le invocazioni rispettino la dichiarazione della classe

```
System.out.length(); // ERRORE DI TIPO (COMPILE-TIME)
```

Controllo di tipi forte

- La classe di un oggetto definisce quali metodi si possono invocare sugli oggetti della classe
- I metodi di un oggetto sono l'unico modo per agire sui campi dell'oggetto

```
Rectangle box = new Rectangle(5,10,20,20).  
System.out.println(box.width); // ERRORE DI TIPO
```

Metodi della classe Rectangle

- `getWidth()` : restituisce il valore che corrisponde alla base del rettangolo

```
Rectangle box = new Rectangle(5,10,20,20).  
System.out.println(box.width); // ERRORE DI TIPO  
System.out.println(box.getWidth()); // OK
```

Continua...

Metodi della classe Rectangle

- **Translate()**: modifica l'oggetto su cui viene invocato, traslando di *x* a destra, e di *y* verso il basso

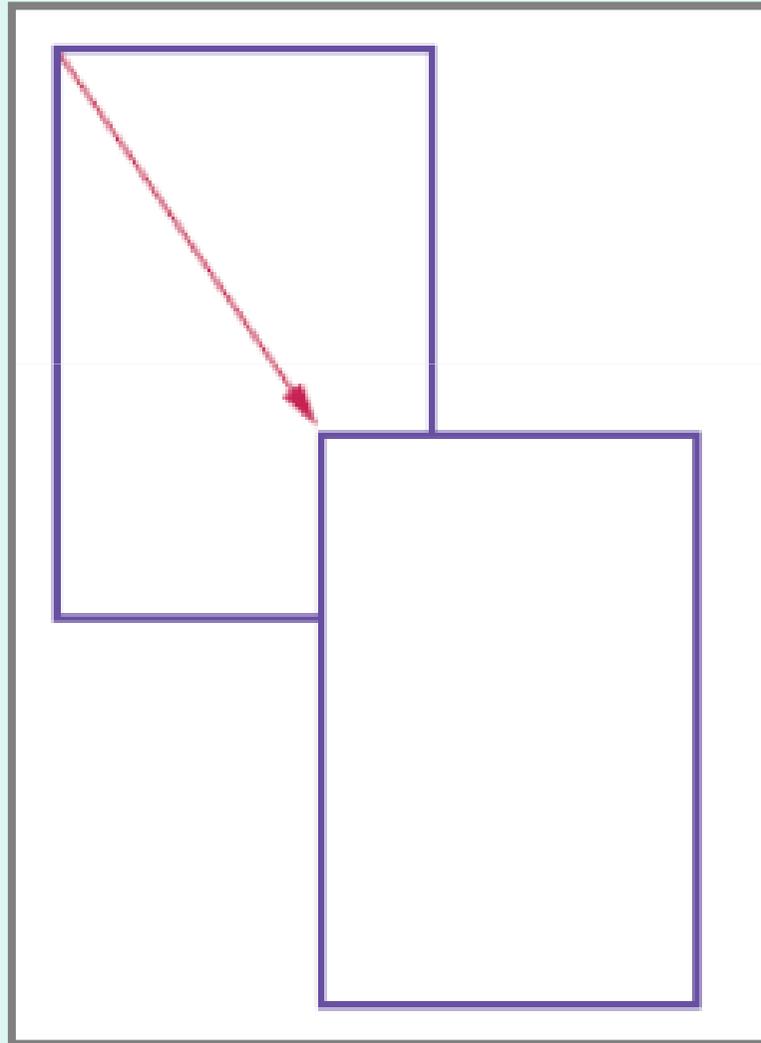
```
box.translate(x, y);
```

- **Mutators**: metodi con side effects per modificare i valori dei campi

Continua...

Metodi e side-effects

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



Oggetti e riferimenti

- Un riferimento è una astrazione del puntatore ad un oggetto
- La creazione di un oggetto con `new` restituisce un riferimento al nuovo oggetto

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

- Diverse variabili di tipo oggetto possono condividere lo stesso riferimento

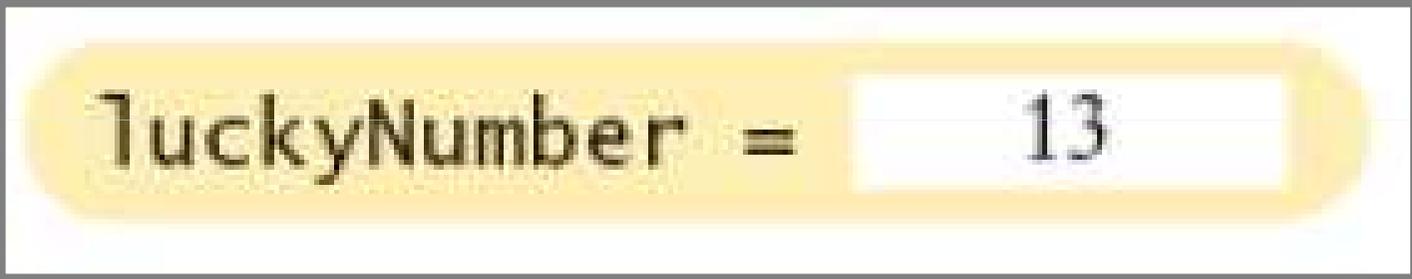
```
Rectangle box = new Rectangle(5, 10, 20, 30);  
Rectangle box2 = box;
```

Continua...

