

OOP

Ambiente di programmazione Java, Analisi di un semplice programma,
Programma, Errori comuni

Mettere in esecuzione un ambiente di sviluppo Java

CONSIGLIO:

Scaricate GEANY dal sito → <https://www.geany.org>

È Quello che utilizzeremo anche in laboratorio è utile avere dimestichezza.

Altrimenti esiste anche BlueJ (graficamente più intuitivo).

Non cambia assolutamente nulla a livello di sintassi.

Sono entrambi due editor di testo!

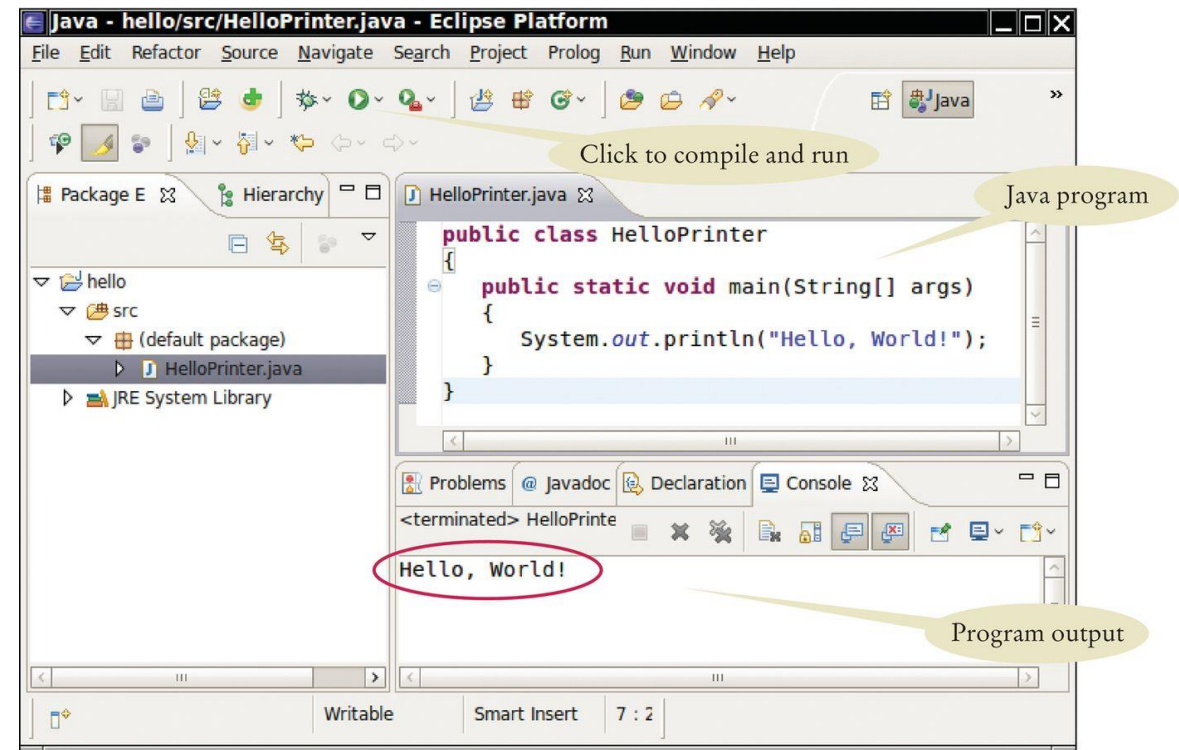
A noi interessa prima di tutto capire come scrivere il programma. 😊

JAVA

- Un editor è un programma per l'immissione e la modifica di testo, come un programma Java.
- Java fa distinzione tra maiuscole e minuscole.
- Il compilatore Java traduce il codice sorgente in file di classe.
- I file di classe contengono istruzioni per la macchina virtuale Java.

