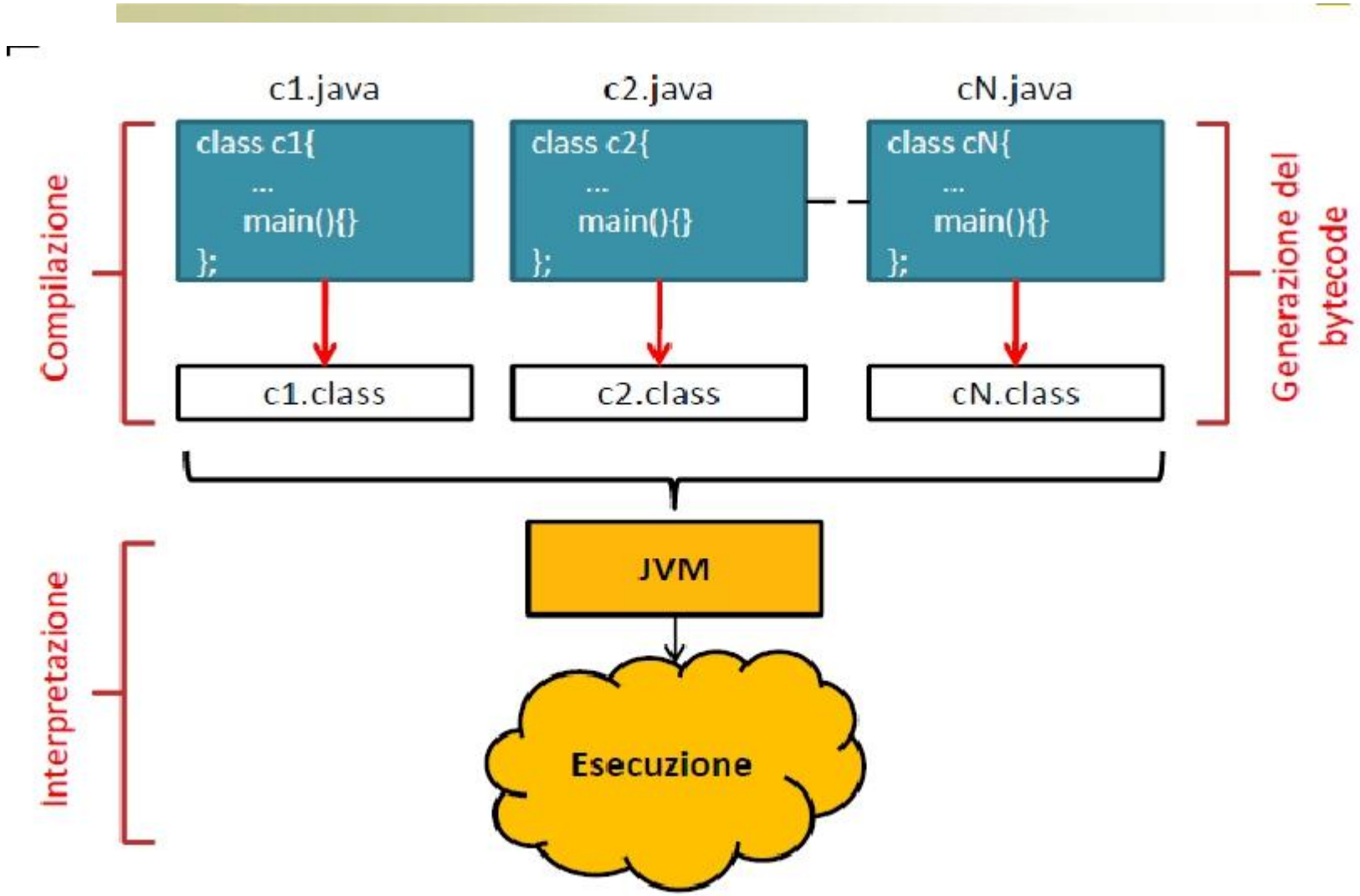
Java: Compilatore e Interprete



Java Virtual Machine

• Il bytecode non è Linguaggio Macchina. Per diventarlo, deve subire un'ulteriore trasformazione che viene operata dall'interprete Java in modalità JIT (Just In Time)

Pro: Indipendente dalla piattaforma (portabilità)
Contro: L'ulteriore trasformazione rallenta l'esecuzione

• La JVM esegue il file specificato dall'utente (esempio nel lucido successivo), che a sua volta deve contenere un metodo main().

La JVM cerca i file (.class) nel momento in cui servono (collegamento dinamico).

È possibile specificare il percorso nel quale cercare tramite la variabile classpath.

Il Linguaggio Java

•È un linguaggio totalmente ad oggetti

• Si basa strettamente sul concetto di classe e di oggetto

• Un programma è un insieme di classi. Anche il main è in una classe!