Assegnamento

- **Il comportamento dell'operazione di assegnamento varia a seconda del tipo della variabile (e quindi del valore) coinvolto.**
- **L'assegnamento su variabili di tipo primitivo ha un effetto diverso dall'assegnamento su variabili di tipo oggetto**

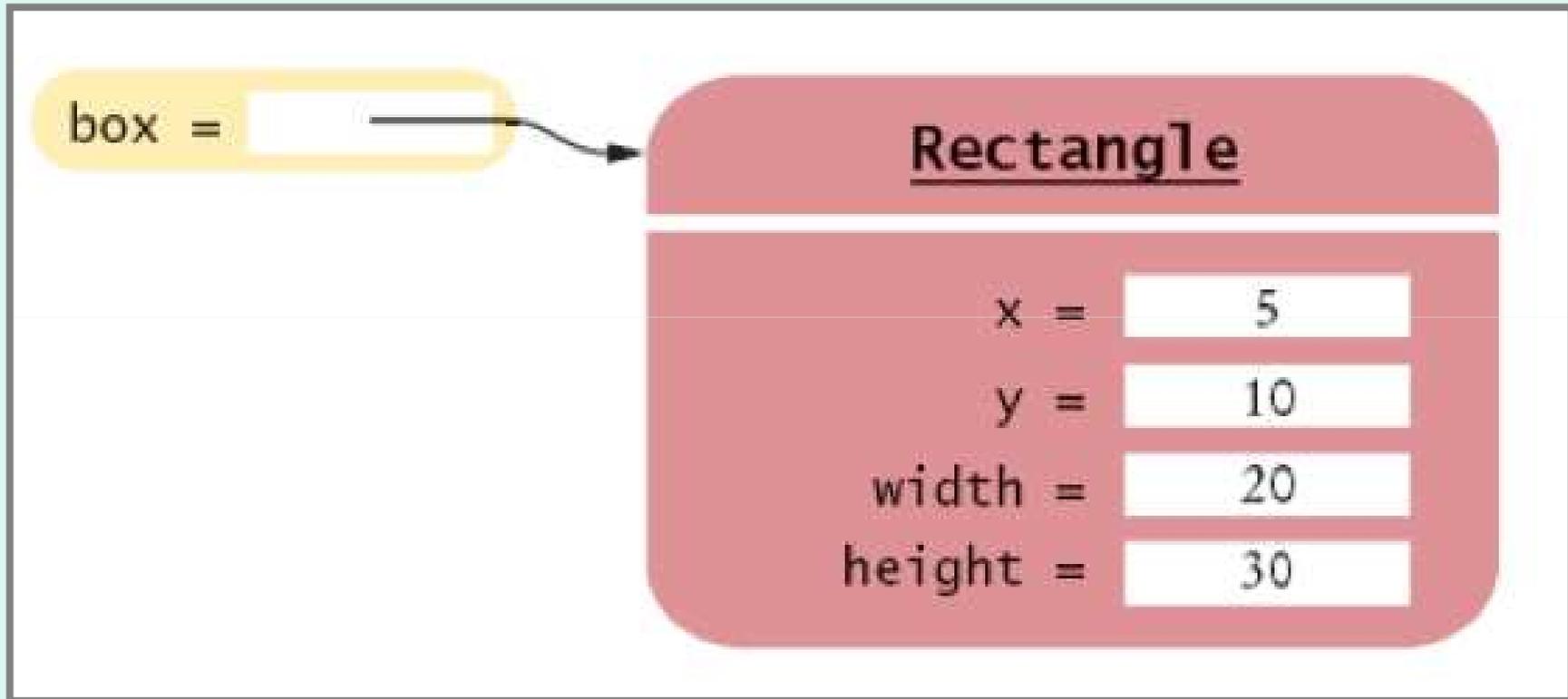
Variabili di tipo primitivo



luckyNumber = 13

Contengono valori del loro tipo

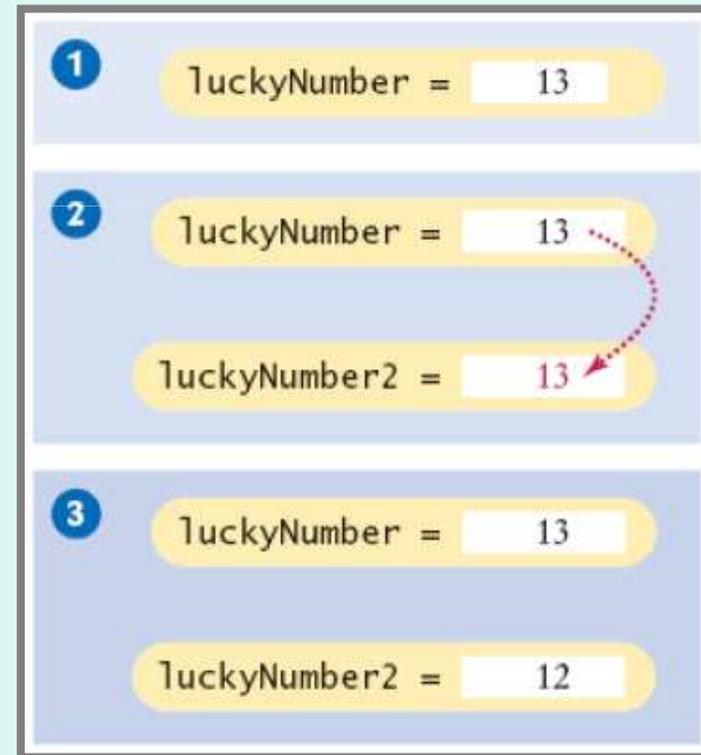
Variabili di tipo oggetto



Contengono riferimenti ad oggetti, non oggetti

Assegnamento su tipi primitivi

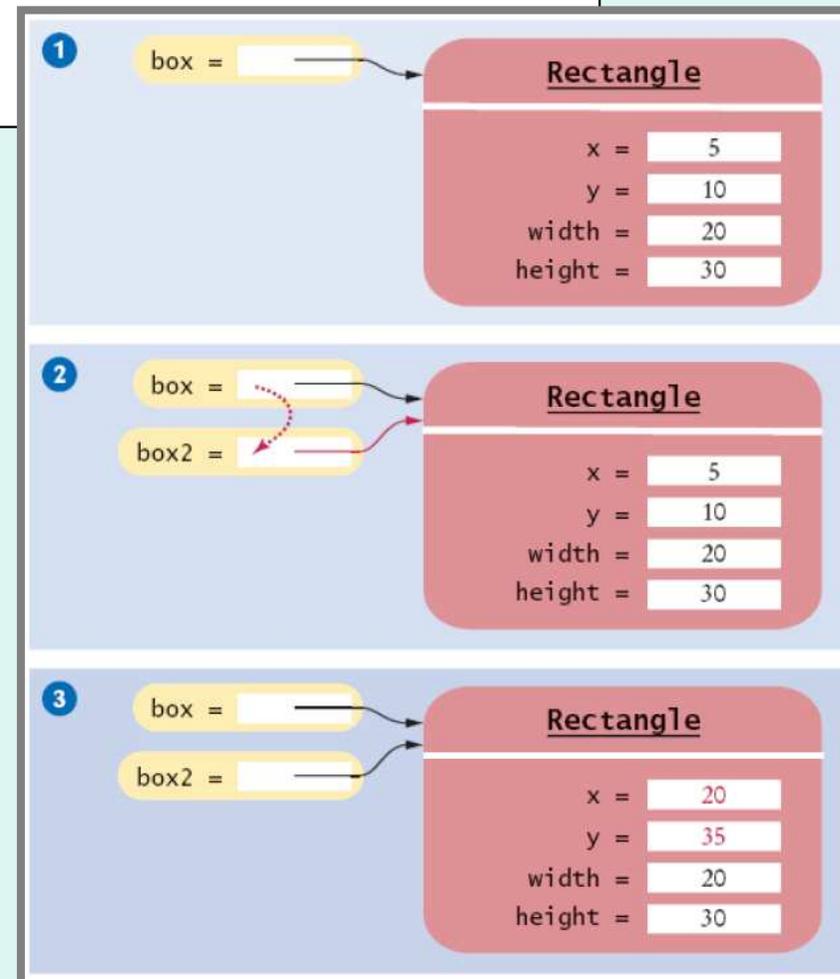
- ```
int luckyNumber = 13;
int luckyNumber2 = luckyNumber;
luckyNumber2 = 12;
```



**Due variabili, due valori distinti**

# Assegnamento su tipi oggetto

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



Due variabili, stesso riferimento

Programmi e classi applicazione

Un programma è costituito da più classi

- **Classe applicazione: contiene il `main`**
 - al più una classe applicazione in un programma
- **Il `main` avvia la computazione**
 - costruisce uno o più oggetti delle del programma
 - Invoca i metodi su tali oggetti
 - Stampa i risultati

File RectangleApp.java

```
01: import java.awt.Rectangle;
02:
03: public class RectangleApp
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:         // Stampa nuove info
13:         System.out.println("Origine dopo la traslazione:");
14:         System.out.println(box.getX());
15:         System.out.println(box.getY());
16:     }
17: }
```