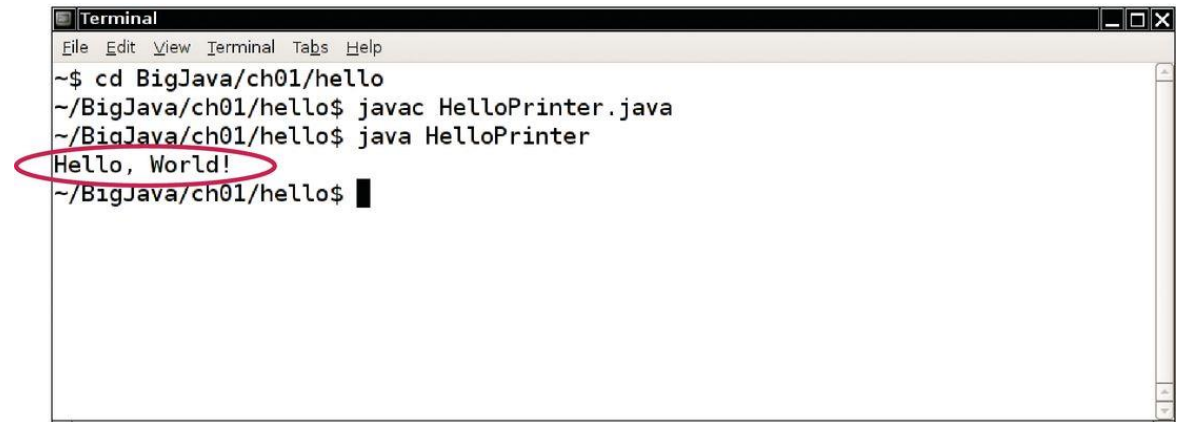
Esecuzione del programma
HelloPrinter in un ambiente di
sviluppo integrato

JAVA



Esecuzione del programma
HelloPrinter in una finestra di
terminale (Prompt dei comandi)

JAVA

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal shows the following commands and output:

```
~$ cd BigJava/ch01/hello
~/BigJava/ch01/hello$ javac HelloPrinter.java
~/BigJava/ch01/hello$ java HelloPrinter
Hello, World!
~/BigJava/ch01/hello$
```

The output "Hello, World!" is circled in red. The prompt character is a black square.

JAVA

Caratteristica fondamentale del linguaggio Java è **la portabilità**, cioè la capacità di un programma (scritto in Java) di tenere lo stesso comportamento in modo indipendente dalla piattaforma in cui il programma stesso viene eseguito.

Tutto questo grazie alla **Java Virtual Machine (JVM)**.

La definizione:

«La Java Virtual Machine è una macchina immaginaria (astratta) la cui implementazione può essere effettuata attraverso l'utilizzo di un software di emulazione che venga eseguito su una macchina reale. I programmi che una JVM sono in grado di eseguire devono essere scritti in appositi file con estensione .class, ognuno dei quali deve contenere al suo interno il codice per, al massimo, una classe pubblica.»

