Capitolo 2
Utilizzare oggetti

Obiettivi del capitolo

- Imparare a utilizzare variabili
- Capire i concetti di classe e oggetto
- Saper invocare metodi
- Essere in grado di consultare la documentazione dell'API di Java
- Capire la differenza tra oggetti e riferimenti a oggetti

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 I valori
  - Possono essere utilizzate al posto degli oggetti che memorizzano

Sintassi di Java
2.1: Definizione di variabile

typeName variableName = *value*;
or
typeName variableName;

Example:

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

Purpose:
To define a new variable of a particular type and optionally supply an initial value

Identificatori

- Identificatore: nome di una variabile, di un metodo, di una classe.
- Regole per gli identificatori in Java:
  - Possono essere composti di lettere, cifre, e segni di sottolineatura (_)
  - Non possono iniziare con una cifra
  - Non si possono usare altri simboli, come, ad esempio ? o %
  - Gli spazi non sono ammesi all' interno degli identificatori
  - Le *parole riservate* non possono essere usate
  - Sono sensibli alla differenza tra lettere maiuscole e minuscole

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

Verifica

1. Di che tipo sono i valori `0` e `"0"`?
2. Quali dei seguenti identificatori sono validi?

```
Greeting1
g
void
101dalmatians
Hello, World
<greeting>
```

3. Definite una variabile adatta a memorizzare il vostro nome, usando un nome con "le gobbe".

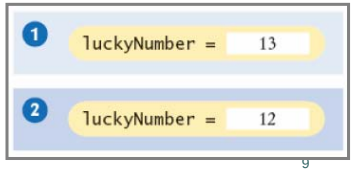
Risposte

1. `int` e `String`
2. Solo i primi due sono identificatori validi
3. `String myName = "John Q. Public";`

L'operatore di assegnazione

- L'operatore di assegnazione: =
- Non identifica un'uguaglianza
- Usato per cambiare il valore di una variabile

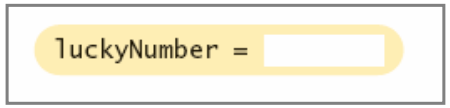
```
int luckyNumber = 13; (1)
luckyNumber = 12; (2)
```



Variabili non utilizzate

- Errore:

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



Sintassi di Java
2.2: Assegnazione

***variableName* = *value*;**

Esempio:
```
luckyNumber = 12;
```

Obiettivo:
Assegnare un valore a una variabile definita in precedenza.

Verifica

4. L'espressione `12 = 12` è valida nel linguaggio Java?
5. Come si fa a fare in modo che la variabile *greeting* abbia il valore `"Hello, Nina!"`?

Risposte

4. No, a sinistra dell'operatore = deve esserci una variabile
5.
```
greeting = "Hello, Nina!";
```

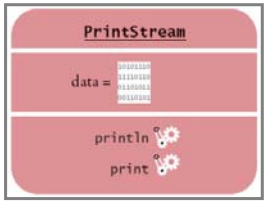
Notare che

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

non è la risposta corretta, perchè in questo caso verrebbe definita una nuova variabile

Oggetti e classi

- Oggetti: entità di un programma che si possono manipolare (invocando metodi)
- ∀ oggetto ∈ una classe.
  - `System.out` ∈ `PrintStream`



Metodi

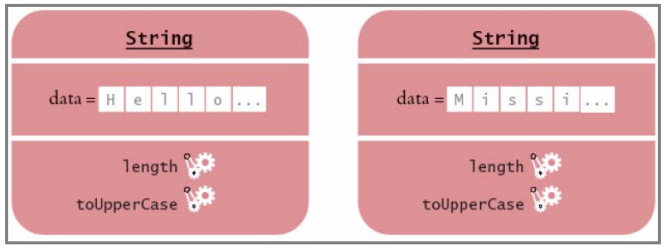
- Metodo
  - sequenza di istruzioni che accede ai dati di un oggetto
- Si manipolano gli oggetti invocando i metodi
- Classe
  - insieme di oggetti con lo stesso comportamento
- La classe specifica i metodi che possono essere applicati ai suoi oggetti

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


Metodi

- L'nterfaccia pubblica:
  - specifica che cosa si può fare con gli oggetti di una classe

Rappresentazione di due oggetti di tipo String



I metodi di String

- `length:` fornisce il conteggio dei caratteri

```
String greeting = "Hello, World!";
int n = greeting.length(); // sets
n to 13
```

Continua...