• Il linguaggio definisce alcuni tipi primitivi/di base

Tipo primitivo	Descrizione
boolean	valori che possono essere true e false
char	caratteri sono di 16 bit e codificati Unicode
byte	interi di 8 bit con segno e codifica in complemento a due
short	interi di 16 bit con segno e codifica in complemento a due
int	interi di 32 bit con segno e codifica in complemento a due
long	interi di 64 bit con segno e codifica in complemento a due
float	reali di 32 bit in virgola mobile (IEEE 754)
double	reali di 64 bit in virgola mobile (IEEE 754)

Il Linguaggio Java

- Una classe Java può:
  - contenere dati/operazioni proprie
  - definire un tipo di dato astratto e fungere da "stampo" per creare nuovi oggetti

- Una classe Java è un'entità sintatticamente in grado di contenere sia i dati che le funzioni che operano su quest'ultimi. Sia i dati che le operazioni hanno differenti livelli di visibilità/protezione:

  pubblico: visibile anche dall'esterno della classe (keyword public)
  privato: visibile solo dentro la classe (keyword private)

- Java impone la corrispondenza fra nome di una classe pubblica e nome del file in cui essa deve essere definita.

- E' essenziale rispettare maiuscole/minuscole (Java è case-sensitive)

Classe

Una classe public viene definita utilizzando la seguente notazione:

public class *NomeClasse* {....}

All'interno del costrutto class,è necessario dichiarare:

Metodi e dati associati alla classe
Metodi e dati associati agli oggetti

Nel primo caso, i dati appartengono alla classe e sono unici all'interno della JVM. Nel secondo caso, invece, essi appartengono ad ogni singola istanza.

value dichiarato a livello di oggetto

```
public class Esempio1 {
    private int value;
    ....
}
```

value dichiarato a livello di classe

```
public class Esempio1 {
    private static int value;
    ....
}
```

Classe

In definitiva, la dichiarazione di una variabile segue la sintassi:

[private|public] [static] *Tipo Nome;*

mentre, per dichiarare un metodo, usiamo la sintassi:

[private|public] [static] *TipoDiRitorno Nome(TArg1 nome1,…)*{
**   *Istruzioni da eseguire***
};

Esempio

```
public class Esempio1 {
    public int getValue(){…}
    public void setValue(int newValue){…}
}
```

Classe

```
public class Esempio1 {
  private int value;
  public int getValue() {
          return value;
  }
  public void setValue(int newValue){
          value = newValue;
  }
}
```

• La classe Esempio1 dichiara un unico campo (value) privato e quindi non accessibile dall'esterno. Ogni oggetto di tipo Esempio1 ha un proprio campo value

•Gli oggetti di tipo Esempio1 espongono un metodo getValue() che ritorna il valore attuale di value

•Gli oggetti di tipo Esempio1 espongono un metodo setValue(int newValue) che consente di impostare il valore di value

Oggetto

Una classe può fungere da “stampo” per creare oggetti. **Per creare un oggetto, bisogna:**

Definire la classe che modella il nuovo tipo di dato

Definire un riferimento, il cui tipo è il nome della classe modello

Creare dinamicamente l'oggetto tramite l'operatore new

Ipotizzando che sia stata già definita la classe Esempio1:

```
Esempio1 esempio; // def. del riferimento
esempio = new Esempio1(); // creazione oggetto
```

O, in maniera più compatta:

```
Esempio1 esempio = new Esempio1();
```

Per invocare i metodi offerti da un oggetto, si usa la notazione puntata **riferimento.nomeMetodo(…)**.

Ad esempio, per invocare getValue() sull’oggetto esempio, scriveremo esempio.getValue();

Esecuzione di un programma

Un programma Java è un insieme di classi e oggetti:

Le classi sono componenti statici, che esistono già all'inizio del programma

Gli oggetti sono invece componenti dinamici, creati su necessità e durante l'esecuzione

Il più semplice programma Java è costituito da una singola classe che, come minimo, dovrà definire una singola funzione statica, il main():

```
public class Esempio0 {
    public static void main(String args[]) {
        System.out.println("Hello World!");
    }
}
```

Per convenzione, in Java i nomi delle classi iniziano sempre con la lettera maiuscola e i nomi dei singoli campi (dati e funzioni) iniziano sempre con la lettera minuscola.

Java main

Il main() in Java è una funzione pubblica con la seguente struttura:

public static void main(String args[]) {…}

Deve essere obbligatoriamente dichiarato:

public: Il main deve essere visibile all'esterno della classe per consentire l'invocazione da parte della JVM

static: È statico dato che esiste ancora prima di qualunque oggetto

void: Non deve avere valore di ritorno

String args[]: Deve sempre prevedere gli argomenti dalla linea di comando, anche se non vengono usati, sotto forma di array di String

Esempio

```
public class Esempio {
        public static void main(String args[]) {
                System.out.println("Ciao Mondo!");
        }
}
```

Importante:

- L'uso della notazione puntata System.out.println("Hello World") invoca il metodo println(String input) sull'oggetto out, che è un campo (statico) della classe System predefinita in Java

- Sia il main che la classe devono essere public per consentire l'invocazione da parte della JVM

Un altro esempio

```
public class Esempio2 {
    public static void main(String args[]) {
            int x = 3, y = 4;
            int z = x + y;
            System.out.println("La somma vale: "+z);
    }
}
```

Importante:

- L’operatore + è utilizzato per la concatenazione delle stringhe

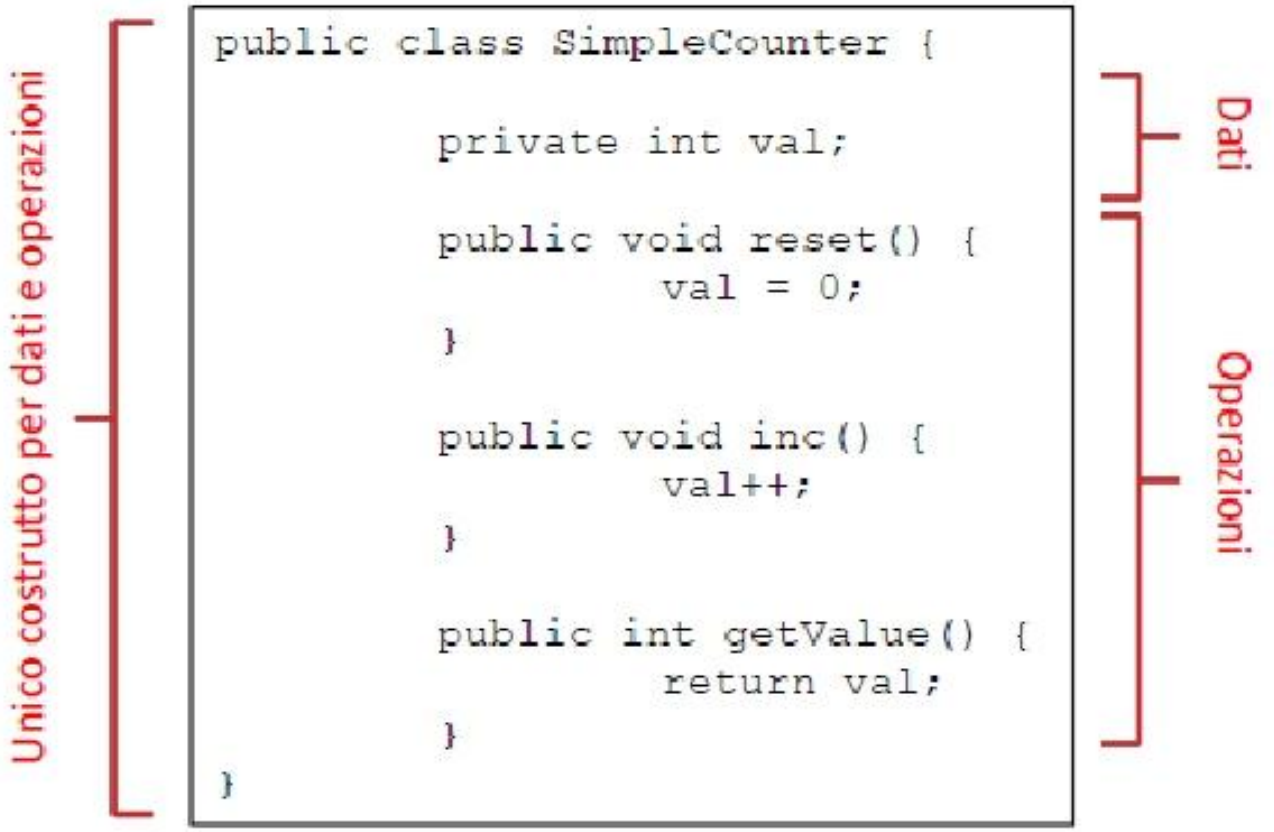
- La variabile z è di tipo int, ma viene implicitamente convertita nella stringa associata

Esempio: creazione Oggetto contatore

Vogliamo creare un oggetto contatore. Ogni contatore deve avere un proprio valore, e deve offrire tre metodi:

inc(): consente di incrementare il valore attuale del contatore
reset(): consente di portare a zero il valore del contatore
getValue(): consente di reperire il valore attuale del contatore

SimpleCounter.java



Il campo val è privato (keyword private): può essere acceduto solo dalle operazioni definite nella medesima classe (reset, inc, getValue). Si garantisce l'incapsulamento.

Usiamo SimpleCounter

Inseriamo un main nella classe SimpleCounter.

All'interno, creiamo un oggetto ed invochiamo i suoi metodi.
Infine, recuperiamo il valore attuale e lo stampiamo

```
public static void main(String args[]){
        SimpleCounter counter = new SimpleCounter();
        counter.reset();
        counter.inc();
        counter.inc();
        int x = counter.getValue();
        System.out.println("Il contatore vale "+x);
}
```

Cosa viene stampato?

Usiamo SimpleCounter

E se invece …

```
public static void main(String args[]){
    SimpleCounter counter1 = new SimpleCounter();
    SimpleCounter counter2 = new SimpleCounter();
    counter1.inc();
    counter1.inc();
    counter2.reset();
    int x1 = counter1.getValue();
    int x2 = counter2.getValue();
    System.out.println("Il contatore1 vale "+x1);
    System.out.println("Il contatore2 vale "+x2);
}
```

Cosa viene stampato?