Classi di libreria

- **Sempre includere le classi di libreria utilizzate dall'applicazione**
 - classi delle librerie raggruppate in packages
 - Importiamo le classi specificando i nomi di package e di classe
- ```
import java.awt.Rectangle;
```
- Le classi del package `java.lang` (ad esempio `String` e `System`) sono importate automaticamente

# Classi di libreria

```
import packageName.ClassName;
```

**Esempio:**

```
import java.awt.Rectangle; // importa la classe Rectangle
```

```
import packageName.*;
```

**Esempio:**

```
import java.util.*; // importa tutte le classi del package
```

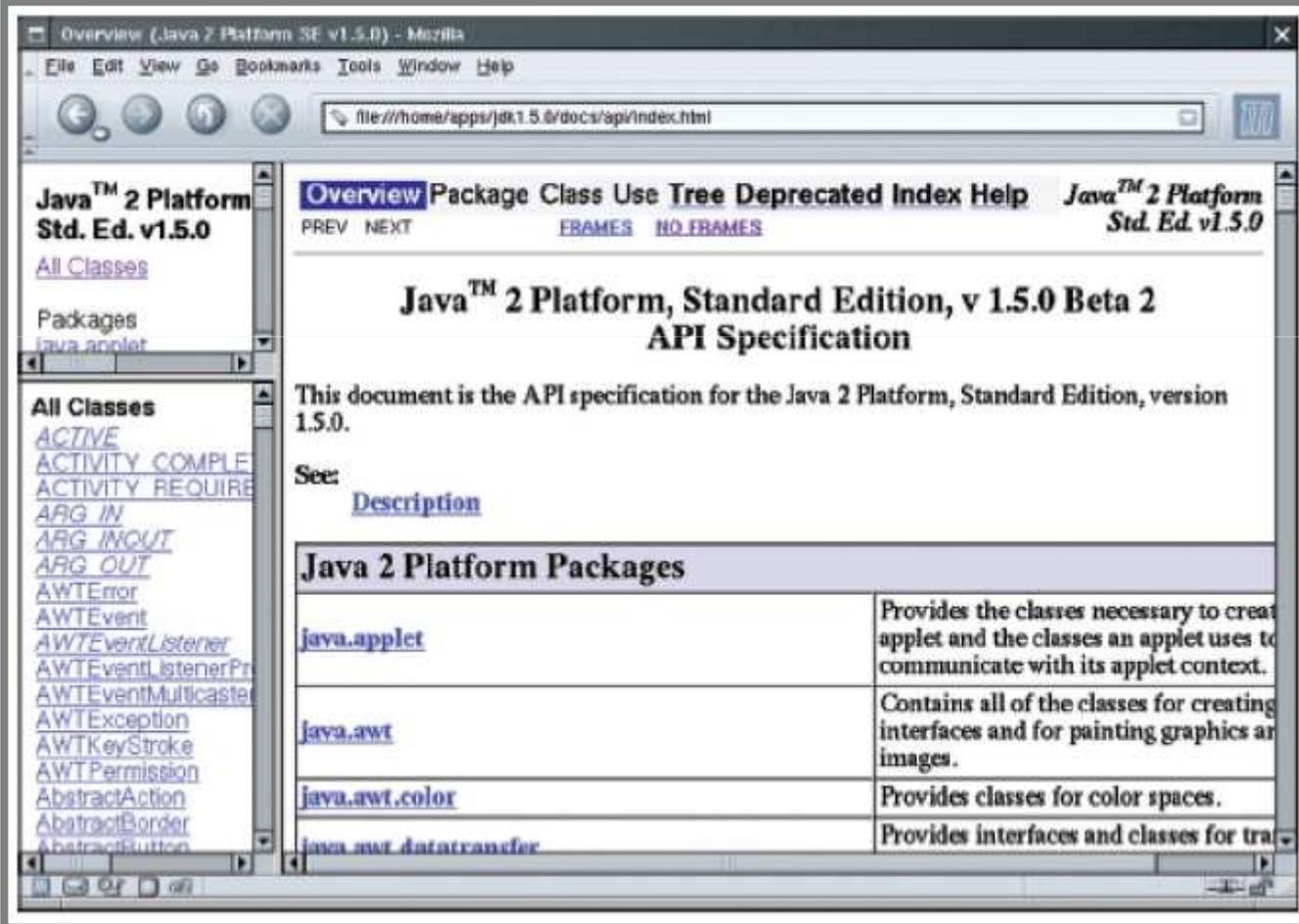
**Import in Java ~ #include in C**

# Librerie – Documentazione

---

- **API: Application Programming Interface**
- **Include la descrizione delle classi e dei relativi metodi della (fornitissima!) libreria Java**
- **<http://java.sun.com/j2se/1.5/docs/api/index.html>**
- **<http://java.sun.com/j2se/1.6/docs/api/index.html>**

