Programmazione Orientata agli Oggetti

Appunti delle Lezioni di Programmazione Orientata agli Oggetti

Anno Accademico: 2023/24

Giacomo Sturm

Dipartimento di Ingegneria Civile, Informatica e delle Tecnologie Aeronautiche
Università degli Studi "Roma Tre"

Sorgente del file LaTeX disponibile al seguente link:
https://github.com/00Darxk/Programmazione-Orientata-agli-Oggetti

Indice

1 Introduzione al Linguaggio Java

Java venne introdotto nel 1995, progettato per programmare dispositivi embedded, ma velocemente si utilizzò per scopi diversi da quelli descritti dai suoi creatori. Il team di Java sviluppò un sistema di embedded per i browser, le applet, delle piccole applicazioni che potevano essere eseguite all'interno del browser. Questo fu uno dei primi motivi per il successo di Java, un'altro aspetto importante per la diffusione di Java fu la sua caratteristica di semplificare aspetti di C e C++. Utilizzando un linguaggio più semplice diminuiva il costo associato ad un programmatore, per cui l'adozione di Java venne trainata dalle aziende alla ricerca di abbassare gli stipendi dei suoi dipendenti. In seguito venne utilizzato per creare la macchina virtuale su cui operano tutti i dispositivi Android.

Java è stato ideato con la capacità di mantenere la retro-compatibilità, nonostante le spinte di innovazione tecnologica che il linguaggio ha sostenuto negli anni. Nonostante la sua età Java è uno dei linguaggi di programmazione più usati dalle aziende. La diffusione più importante di Java si ebbe tra il '95 ed il '98.

Java fu il primo progetto industriale che introdusse un nuovo concetto di portabilità su scala industriale, un processore virtuale, poiché non viene progettato per un calcolatore fisico, ma la piattaforma Java Virtual Machine (JVM), per cui uno stesso programma Java può essere eseguito su tutte le piattaforme senza dover ricompilare nuovamente il codice per quella specifica piattaforma. Questa rappresenta una rivoluzione in ambito industriale, il codice oggetto Java "bytecode" prodotto dalla compilazione, come un file di estensione `.class`. Questo bytecode può quindi essere eseguito da qualsiasi piattaforma che dispone di un'implementazione della JVM. Invece programmi creati da un linguaggio più vecchio come C, producono un codice oggetto che presenta istruzioni macchina uniche alla piattaforma in cui è stato compilato. Astraendo il processore fisico si è reso possibile la creazione di programmi che non devono essere ricompilati per ogni piattaforma su cui deve essere eseguito.

Gli applet realizzati tramite Java, venivano eseguiti dalla JVM, e permise la creazione di primi siti web, rivoluzionando il paradigma di programmazione dell'epoca. La sintassi e la semantica del linguaggio sono descritte in un documento noto come Java Language Specification, per risolvere eventuali ambiguità. Inoltre fu uno dei primi linguaggi ad aver introdotto librerie incluse insieme al compilatore, sulla piattaforma Java.

1.1 Rapporto con il Linguaggio C

Il linguaggio C viene descritto come un linguaggio ad un medio livello di programmazione, di alto livello rispetto all'assembly, e permette di realizzare programmi ad alte prestazioni, utilizzato per scrivere sistemi operativi. Java invece venne introdotto in un ambiente diverso rispetto al linguaggio C, per cui si indica come un linguaggio di alto livello di astrazione, non è un linguaggio che cerca le migliori prestazioni, ma cerca di creare programmi semplici e portabili. Nel periodo in cui venne introdotto Java l'andamento delle prestazioni dei calcolatori permise di realizzare linguaggi che non avessero come scopo principale il risparmiare risorse del sistema. Il linguaggio C++ permette di avere prestazioni simili al linguaggio C, ma introducendo paradigmi realizzati in Java. Ma si preferisce per realizzare programmi ad alte prestazioni il C, rispetto al C++, poiché fornisce troppe scelte, complicando l'implementazione e l'uso del linguaggio.

In Java una dichiarazione di una variabile comprende la sua inizializzazione ad un valore nullo. Le dichiarazioni e assegnazioni in Java si ottengono mediante la stessa sintassi del linguaggio C. Si introduce un tipo `boolean` utilizzato per rappresentare i due possibili valori booleani `true` e `false`. Le descrizioni di tutti i tipi utilizzati in Java vengono definite nel JLS. Entrambi sono due linguaggi staticamente tipati, ovvero a tempo di compilazione deve essere noto il tipo di ogni dato utilizzato nel programma. Java introduce il tipo `String` utilizzato per rappresentare stringhe, non è un dato primitivo, e permette l'uso molto più semplice delle stringhe rispetto a C. Rappresenta un tipo particolare di classe, in appoggio al compilatore per favorire la loro gestione da parte del compilatore. Le stringhe vengono inizializzate a letterali stringhe, contenuti tra due doppi apici `" "`. Da Java 13 è possibile scrivere stringhe letterali multi-linea, utilizzando tripli doppi apici `""" """`.

La sintassi di controlli di flusso è esattamente la stessa del linguaggio C, ma le condizioni sono diverse a causa del tipo booleano.