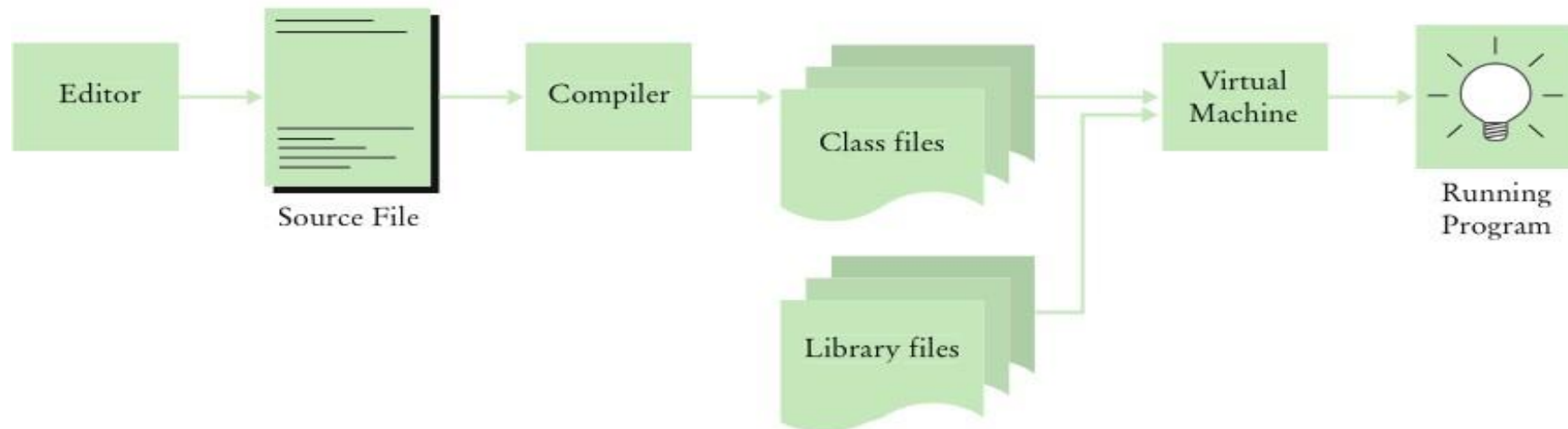
JAVA

il cammino che viene seguito dalla scrittura del codice alla sua esecuzione, può essere identificato nei seguenti passi (vedi immagine successiva):

1. Viene scritto il codice sorgente utilizzando la sintassi propria del linguaggio in questione
2. Il codice sorgente viene, quindi, analizzato dal compilatore del linguaggio il quale (se non vi sono errori di sintassi) produce un codice macchina (object code), in stretta relazione al sistema operativo e all'hardware in uso.
3. Il codice macchina viene "dato in pasto" ad un linker, il quale si occupa di effettuare il collegamento con eventuali librerie esterne richiamate dal codice sorgente e di produrre, al termine, un file eseguibile (in Windows un .exe).

La Java Virtual Machine è che una sorta di processore in grado di interpretare un particolare codice macchina denominato bytecode (contenuto all'interno dei file .class), che rappresenta il prodotto della compilazione di un file sorgente scritto in linguaggio Java.

Dal codice sorgente al programma in esecuzione



JAVA

Il comando per generare un bytecode, a partire da un sorgente java è il seguente:

```
javac nomeapplicazione.java
```

Quando, invece, si vuole mandare in esecuzione un'applicazione (che contenga al suo interno l'implementazione del metodo `main()`) viene utilizzata la riga di comando:

```
java nomeapplicazione
```

Ogni volta che si vuole eseguire un programma java, viene creata un'istanza di una virtual machine in grado di interpretare il contenuto di un bytecode ed eseguire, al suo interno, l'applicazione stessa. È importante sapere che ogni applicazione Java in esecuzione dà vita ad una nuova istanza di una virtual machine, che viene dismessa al termine dell'esecuzione stessa.



JAVA

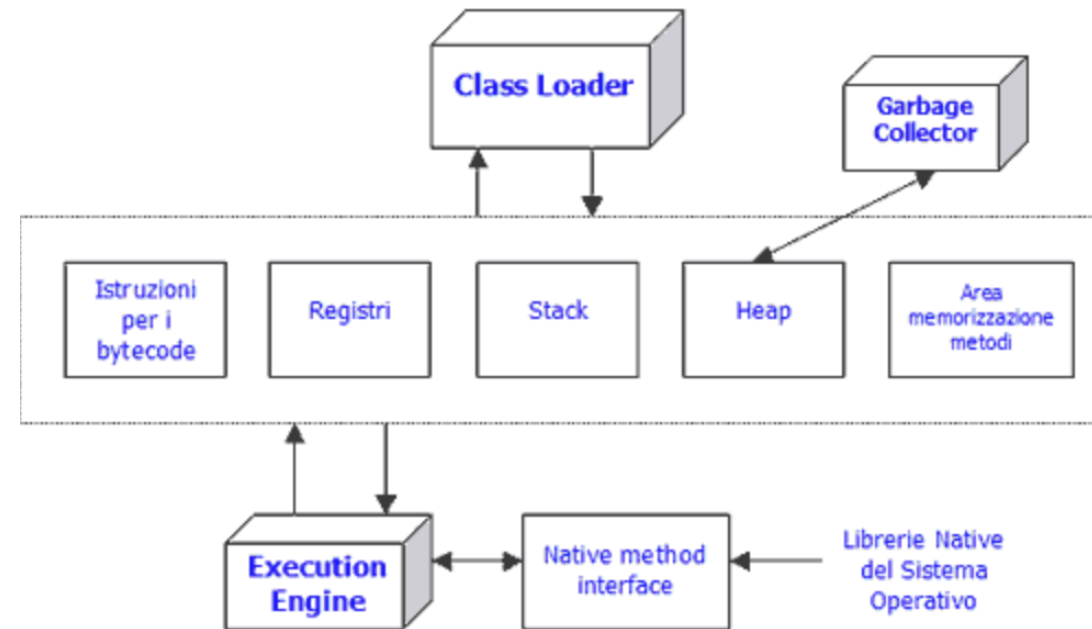
Per consentire, dunque, ad un programma Java di essere "portabile" su una particolare piattaforma basterà installare una JVM all'interno di una macchina reale (basata sulla piattaforma in questione) e delegare alla JVM stessa il compito di interpretare il bytecode in modo opportuno ed efficace per l'hardware in uso.