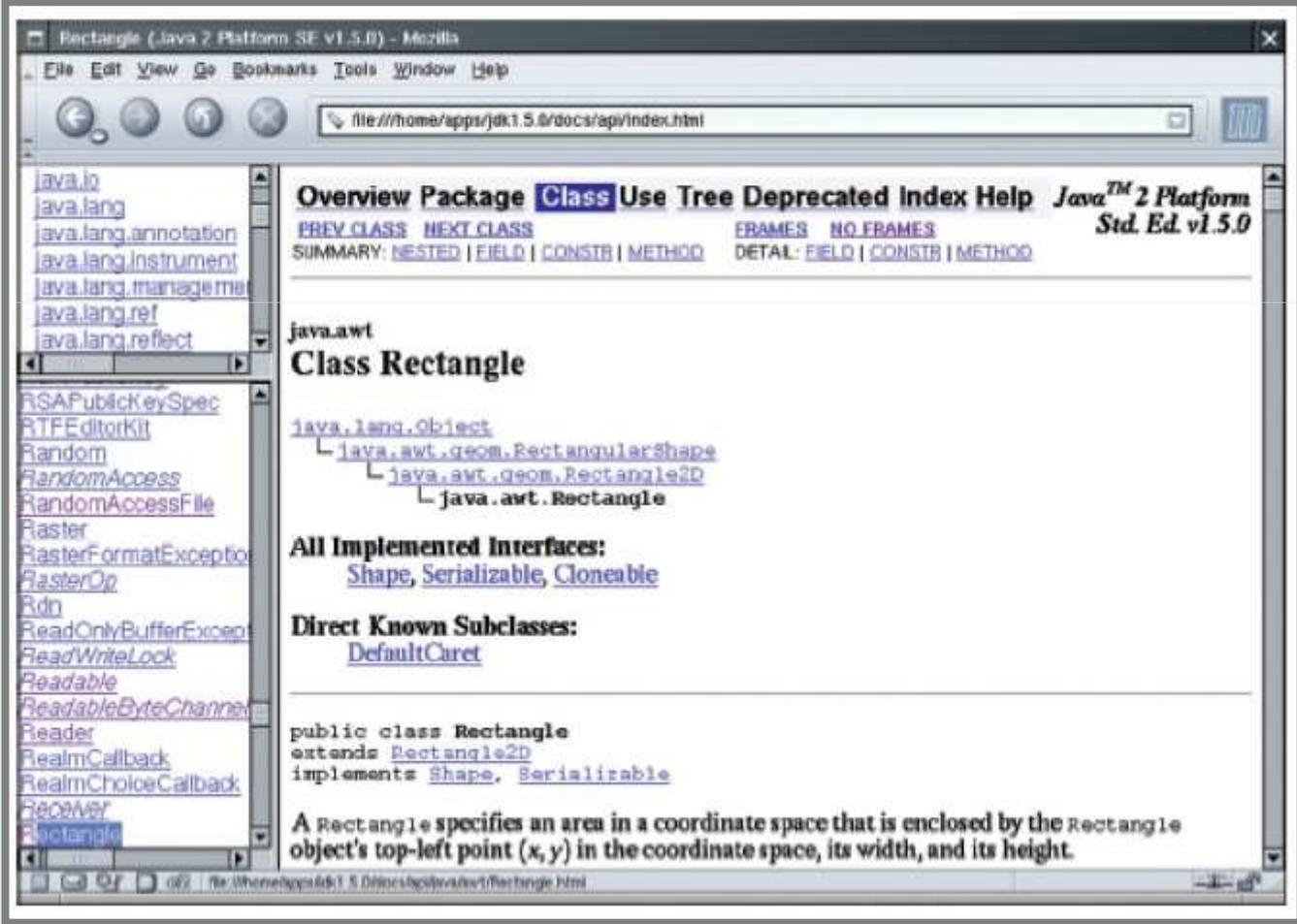
# Documentazione sulle API



The screenshot shows a Mozilla browser window displaying the Java 2 Platform API Specification. The browser title is "Overview (Java 2 Platform SE v1.5.0) - Mozilla". The address bar shows the file path: "file:///home/apps/jdk1.5.0/docs/api/index.html". The page content includes a navigation menu with "Overview", "Package", "Class", "Use Tree", "Deprecated", "Index", and "Help". The main heading is "Java™ 2 Platform, Standard Edition, v 1.5.0 Beta 2 API Specification". Below the heading, it states: "This document is the API specification for the Java 2 Platform, Standard Edition, version 1.5.0." and "See: [Description](#)". A table titled "Java 2 Platform Packages" lists several packages and their descriptions.