I metodi di 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();
// sets bigRiver to "MISSISSIPPI"
```


I metodi di String

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

```
System.out.length();
// This method call is an error
```

Verifica

6. In che modo si può calcolare la lunghezza della stringa `"Mississippi"`?
7. In che modo si può visualizzare la versione maiuscola della stringa `"Hello, World!"`?
8. Si può invocare `river.println()`? Perchè o perchè no?

Risposte

6. `river.length() or "Mississippi".length()`
7. `System.out.println(greeting.toUpperCase());`
8. No perchè la variabile `river` è di tipo `String`, mentre il metodo `println` non esiste nella classe `String`.

Parametri impliciti ed espliciti

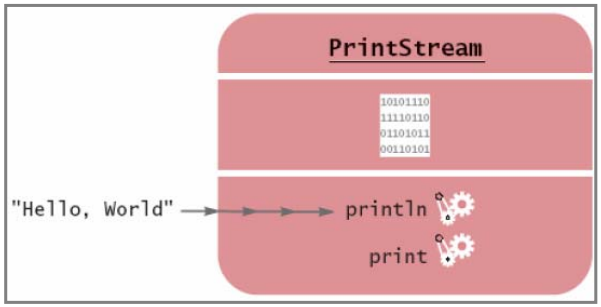
- Parametro (parametro esplicito): è quel parametro inserito in un metodo. Non tutti i metodi hanno parametri espliciti.

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

- Parametro implicito: l'oggetto col quale un metodo è invocato.

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


Parametri espliciti ed impliciti

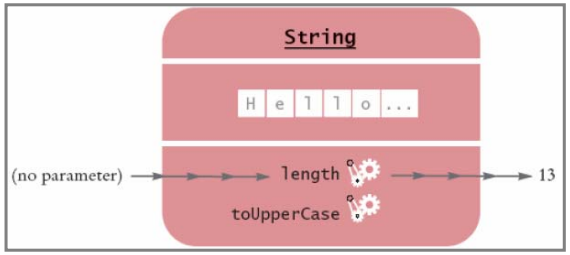


Valori di ritorno

- Valori di ritorno
  - risultato che il metodo ha calcolato tramite il metodo che abbiamo chiamato.

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


Valori di ritorno



Passare i Valori di ritorno

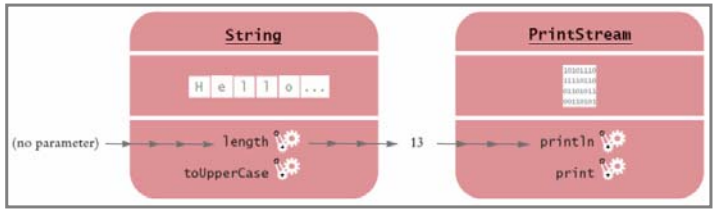
- Puoi usare i valori di ritorno come parametri per altri metodi:

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

- Non tutti i metodi hanno dei valori di ritorno. Esempio:

```
println
```


Passare i Valori di ritorno



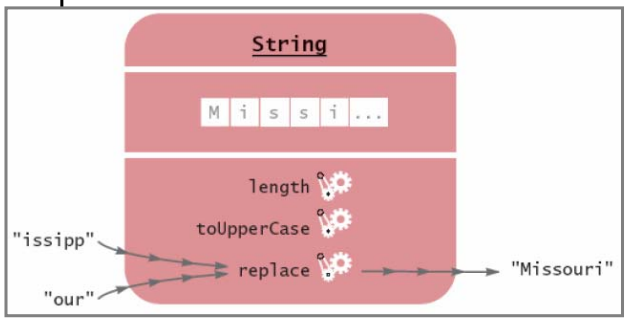
Chiamate complesse

- Il metodo `replace` esegue un operazione di ricerca e sostituzione

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

- Questa chiamata al metodo ha:
  - Un parametro implicito: la stringa **`"Mississippi"`**
  - Due parametri espliciti: le stringhe **`"issipp"`** e **`"our"`**
  - Valore di ritorno: la stringa **`"Missouri"`**

Chiamate complesse



Definizione di un metodo

- La definizione del metodo specifica il tipo del parametro esplicito e il valore di ritorno.
- Tipo del parametro implicito = classe corrente
  - non menzionato nella definizione del metodo.

Definizione di un metodo

- Esempio: `Class String` definisce