JVM è responsabile della creazione degli oggetti e la "pulizia" della memoria, svolta attraverso la routine di garbage collection.

JVM

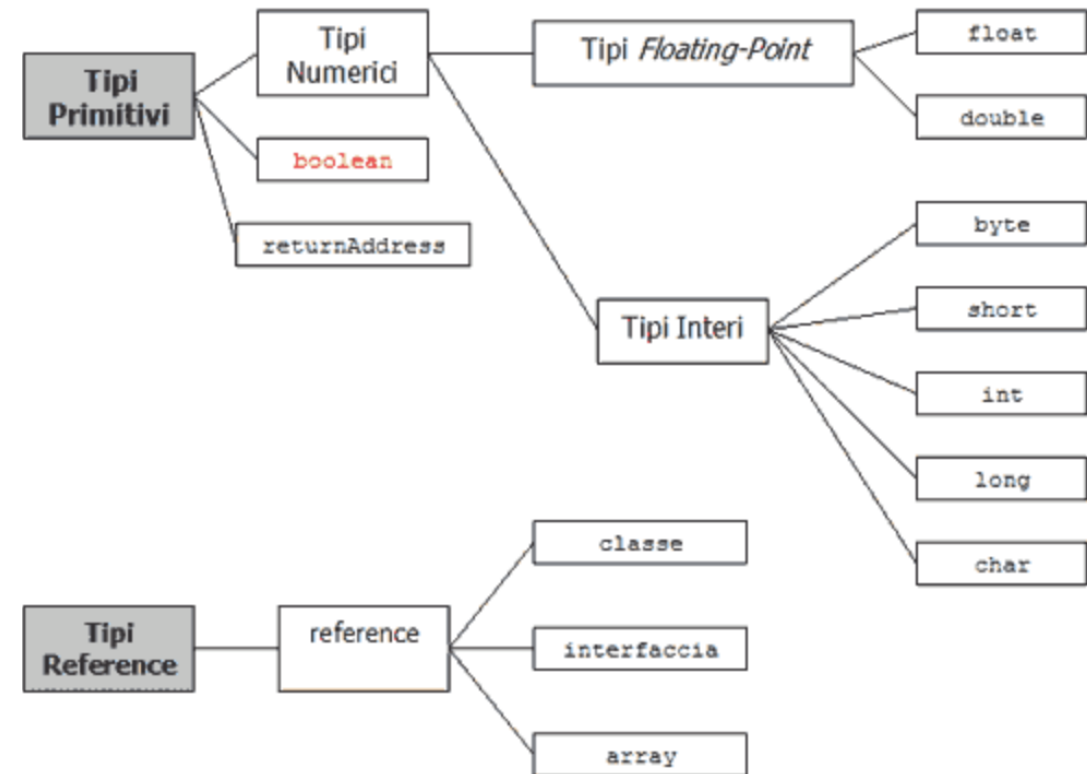
Il Class Loader

- Un programma Java è solitamente organizzato in più classi, ognuna delle quali è responsabile di un particolare compito all'interno dell'applicazione. Come detto, ognuna di queste classi, al termine della compilazione, viene memorizzata in un file con estensione .class.
- I file .class non vengono caricati in memoria tutti in una volta, alla partenza dell'applicazione ma, piuttosto, vengono caricati su richiesta ogni volta che sia necessario utilizzare una specifica classe. Il Class Loader è il componente della Java Virtual Machine che si occupa del caricamento in memoria delle classi.