Una grande differenza tra i due linguaggi consiste nella diversa diagnostica a tempo di compilazione e di esecuzione degli errori, poiché in Java i gli errori ed i messaggi di errore forniscono informazioni molto utili per la sua risoluzione. Gli errori a tempo di compilazione sono altrettanto efficaci in Java quanto in C, ma la differenza principale consiste negli errori a tempo di esecuzione tra i due linguaggi, poiché in C, in molte piattaforme, viene fornito un errore estremamente generico, in base alla versione di C, del compilatore, e da altre condizioni.

1.2 Paradigma Orientato agli Oggetti

Il paradigma procedurale, usato nel linguaggio C, separa in maniera netta tra il codice e le operazioni, ovvero la memoria e ed il processore, seguendo l'architettura definita da von Neumann. Anche se rappresenta il modo più naturale per scrivere programmi, nell'industria è stato ampiamente superato a favore del paradigma orientato agli oggetti. Questo paradigma segue la filosofia secondo cui le operazioni e lo stato sono connesse tra di loro, accomunate utilizzando "oggetti". Si è quindi rimossa la divisione imposta sui vecchi linguaggi di programmazione, che seguono l'architettura reale di un calcolatore. Per cui un problema è più facilmente modellabile se viene diviso in una pluralità di oggetti e classi di oggetti, contenenti lo stato e le operazione eseguibili su quell'oggetto. Questi oggetti si scambiano informazioni tra di loro, invocando particolari comandi; in questo modo gli oggetti conoscono ed interagiscono con altri esemplari, anche dello stesso tipo, o classe. Per conoscersi, questi oggetti contengono riferimenti agli altri oggetti.

Questo modo di interpretare un problema è vicino al nostro modo di pensare, il che aiuta nella creazione di programmi, utilizzando linguaggi di programmazione che seguono questo paradigma. Utilizzando un linguaggio come C, progettato per il paradigma procedurale, è possibile programmare seguendo il paradigma orientato agli oggetti, ma non essendo stato creato con questo scopo, la realizzazione di programmi è notevolmente più complesso.

L'esecuzione di un programma è uno scambio di messaggi tra questi oggetti connessi tra di loro tramite riferimenti interni, creando una rete di oggetti, modificando lo stato degli oggetti coinvolti nelle operazioni. Si può rappresentare la rete di oggetti tramite un diagramma di oggetti, indicati come rettangoli contenenti diversi campi, ed eventuali riferimenti ad altri oggetti, rappresentati come archi orientati verso altri oggetti.

Ogni oggetto contiene lo stato, un comportamento, le operazioni che offre, ed un identità, per differenziare diversi esemplari della stessa classe di oggetti. Si è dimostrato nel tempo come questo paradigma permette di scrivere e mantenere il codice molto più efficiente rispetto ad altri paradigmi. Gli oggetti vengono definiti tramite classi, rappresentate come file diversi del programma, contenenti lo stato, e la definizione dei metodi. Ogni oggetto creato da una classe possiede un'identità unica, e presenta lo stesso stato e le operazioni della classe di appartenenza. Le classi sono quindi delle istruzioni di montaggio, o fabbriche che producono oggetti.

Per studiare questo paradigma, si utilizzeranno esempi di forme geometriche, come studio di caso più semplice e limitato, ed uno studio di caso più esteso, mirato al refactoring, chiamato "diadia".

In Java ogni file contenente classi deve essere salvato con lo stesso nome della classe contenente, con estensione ".java". Le operazioni di oggetti in Java si indicano come metodi, invece di funzioni come nel linguaggio C, poiché rappresentano operazioni nel contesto di una classe, mentre le funzioni possono operare al di fuori di una classe.

Si considera un esempio semplice di una classe chiamata `Punto`, che rappresenta un punto in un piano di coordinate cartesiane bidimensionali:

```
public class Punto{
    // stato dell'oggetto
    private int x;
    private int y;

    // metodi propri della classe
    public void setX(int posX){
        this.x = posX;
    }
    public void setY(int posY){
        this.y = posY;
    }
    public int getX(){
        return x;
    }
    public int getY(){
        return y;
    }
}
```

Le variabili della classe `Punto` `x` e `y`, vengono chiamate variabili d'istanza. La creazione di un oggetto di una classe si ottiene tramite l'operatore `new`, secondo la seguente sintassi:

```
// creazione ed assegnazione di un nuovo oggetto Punto tramite riferimento
Punto origine = new Punto();
```

Questa riga di codice invoca un costruttore tramite l'operatore `new`, restituisce un riferimento ad un oggetto del tipo `Punto` appena creato, e viene conservato con un'assegnazione ad una variabile locale, in questo caso chiamato `origine`.

Tramite questo riferimenti è possibile accedere ai metodi, ed eventualmente allo stato dell'oggetto, tramite la notazione puntata, che segue la sintassi `<riferimento-oggetto>.<metodo>(<parametri-attuali>);`:

```
origine.setX(0); // assegna lo stato alla variabile x
origine.setY(0); // assegna lo stato alla variabile y
```

Questa notazione è molto efficace nell'esprimere la vicinanza tra i dati e le operazioni, per cui viene ampiamente usata in tutti i linguaggi OO (Orientati agli Oggetti). Ogni nuovo oggetto creato viene salvato in memoria, ed è possibile creare nuovi oggetti fino all'esaurimento della memoria virtuale. Questa classe viene quindi contenuta in un file `Punto.java`.

Per compilare un codice bisogna rendere il compilatore in grado di poter leggere il codice