```
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
```

Definizioni del metodo

- Se il metodo non ha valori di ritorno, il tipo del valore di ritorno è dichirato come

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

- Il nome del metodo è sovrascritto se la classe ha più di un metodo con lo stesso nome (ma differenti tipi di parametri)

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


Verifica

- Quali sono i parametri impliciti, espliciti, e i valori di ritorno chiamando il metodo `river.length()`?
- Qual'è il risulatato della chiamata `river.replace("p", "s")`?
- Qual'è il risulatato della chiamata `greeting.replace("World", "Dave").length()`?
- Com'è definito il metodo `toUpperCase` nella classe `String` ?

Risposte

- Il parametro implicito è river. Non c'è un parametro esplicito. Il valore di ritorno è

  "Missississi"
- 12
- Come public String toUpperCase(), senza un parametro esplicito e il tipo del valore di ritorno è String.

Tipi di numero

- Interi: `short, int, long`
  `13`
- Reali: `float, double`
  `1.3`
  `0.00013`

Tipi di numero

- Quando un numero reale è moltiplicato o diviso per 10, cambia solo la posizione della virgola: essa "fluttua".
- Questa rappresentazione è relativa alla notazione "scientifica"

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

- I numeri non sono oggetti, ma
- I numeri sono tipi primitivi.

Operazioni aritmetiche

- Operatori: `+ - *`

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

Come in matematica, l' operatore * ha un legame più forte rispetto all' operatore +.

```
x + y * 2    // means the sum of x and y * 2
(x + y) * 2  // multiplies the sum of x and y with 2
```


Verifica

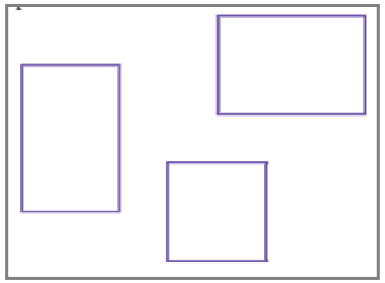
- Quale tipo di numero potresti usare per immagazzinare l'area di un cerchio?
- Perchè l'espressione `13.println()` è errata?
- Scrivi l'espressione per calcolare la media dei valori `x` e `y`.

Risposte

- `double`
- Il tipo `int`, non è un oggetto e non puoi chimare un metodo in esso.
- `(x + y) * 0.5`

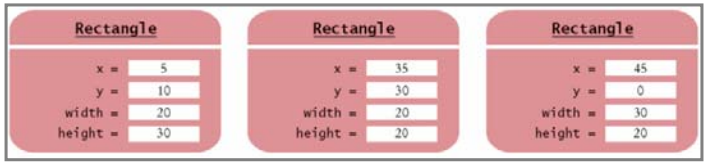
Rettangoli e oggetti rettangolari

- Oggetti del tipo **`Rectangle`** descrivono forme rettangolari



Rettangolari e oggetti rettangolari

- L'oggetto **`Rectangle`** non è una figura rettangolare. Esso è un oggetto che contiene un set di numeri che descrivono un rettangolo



Costruzione di oggetti

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

- In dettaglio:
  - Il nuovo operatore crea l'oggetto
  - Esso usa parametri (in questo caso, `5`, `10`, `20`, e `30`) per inizializzare l'oggetto.
  - Esso ha come valore di ritorno l'oggetto stesso
- Di solito l'output di un nuovo operatore è contenuto in una variabile.

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


Costruzione di oggetti

- Il processo di creazione di un nuovo oggetto è chiamato costruzione.
- I quattro valori `5`, `10`, `20`, e `30` sono chiamati parametri di costruzione.
- Alcune classi ti permettono di costruire gli oggetti in più modi.