JVM

Esistono, come per il linguaggio Java, due tipi di dati su cui opera la Java Virtual Machine: i tipi di dati primitivi e i tipi di dati reference. Lo schema seguente illustra in dettaglio tale suddivisione:



JVM

Dati primitivi:

- non è presente, tra i tipi primitivi, il tipo `boolean` (in rosso).
- Esiste, inoltre, un altro tipo di dato primitivo numerico noto come `returnAddress` ("tipo di ritorno") che viene utilizzato dalle istruzioni che indicano il ritorno da una funzione al programma principale. Un `returnAddress` rappresenta un puntatore ad un opcode (con tale termine si intende, in generale, un codice numerico che rappresenta una particolare istruzione eseguita dal processore) relativo ad un'istruzione della Java Virtual Machine.

I tipi reference utilizzati sono suddivisi in tre categorie:

- Riferimenti a classi
- Riferimenti a interfacce
- Riferimenti ad array
- Esiste anche il riferimento a nessun oggetto, nel qual caso il valore del reference è definito come `null`.

Analisi di un semplice programma

```
1 public class HelloPrinter
2 {
3     public static void main(String[] args)
4     {
5         // visualizza un saluto nella finestra di
console
6
7         System.out.println("Hello, World!");
8     }
9 }
```

Analisi di un semplice programma

Le classi sono gli elementi fondamentali dei programmi

Java: Dichiarazione di una classe chiamata HelloPrinter

```
1 public class HelloPrinter
```

In Java, ogni file sorgente può contenere al massimo una classe pubblica.

Il nome della classe pubblica deve corrispondere al nome del file contenente la classe:

La classe HelloPrinter deve essere contenuta in un file denominato HelloPrinter.java

Analisi di un semplice programma

- Ogni classe contiene dichiarazioni di metodi.
- Ogni metodo contiene una sequenza di istruzioni.
- Un metodo contiene una raccolta di istruzioni di programmazione che descrivono come eseguire un determinato compito.
- Un metodo viene «invocato» specificando il metodo e i suoi argomenti.

Analisi di un semplice programma

- Ogni applicazione Java contiene **una classe con un metodo principale**
- All'avvio dell'applicazione, vengono eseguite le istruzioni nel metodo principale
- Dichiarare un metodo principale:

```
3    public static void main(String[] args)
4    {
5        ...
6    }
```

Analisi di un semplice programma

- Il corpo del metodo principale contiene istruzioni.
- Il nostro metodo ha una sola istruzione:
`System.out.println("Hello, World!");`
- Che stamperà Hello, World!

Analisi di un semplice programma

Abbiamo già fatto un'invocazione ad un metodo:

```
System.out.println("Hello, World!");
```

Una chiamata al metodo richiede:

- Il metodo che vuoi usare (in questo caso, `System.out.println`)
- Tutti i valori necessari al metodo per svolgere il proprio compito racchiusi tra parentesi (in questo caso, `"Hello, World!"`).

Il termine tecnico per tali valori è "argomenti"

System.out.println()

- Si può stampare valori numerici:

```
System.out.println(3 + 4);
```

Quindi si stamperà 7

Esiste un secondo metodo, System.out.print, che puoi utilizzare per stampare un elemento senza iniziare una nuova riga.

Oppure:

```
System.out.println("Hello, ");
```

```
System.out.println("World!");
```

Errori

- Un errore in fase di compilazione (errore di sintassi) è una violazione delle regole del linguaggio di programmazione rilevata dal compilatore.

```
System.ou.println("Hello, World!");
```

- Un errore di runtime (errore logico) fa sì che un programma esegua un'azione che il programmatore non intendeva.

```
System.out.println("Hello, Word!");
```

Errori

- Eccezione: un tipo di errore di runtime
- Questa istruzione:

```
System.out.println(1 / 0)
```

- Genera un messaggio di errore dalla macchina virtuale Java
"Division by zero"

Errori comuni

- Dimenticarsi il punto e virgola alla fine delle istruzioni
- Proviamo a scrivere un metodo main con qualche errore

Progettazione di algoritmi

Algoritmo: una sequenza di passi che:

- **arrivi a compimento (in un tempo finito)**
- **sia eseguibile**
- **sia non ambigua**

Progettazione di algoritmi


Il problema:

Vengono depositati 10.000 euro in un conto bancario che guadagna il 5% di interesse all'anno.

