Fondamenti di Java

Il linguaggio cult per la programmazione a oggetti in quattro lezioni

La nascita, la metodologia, lo sviluppo e le caratteristiche uniche che hanno fatto di Java il linguaggio più usato in rete

La programmazione tradizionale imperativa

La programmazione imperativa si basa su alcuni punti fondamentali:

- Il teorema di Bohm-Jacopini
- L'algoritmo risolutivo
- La programmazione top-down

Si presume dunque che per il programmatore sia sempre possibile ipotizzare l'evoluzione del programma e codificarla in un numero finito di passi prima della fase di esecuzione(run-time)

Un nuovo modo di fare programmazione: la object oriented programming

Nei programmi moderni, complessi e caratterizzati da un alto grado di interazione con l'utente, non sempre è possibile elaborare un algoritmo risolutivo efficace.

L'approccio attuale consiste nel modellare la realtà che si vuole rappresentare nel programma attraverso moduli software che descrivono lo stato e il comportamento dei diversi soggetti che ne fanno parte.

I moduli realizzati possono interagire fra loro scambiandosi messaggi ed essere utilizzati in programmi diversi

Il progetto

- Nella programmazione a oggetti un programma si può immaginare come il modello di una situazione reale.
- Ad esempio un programma di geometria deve descrivere entità come triangoli, quadrati, ecc., mentre il programma di gestione di un negozio deve descrivere caratteristiche e comportamento di clienti, prodotti e così via.

L'approccio orientato agli oggetti

- Nella programmazione a oggetti ogni aspetto della realtà che voglio rappresentare diventa un modulo software del tutto indipendente e riutilizzabile in altri contesti.
- Questi moduli software prendono il nome di classi e sono le unità base della programmazione a oggetti.

Le classi

- Le classi sono unità complete, che racchiudono tutti i dati che descrivono lo stato di un oggetto e tutte le azioni che ne descrivono il comportamento.
- Nei linguaggi completamente orientati agli oggetti come Java, tutto il codice deve essere scritto nelle classi.
- Nei linguaggi detti ibridi (Visual Basic, C++, C#) è possibile introdurre anche codice esterno alle classi

Attributi e metodi

- Le classi contengono al loro interno attributi e metodi.
- I dati che descrivono le caratteristiche di un oggetto sono conservati all'interno della classe in opportune variabili dette variabili di classe o attributi della classe.
- I comportamenti possibili per gli oggetti sono descritti nelle classi da opportune funzioni (o procedure) che prendono il nome di metodi.

Le classi e gli oggetti

- La classe è dunque una descrizione teorica delle caratteristiche e delle capacità (in termini di azioni) di una certa categoria.
- Assegnando dei valori specifici agli attributi della classe è possibile ricavare da questo modello teorico un oggetto specifico.
- Questa operazione di assegnazione è compiuta da metodi speciali denominati costruttori, proprio perché permettono la costruzione di un oggetto a partire dalla classe

La comunicazione tra i moduli

- Riassumendo, un programma orientato agli oggetti si costruisce creando diversi moduli indipendenti e facendoli interagire tra di loro
- I moduli possono interagire scambiandosi messaggi
- Attributi = stato dell'oggetto
- Metodi = comportamento dell'oggetto

Variabili di classe e variabili di istanza

- I metodi sono dunque funzioni o procedure che operano sui dati descritti dagli attributi della classe
- Tutte le altre variabili necessarie all'implementazione dei metodi e dichiarate nei metodi stessi sono detti variabili di istanza
- Ogni istruzione nei linguaggi OOP deve essere inserita in un metodo opportuno per poter essere eseguita.

Un esempio: la classe Rettangolo

```
class Rettangolo {
  double base;
  double altezza;

  double calcolaPerimetro(){
    return 2*(base+altezza);}

  double calcolaArea(){
    return base*altezza;}
}
```

Il linguaggio Java

- Java è un linguaggio orientato agli oggetti, derivato dal C++, e creato da James Gosling e altri ingegneri della Sun Microsystems .
- Per facilitare il passaggio dal C al nuovo linguaggio, la sintassi base del C in termini di strutture di controllo, operatori, tipi primitivi e simboli è stata mantenuta pressoché identica.

Caratteristiche di Java

- Java è un linguaggio Object Oriented
- È indipendente dalla piattaforma
- È ottimizzato per lavorare in rete, grazie alla presenza di opportuni strumenti e package
- È progettato per eseguire codice da sorgenti remote in modo sicuro

Write once, run everywhere

- L'indipendenza dalla piattaforma, significa si dovrebbe essere in grado di scrivere il programma una volta e farlo eseguire su qualunque sistema operativo.
- Questo è possibile con la compilazione del codice di Java in un linguaggio intermedio detto bytecode, basato su istruzioni semplificate che ricalcano il linguaggio macchina. Il bytecode verrà quindi eseguito da una macchina virtuale.
- Oggi si parla di portabilità di un componente software quando questo può essere riutilizzato su piattaforma diverse senza richiedere alcuna modifica.
- La Sun ha publicizzato questa caratteristica del linguaggio con lo slogan *"write once, run everywhere“*

Write once, debug anywhere

- La portabilità è un obiettivo tecnicamente difficile da raggiungere. Sebbene sia in effetti possibile scrivere in Java programmi che si comportano in modo consistente attraverso molte piattaforme hardware diverse, bisogna tenere presente che questi dipendono dalle macchine virtuali, che sono a loro volta dei programmi e hanno inevitabilmente i loro bug.