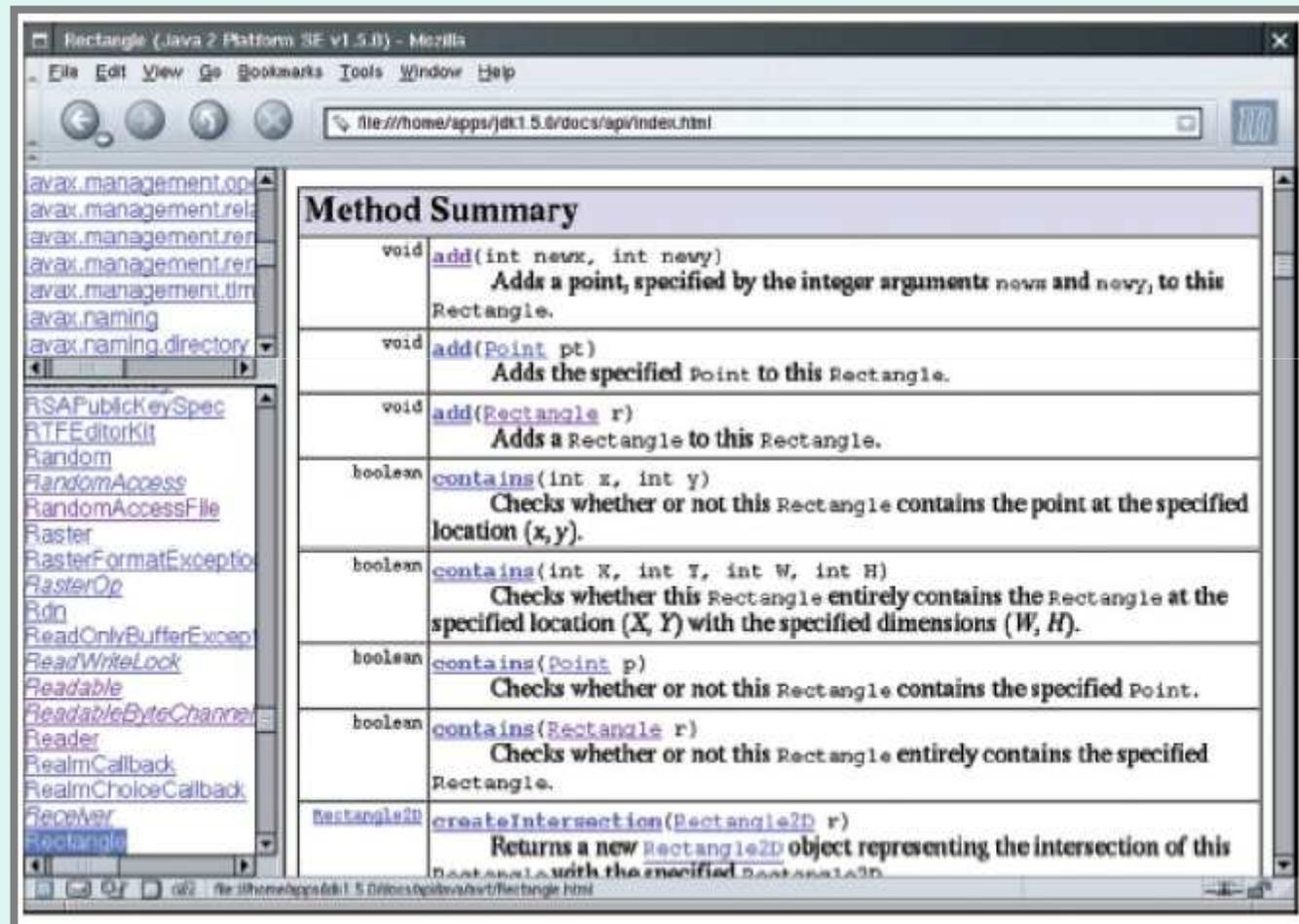
| Java 2 Platform Packages              |                                                                                                                           |
|---------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <a href="#">java.applet</a>           | Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context. |
| <a href="#">java.awt</a>              | Contains all of the classes for creating interfaces and for painting graphics and images.                                 |
| <a href="#">java.awt.color</a>        | Provides classes for color spaces.                                                                                        |
| <a href="#">java.awt.datatransfer</a> | Provides interfaces and classes for transferring data between applications.                                               |

# La API della classe Rectangle



The screenshot shows a Mozilla browser window displaying the Java API documentation for the `Rectangle` class. The browser title is "Rectangle (Java 2 Platform SE v1.5.0) - Mozilla". The address bar shows the file path: `file:///home/apps/jdk1.5.0/docs/api/index.html`. The page content includes a navigation menu with tabs for Overview, Package, Class (selected), Use, Tree, Deprecated, Index, and Help. The main content area is titled "Class Rectangle" and shows the class hierarchy: `java.lang.Object` (parent), `java.awt.geom.RectangularShape` (parent), `java.awt.geom.Rectangle2D` (parent), and `java.awt.Rectangle` (current class). It also lists "All Implemented Interfaces: Shape, Serializable, Cloneable" and "Direct Known Subclasses: DefaultCaret". The class signature is shown as `public class Rectangle extends Rectangle2D implements Shape, Serializable`. A descriptive paragraph states: "A Rectangle specifies an area in a coordinate space that is enclosed by the Rectangle object's top-left point (x, y) in the coordinate space, its width, and its height."