Quanti anni ci vogliono perché il saldo del conto sia il doppio dell'originale?
Calcoliamolo a mano


Calcoli a mano

Anno	Interesse	Saldo
0		10000
1	$10000 \times 0,05 = 500$	$10000 + 500 = 10500$
2	$10500 \times 0,05 = 525$	$10500 + 525 = 11025$
3	$11025 \times 0,05 = 551,25$	$11025 + 551,25 = 11576,25$
4	$11576,25 \times 0,05 = 578,81$	$11576,25 + 578,81 = 12155,06$



Descriviamo questo procedimento/algoritmo

1. Iniziare con Anno=0 e saldo=10000 euro
2. Ripetere i passi seguenti finchè il saldo è minore di 20000:
 - Aggiungere 1 all'anno
 - Calcolare l'interesse come saldo per 0,05 (5%)
 - Aggiungere l'interesse al saldo (→ sovrascrivo!)
3. Riportare il valore finale dell'anno come risposta.




Descriviamo questo procedimento/algoritmo

Proviamo a scrivere tutto con una pseudocodifica.

Pseudocodice: una descrizione informale di una sequenza di passaggi per risolvere un problema

1. Descrivi come viene impostato o modificato un valore:
 - $\text{costo totale} = \text{prezzo di acquisto} \times \text{costo operativo}$
 - Moltiplicare il valore del saldo per 1.05
2. Descrivere decisioni e ripetizioni:
 - Se $\text{il costo totale 1} < \text{costo totale 2}$
 - Finchè il saldo è minore di 20000



Descriviamo questo procedimento/algoritmo

3. Usa il rientro per indicare quali affermazioni devono essere selezionate o ripetute:
 - Per ogni automobile
 - costo operativo= $10 \times$ costo annuo
 - costo totale= prezzo di acquisto + costo operativo
4. Indicare risultati:
 - Riportare come risposta il valore finale dell'anno



Esercizio 1:

Dovete scegliere l'acquisto tra due automobili, una delle quali è più efficiente dell'altra per quanto riguarda il consumo del carburante ma è più costosa. Vi sono noti prezzo ed efficienza (in km per gallone) di entrambe le vetture.

Pensate di usare l'auto per 10 anni percorrendo 15000 km l'anno, con un prezzo di carburante di 4 euro al gallone.

Non ci sono finanziamenti.

Qual è l'acquisto migliore?



Esercizio 2:

Supponiamo che il tuo gestore di telefonia mobile ti addebiti euro 29,95 per un massimo di 300 minuti di chiamate e euro 0,45 per ogni minuto aggiuntivo, più il 12,5% di tasse e commissioni. Fornire un algoritmo per calcolare l'addebito mensile per un determinato numero di minuti.

Utilizzare oggetti

Variabili, tipi, commenti, assegnazioni, invocare metodi....



Oggetti e classi

Ogni parte utilizzata da un costruttore di case, come un forno o uno scaldabagno, svolge una funzione particolare. Allo stesso modo, crei programmi da oggetti, ognuno dei quali ha un comportamento particolare.

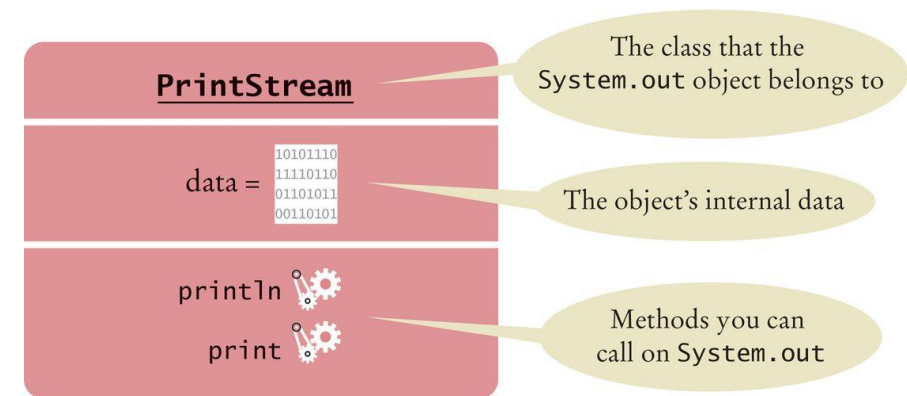
- In Java, creiamo programmi per oggetti.
- Ogni oggetto ha determinati comportamenti.
- Si Può manipolare l'oggetto per ottenere determinati effetti