```
new Rectangle()
    // constructs a rectangle with its top-left corner
    // at the origin (0, 0), width 0, and height 0
```

Sintassi 2.3: Costruzione di Oggetti

```
new ClassName(parameters)
```

Esempio:

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

Obiettivo:

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

Verifica

16. Come costruisci un quadrato di centro (100,100) e lato 20?
17. Cosa viene visualizzato dal seguente enunciato?

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


Risposte

16. `new Rectangle(90, 90, 20, 20)`
17. 0

Metodi d'accesso e modificatori

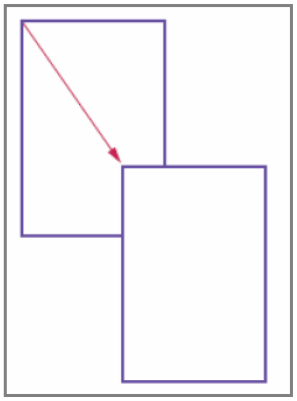
- Metodi `d'accesso`: non cambiano lo stato dei parametri impliciti di un oggetto.

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

- Metodi `modificatori`: cambiano lo stato dei parametri impliciti di un oggetto.

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

Metodi d'accesso e modificatori



Verifica

18. Il metodo `toUpperCase` della classe `String` è un metodo d'accesso o modificatore?
19. Quale chiamata a `translate` è necessaria per spostare il rettangolo `box` in modo che il suo vertice superiore sinistro si trovi all'origine (0, 0)?

Risposte

18. Un metodo d'accesso, non modifica la stringa originale ma crea una nuova stringa con i caratteri maiuscoli
19. `box.translate(-5, -10),` purchè il metodo sia chiamato dopo la memorizzazione del nuovo rettangolo box.

Realizzare un programma di Collaudo

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

Importare pacchetti

Non dimenticare di includere i pacchetti necessari:

- I pacchetti sono una raccolta di classi Java
- Importa le classi della libreria specificando il pacchetto e il nome della classe:

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

- Non hai bisogno di importare le classi del pacchetto `java.lang` come `String` o `System`

Sintassi 2.4: Importazione di una classe da un pacchetto