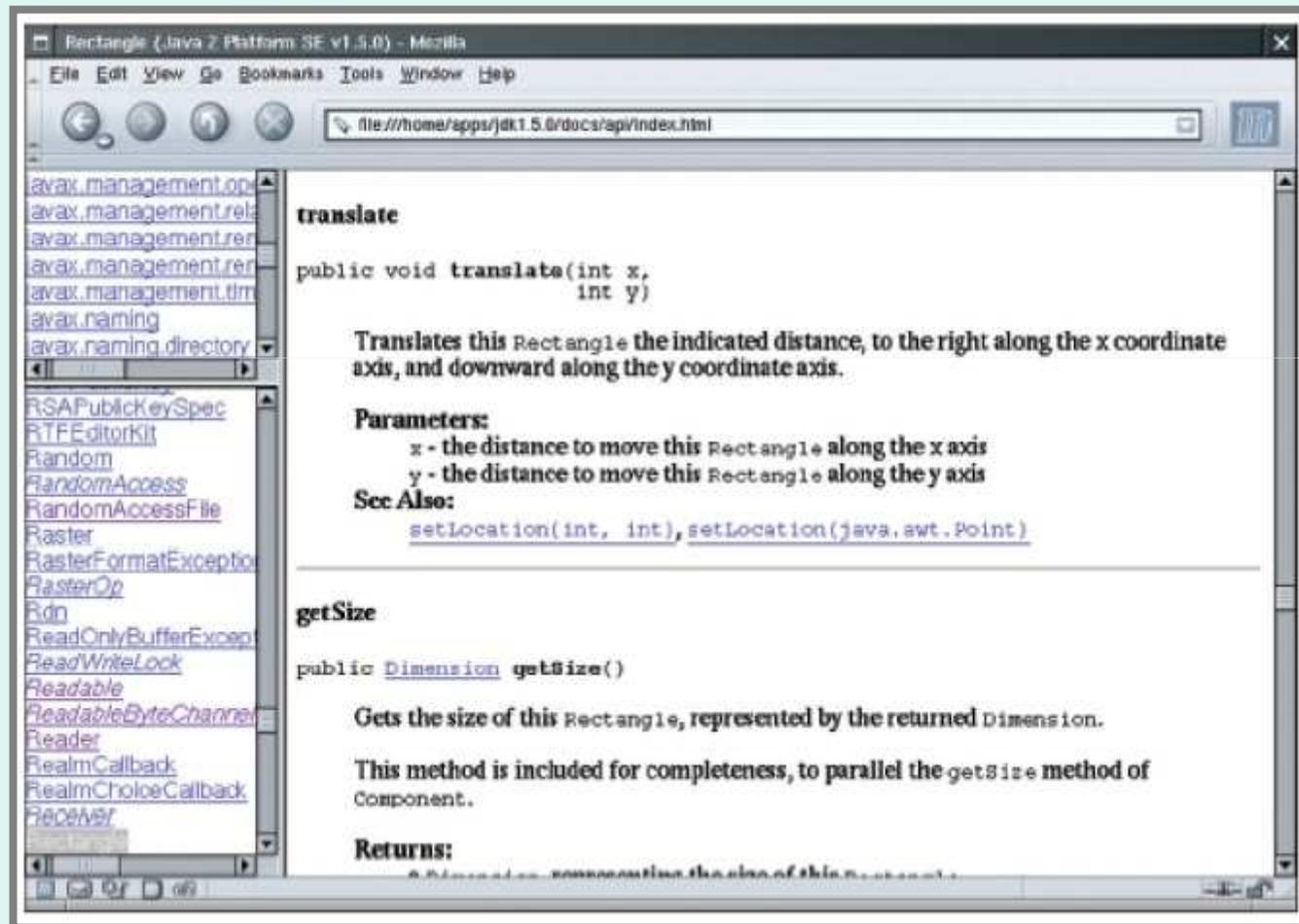
# Javadoc Method Summary



The screenshot shows a web browser window titled "Rectangle (Java 2 Platform SE v1.5.0) - Mozilla". The address bar displays "file:///home/apps/jdk1.5.0/docs/api/index.html". The browser content is divided into two main sections. On the left is a navigation pane with a list of Java classes, including "Rectangle" which is currently selected. On the right is the "Method Summary" section, which lists the following methods:

| Return Type              | Method Name                                       | Description                                                                                                                                                     |
|--------------------------|---------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| void                     | <code>add(int newx, int newy)</code>              | Adds a point, specified by the integer arguments <code>newx</code> and <code>newy</code> , to this Rectangle.                                                   |
| void                     | <code>add(Point pt)</code>                        | Adds the specified Point to this Rectangle.                                                                                                                     |
| void                     | <code>add(Rectangle r)</code>                     | Adds a Rectangle to this Rectangle.                                                                                                                             |
| boolean                  | <code>contains(int x, int y)</code>               | Checks whether or not this Rectangle contains the point at the specified location <code>(x, y)</code> .                                                         |
| boolean                  | <code>contains(int X, int Y, int W, int H)</code> | Checks whether this Rectangle entirely contains the Rectangle at the specified location <code>(X, Y)</code> with the specified dimensions <code>(W, H)</code> . |
| boolean                  | <code>contains(Point p)</code>                    | Checks whether or not this Rectangle contains the specified Point.                                                                                              |
| boolean                  | <code>contains(Rectangle r)</code>                | Checks whether or not this Rectangle entirely contains the specified Rectangle.                                                                                 |
| <code>Rectangle2D</code> | <code>createIntersection(Rectangle2D r)</code>    | Returns a new <code>Rectangle2D</code> object representing the intersection of this Rectangle with the specified <code>Rectangle2D</code> .                     |