- Per questo è nata una parodia dello slogan della Sun "write once, run everywhere“ (“Scrivi una volta, esegui dovunque“), diventato *"write once, debug anywhere“.*

L'interprete di Java

- Nelle prime implementazioni il linguaggio Java era esclusivamente interpretato,per garantire la massima portabilità.
- Il programma interprete del bytecode prendeva il nome di Java Virtual Machine e questa soluzione era definita che interpretava il per ottenere, definita *Architecture Neutral*. Questa soluzione si è però rivelata poco efficiente, in quanto i programmi interpretati erano molto lenti.

La compilazione just in time

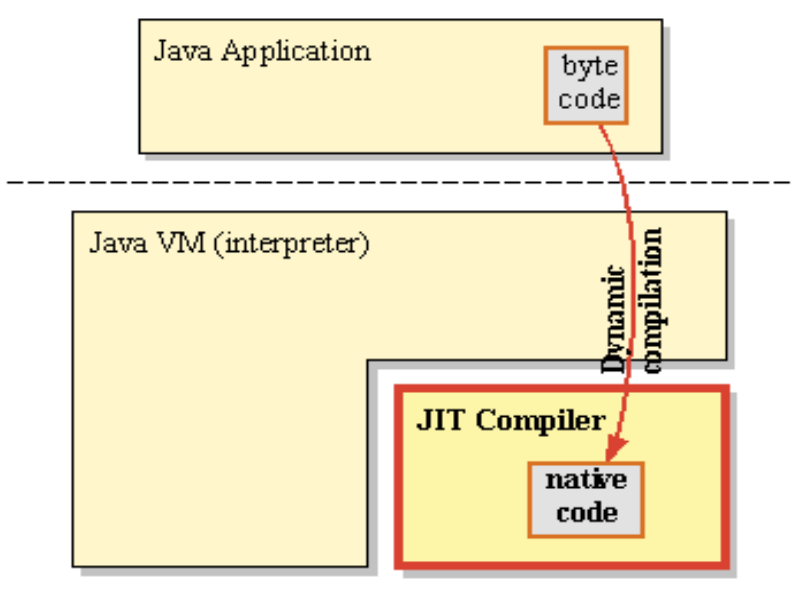
- Per ovviare al problema, le recenti implementazioni di Java hanno incorporato un compilatore interno, che al momento del lancio traduce al volo il programma bytecode Java in un normale programma nel linguaggio macchina del computer ospite. Questo programma è chiamato compilatore *just in time* o compilatore dinamico (*JIT compiler*).

- La compilazione è definita dinamica, perchè la macchina virtuale analizza il modello di esecuzione del codice allo scopo di ottimizzare le parti più frequentemente ripetute mentre il programma è in esecuzione.

- Al prezzo di una piccola attesa in fase di lancio del programma, il JIT compiler permette di avere delle applicazioni Java decisamente più veloci e leggere, anche se mano efficienti di quelle scritte in linguaggi compilati

Java Virtual Machine

- Il compilatore traduce il bytecode in linguaggio macchina in esecuzione



La creazione di un programma Java

- Per creare un programma Java occorre avere una Java Virtual Machine e un editor di testo
- Su un editor di testo si scrive il programma nella sintassi opportuna
- La compilazione si esegue da riga di comando con l'istruzione:

  `javac CiaoMondo.java`

  viene così creato il file `CiaoMondo.class` scritto in bytecode
- Il programma viene poi mandato in esecuzione con l'istruzione

  `java CiaoMondo`

  data nella stessa directory in cui si trova il file.

La fase di esecuzione

- Un programma Java è solitamente costituito da molte classi
- In ciascuna classe si trovano numerosi attributi e metodi
- Per risolvere le possibili ambiguità, l'esecuzione inizia sempre da una classe principale, riconoscibile perché porta sempre lo stesso nome del file
- Questa classe deve pertanto contenere un metodo main() che, come avviene per la funzione main() del C, è sempre ilprimo ad essere eseguito. Tutte le altre patri del programma si attivano tramite invocazione a cascata a partire dal metodo main()
- Nel caso precedente, l'esecuzione parte quindi dal metodo main() della classe CiaoMondo

Classe CiaoMondo

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

Classe Persona

```
public class Persona {
  String nome;
  int n;

  void presenta(){
  System.out.println("Buongiorno, mi chiamo
    "+nome+"e ho "+n+" anni");
  }
}
```

Classe ProvaPersona

```
public class ProvaPersona {
  public static void main(String[] args) {
    Persona p1=new Persona();
    p1.nome="Mario Rossi";
    p1.n=20;
    p1.presenta();
  }
}
```

Classe Cliente

```
public class Cliente {
  String nome;
  float soldi;
  void compra(float prezzo){
    if(soldi>=prezzo)
       soldi=soldi-prezzo;
    else
       System.out.println(“Credito insufficiente”);}
  void mostraCredito(){
  System.out.println(“Il cliente "+nome+" ha "+soldi+"
     euro");
  }
}
```

Classe Prodotto

```
public class Prodotto {
  String nome;
  float prezzo;
  void mostraPrezzo(){
   System.out.println("Il prodotto "+"
    costa " +prezzo);}
   }
}
```

Classe Vendita

```
public class Vendita {
  public static void main(String[] args) {
    Cliente c=new Cliente();
    Prodotto p=new Prodotto();
    c.nome="Mario Rossi";
    c.soldi=500;
    p.mostraPrezzo();
    c.compra(p.prezzo);
    c. mostraCredito();}
}
```