Classi e oggetti

- Oggetto: un'entità nel tuo programma che puoi manipolare chiamando uno o più dei suoi metodi.
- Metodo: consiste in una sequenza di istruzioni che possono accedere ai dati di un oggetto.
 - Non sai quali sono le istruzioni
 - Sai che il comportamento è ben definito
- System.out ha un metodo println
 - Non sai come funziona
 - L'importante è che faccia il lavoro che gli chiedi

Classi e oggetti

Puoi pensare a uno scaldabagno come a un oggetto in grado di eseguire il metodo "ottenere acqua calda". Quando chiami quel metodo per goderti una doccia calda, non ti interessa se lo scaldabagno utilizza il gas o l'energia solare.



Classe:

- Una classe descrive un insieme di oggetti con lo stesso comportamento.
- Alcuni oggetti stringa
 - "Ciao mondo"
 - "Arrivederci"
 - "Mississippi"
- Puoi invocare gli stessi metodi su tutte le stringhe.
- `System.out` è un membro della classe `PrintStream` che scrive nella finestra della console.
- Puoi costruire altri oggetti della classe `PrintStream` che scrivono su destinazioni diverse.
- Tutti gli oggetti `PrintStream` hanno i metodi `println` e `print`.

Classe:

- Gli oggetti della classe `PrintStream` hanno un comportamento completamente diverso rispetto agli oggetti della classe `String`.
- Classi diverse hanno responsabilità diverse
 - Una stringa conosce le lettere che contiene
 - Una stringa non sa come inviare se stessa ad una finestra o ad un file della console.

Variabili



- Utilizzare una variabile per memorizzare un valore che si desidera utilizzare in seguito
- Per dichiarare una variabile denominata larghezza

`int larghezza = 5;`

Come una variabile in un programma per computer, un parcheggio ha un identificatore e un contenuto.

Variabili: sintassi

nomeTipo nomeVariabile= valore;

Oppure

nomeTipo nomeVariabile;

Posso inizializzare la
variabile ma non è
obbligatorio

String saluto= "Hello, World!" ;

Il punto e
virgola è
indispensabile
per terminare
questo tipo di
istruzione

Il tipo specifica cosa si
può fare con il valore
salvato nella variabile

Il nome della variabile
deve avere senso con
quello in cui ha salvato.

Variabili

- Una variabile è una posizione di archiviazione
- Ha un nome e contiene un valore
- Quando si dichiara una variabile, di solito si specifica un valore iniziale.
- Quando si dichiara una variabile, si specifica anche il tipo dei suoi valori.

Variabili

Dichiarazione variabile:

`int larghezza = 20;`

- la larghezza è il nome
- int è il tipo
- 20 è il valore iniziale

ES: Ogni posto auto è adatto ad un particolare tipo di veicolo, così come ogni variabile contiene un valore di un particolare tipo.



© Ingenui/iStockphoto.

Variabili

Nome della variabile	Significato
<code>int larghezza = 20;</code>	dichiara una variabile intera e la inizializza a 20
<code>int perimetro= 4* larghezza;</code>	il valore iniziale non è un valore fisso (ovviamente larghezza deve essere dichiarata → altrimenti errore)
<code>String saluto="Ciao"</code>	variabile di tipo String inizializzata con il valore "Ciao"
<code>altezza= 30</code>	ERRORE: manca il tipo. È un assegnamento, non dichiarazione
<code>int altezza= "30"</code>	ERRORE: sto inizializzando un variabile intera con una stringa
<code>int larghezza;</code>	dichiara una variabile intera
<code>int larghezza, altezza;</code>	dichiara due variabili intere

Variabili

- Si usa il tipo `int` per i numeri che non possono avere una parte frazionaria.
`int larghezza = 20;`
- Si utilizza il tipo `double` per i numeri in virgola mobile.
`double migliaPerGallone = 22,5;`
- I numeri possono essere combinati da operatori aritmetici come `+`, `-` e `*`
- Un altro tipo è `String`
`String saluto = "Ciao";`
- Un tipo specifica le operazioni che possono essere eseguite con i suoi valori.
 - Puoi moltiplicare il valore della `'larghezza'` per un numero
 - Non puoi moltiplicare i saluti per un numero.