# Documentazione del metodo translate



The screenshot shows a Mozilla browser window titled "Rectangle (Java 2 Platform SE v1.5.0) - Mozilla". The address bar contains the file path "file://home/apps/jdk1.5.0/docs/api/index.html". The left sidebar displays a list of Java classes, with "Rectangle" selected. The main content area shows the documentation for the `translate` method.

**translate**

```
public void translate(int x,
 int y)
```

Translates this `Rectangle` the indicated distance, to the right along the x coordinate axis, and downward along the y coordinate axis.

**Parameters:**

- x - the distance to move this `Rectangle` along the x axis
- y - the distance to move this `Rectangle` along the y axis

**See Also:**

- [setLocation\(int, int\)](#), [setLocation\(java.awt.Point\)](#)

---

**getSize**

```
public Dimension getSize()
```

Gets the size of this `Rectangle`, represented by the returned `Dimension`.

This method is included for completeness, to parallel the `getSize` method of `Component`.

**Returns:**

`Dimension`, representing the size of this `Rectangle`.

# Domande

---

- Quali sono gli errori nei seguenti frammenti di codice?
  - `Rectangle r = (5,10,15,20)`
  - `Double w = Rectangle().getWidth();`
  - `Rectangle r; r.translate(5,10);`

# Risposte

---

- **Manca il costruttore**
- **Manca `new`**
- **`r` non è inizializzato**

# Esempio: Tombola!

---

- **Vogliamo progettare una applicazione che realizza il gioco della tombola**
- **Versione semplificata: un banco, un giocatore, ogni giocatore una scheda**
- **Ci vengono già fornite le classi necessarie**
  - **Banco**
  - **Giocatore**

# Banco

---

- **Costruttore:**

- `Banco( )`

- crea una nuova istanza della classe

- **Metodi:**

- `int estraiNumero( )`

- restituisce un numero nell'intervallo [1..90]

- (ad ogni chiamata genera un numero diverso)

# Giocatore

---

- **Costruttore**

- `Giocatore(String nome)`

- crea un giocatore, con il nome indicato ed una sola scheda

- **Metodi:**

- `void controllaNumero(int x)`

- controlla se il numero è contenuto nella sua scheda; se sì lo “marca”

# Giocatore

---

- **Metodi:**

- `boolean tombola()`

- restituisce true quando tutti i numeri della sua scheda sono “marcati”

- `Integer[] verifica()`

- restituisce un array che contiene i numeri della sua scheda

Integer è un tipo reference isomorfo al tipo int

# Tombola

---

- **La classe applicazione che realizza il gioco:**
  - crea un banco ed un giocatore
  - esegue un ciclo in cui ad ogni iterazione in banco estrae un numero, il giocatore lo controlla e controlla se ha completato la sua scheda
  - il ciclo termina non appena il giocatore completa la sua scheda
  - All'uscita dal ciclo stampa i numeri della scheda del giocatore

# Tombola

```
public class Tombola
{
 public static void main(String[] args)
 {
 Banco b = new Banco();
 Giocatore g = new Giocatore("Mario");

 while (true) {
 int x = b.estrainNumero();
 System.out.println("Numero estratto = " + x);
 g.controllaNumero(x);
 if (g.tombola()) break;
 }

 Integer[] scheda = g.verifica();
 for (int i = 0; i < scheda.length; i++)
 System.out.println(scheda[i])
 }
}
```

# NOTE

---

- **Nessuna informazione data sulla struttura interna degli oggetti in gioco**
- **Conosciamo i metodi che essi forniscono**
  - notare bene: conosciamo la specifica dei metodi, non come sono implementati
- **E' tutto quello che serve per programmare l'applicazione**
- **La chiave del concetto di *encapsulation***

# ***Oggetti / Encapsulation***

---

- **Oggetti**
  - esportano la specifica dei propri metodi
    - interfaccia pubblica che definisce l'interazione tra oggetti e codice "cliente"
  - mascherano la propria struttura interna (i campi) e l'implementazione dei metodi
- **Codice cliente**
  - indipendente dall'implementazione degli oggetti

# Esercizio

---

- **Create una versione della tombola in cui possono giocare più giocatori**
  - Il gioco termina non appena uno dei giocatori completa la sua scheda