```
import packageName.ClassName;
```

Esempio:

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

Obiettivo:
Importare una classe da un pacchetto per utilizzarla.

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


Verifica

20. La classe `Random` è definita nel pacchetto `java.util`. Cosa dovete fare per usare tale classe in un vostro programma?
21. Perchè il programma `MoveTester` non visualizza la larghezza e l'altezza del rettangolo?

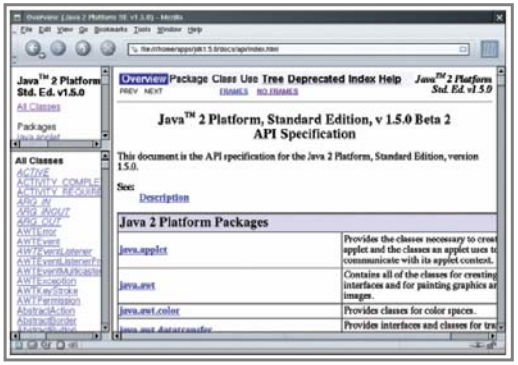
Risposte

20. Aggiungi l'enunciato `import java.util.Random;` all'inizio del tuo programma.
21. Perchè il metodo `translate` non modifica la forma del rettangolo.

La documentazione API

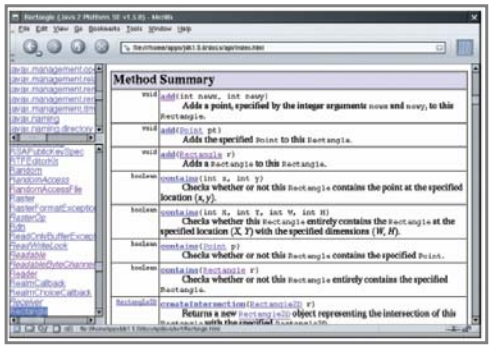
- API: Application Programming Interface
- Elenca le classi e i metodi delle liberie Java
- http://java.sun.com/j2se/1.5.0/docs/api/index.html

La documentazione API per la libreria standard di Java

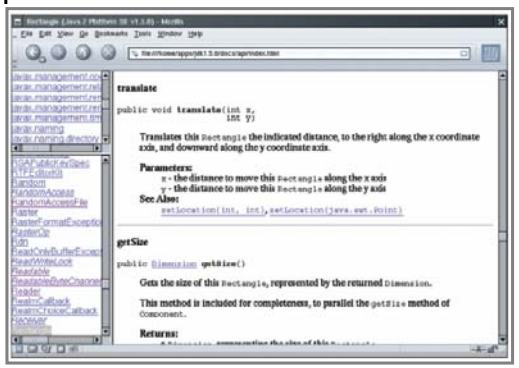


La documentazione API per la classe `Rectangle`

L’elenco dei metodi della classe `Rectangle`



Documentazione di `translate`



Verifica

22. Consultate la documentazione API della classe `String`. Quale metodo usereste per ottenere la stringa `"hello, world!"` a partire da `"Hello, World!"`?
23. Consultate la descrizione del metodo `trim` nella documentazione API della classe `String`. Qual è il risultato dell’applicazione del metodo `trim` alla stringa `" Hello, Space ! "`? (Attenzione agli spazi nella stringa.)

Risposte

22. `toLowerCase`
23. `"Hello, Space !"` solo gli spazi all’inizio e alla fine vengono tolti.

Riferimenti a oggetti

- Descrive la posizione dell'oggetto in memoria
- L'operatore new restituisce un riferimento ad un oggetto

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

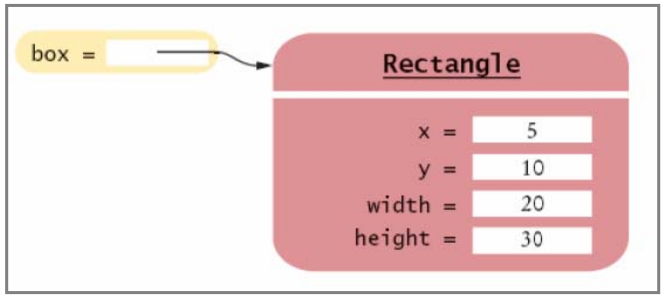
- Più variabili oggetto possono riferirsi al medesimo oggetto:

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

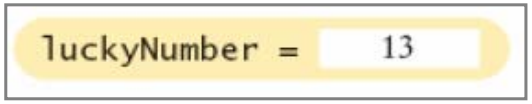

Riferimenti a oggetti

- Le variabili di tipo primitivo sono diverse dalle variabili oggetto

Variabili oggetto e variabili numeriche

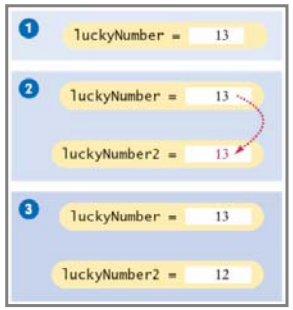


Variabili oggetto e variabili numeriche



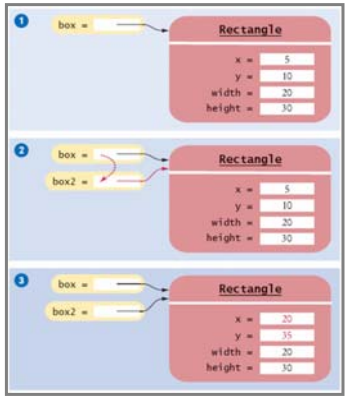
Copiare i numeri

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



Copiare i riferimenti a oggetti

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



Verifica

24. Qual è l'effetto dell'assegnazione `greeting2 = greeting`?
25. Dopo aver invocato `greeting2.toUpperCase()`, cosa contengono `greeting` e `greeting2`?

Risposte

24. Ora `greeting and greeting2` si riferiscono entrambi allo stesso oggetto `String`.
25. Entrambe le variabili si riferiscono ancora alla stessa stringa e la stringa non p stata modificata. Il metodo `toUpperCase` costruisce una nuova stringa che contiene caratteri maiuscoli, lasciando la stringa originale immutata.