Nomi delle variabili



- Scegli un nome per una variabile che ne descriva lo scopo.
- Regole per i nomi di variabili, metodi e classi:
 - Deve iniziare con una lettera o il carattere di sottolineatura (_) e i caratteri rimanenti devono essere lettere, numeri o trattini bassi.
 - Non è possibile utilizzare altri simboli come ? o % o uno spazio



Nomi delle variabili

- Usa le lettere maiuscole per indicare i limiti delle parole, come in `migliaPerGallone`.
- I nomi fanno distinzione tra maiuscole e minuscole

Non puoi usare parole riservate come `double` o `class`

Per convenzione Java:

- i nomi delle variabili iniziano con una lettera minuscola.
- i nomi delle classi iniziano con una lettera maiuscola.

Nomi delle Variabili

Nome della variabile	Commento
distanza_1	i nomi sono costituiti da lettere, numeri e caratteri di sottolineatura
x	come in matematica è lecito usare nomi di incognite ma poco comprensibile
CanVolume	Attenzione: è diverso da canVolume e viola la convenzione di Java
6pack	ERRORE: non può iniziare con un numero
can volume	ERRORE: non può contenere spazi
double	ERRORE: non può essere una parola riservata di Java
miglia/gallone	ERRORE: non si possono usare simboli come /

Commenti

- Usa i commenti per aggiungere spiegazioni per gli **umani** che leggono il tuo codice.

```
double milesPerGallon = 33.8;
```

```
// The average fuel efficiency of new U.S. cars in 2011
```

- Il compilatore non elabora i commenti
- Ignora tutto da `//` alla fine della riga.
- Per un commento più lungo, racchiuderlo tra i delimitatori `/*` e `*/`.
- Il compilatore ignora questi delimitatori e tutto il resto

Commenti

Esempio di commenti più lunghi:

```
/*  
In most countries, fuel efficiency is measured in  
liters per hundred kilometer. Perhaps that is more  
useful—it tells you how much gas you need to  
purchase to drive a given distance. Here is the  
conversion formula.  
*/  
double fuelEfficiency = 235.214583 / milesPerGallon
```


Assegnazioni

- Si utilizza l'operatore di assegnazione (=) per modificare il valore di una variabile.

```
int width = 10;
```

- Per modificare il valore della variabile, assegnare il nuovo valore

```
width = 20;
```

Assegnazioni

- È un errore utilizzare una variabile a cui non è mai stato assegnato un valore:

- Il compilatore si lamenterà di una "variabile non inizializzata»

```
int height;
```

```
int width = height; // ERROR - uninitialized variable height
```

- Rimedio: assegnare un valore alla variabile prima di utilizzarla.

```
int height = 20;
```

```
int width = height; // OK
```

- Tutte le variabili devono essere inizializzate prima di accedervi.

Assegnazioni

- Il lato destro del simbolo = può essere un'espressione matematica:

```
width = height + 10;
```

- Questo significa calcolare il valore dell'height + 10
- memorizzare quel valore nella width variabile

```
width = width + 10;
```

- L'operatore di assegnazione = non denota l'uguaglianza matematica

Syntax *variableName = value;*

The diagram illustrates the syntax of variable declarations and assignments. It shows a sequence of statements separated by semicolons. The first statement is `double width = 20;`, where `double` is highlighted in red. A callout bubble points to it, stating "This is a variable declaration." The second statement is `width = 30;`. A callout bubble points to it, stating "This is an assignment statement." Another callout bubble points to the `width` in this statement, stating "The value of this variable is changed." A third callout bubble points to the `30`, stating "The new value of the variable". The third statement is `width = width + 10;`. A callout bubble points to it, stating "The same name can occur on both sides. See Figure 4."

```
double width = 20;  
.  
.  
width = 30;  
.  
.  
width = width + 10;
```

This is a variable declaration.

This is an assignment statement.

The value of this variable is changed.

The new value of the variable

The same name can occur on both sides. See Figure 4.

Assegnazioni