Università di Trieste

Laurea in ingegneria elettronica e informatica

Enrico Piccin - Corso di Fondamenti di informatica - Prof. Sylvio Barbon Junior

Anno Accademico 2021/2022 - 1 Marzo 2022

Indice

1 Marzo 2022

1 Introduzione a Java

Java è un linguaggio di programmazione rigido e fortemente tupizzato, ovvero esige la dichiarazione di tutte le vairabili, con il loro tipo e con una loro inizializzazione. La **struttura formale** del linguaggio, pertanto, è predomincante ed è ciò che caratterizza Java rispetto ad altri linguaggi di programmazione, come *Python* o *PHP*.
Java, inoltre, è un linguaggio che opera al di sopra del sistema operativo, eliminando l'interdipendenza con il substrato software che permette l'esecuzione dei programmi: in altre parole, cambiando il sistema operativo non cambia il **codice sorgente**, ma solamente la piattaforma sulla quale esso viene eseguito; si dice, in tale caso, che **Java è un linguaggio portabile**.
Tuttavia, se da un lato tale impostazione costituisce un vantaggio, dall'altro rappresenta anche un limite: a parità di programma da eseguire, un programma scritto in Java sarà più lento, nell'esecuzione, di un programma scritto in C, per esempio, il quale è fortemente dipendente dall'ambiente software di esecuzione e permette di controllarne la gestione della memoria e dell'indirizzamento al fine di ottenere un'ottimizzazione massima. Questo in quanto Java è un **linguaggio interpretato**, che necessita prima di una **pseucodifica** per essere seguito su una **macchina virtuale**, mentre C è un *linguaggio compilato direttamente* sfruttrando la struttura del sistema operativo nativo.

1.1 Editor

La realizzazione di un programma in Java parte dalla scrittura del *codice sorgente* mediante un **editor**, anche chiamate **IDE** (dall'inglese *Integrated Development Environment*), ossia un programma di utilità che consente di inserire e memorizzare un testo e di modificarlo successivamente per effettuare aggiunte o correzioni.

1.2 Compilazione

Il codice sorgente scritto in Java che, proprio per questo, presenta estensione **.java**, al fine di poter essere eseguito dalla macchina, deve essere **compilato** da un **compilatore** (che nel caso di Java è il **javac**, o **JDK**, il quale permette di trasformare delle istruzione elaborate con un linguaggio *user-oriented* (di **alto livello**) in un linguaggio *machine-oriented* (di **basso livello**), ovverosia più vicino alla macchina, tale per cui la macchina lo possa comprendere ed eseguire. Nel caso di Java, la "macchina" che andrà ad eseguire le istruzioni così tradotte è una **macchina virtuale**, mentre nel caso del linguaggio C, la compilazione serve ad ottimizzare il programma finale, grazie alla forte interdipendenza tra linguaggio di programmazione e hardware sottostante.
Il risultato della compilazione di un codice sorgente .java è il **codice compilato** (o **bytecode**), con estensione **.class**: esso non deve essere più compilato e può essere eseguito su ogni piattaforma hardware-software.

1.3 Lettura e interpretazione

Il codice compilato (o bytecode), viene letto dal **loader**, ossia dalla macchina virtuale **java JRE**, la quale effettua la verifica del codice compilato (**bytecode verifier**) e ne interpreta le istruzioni.

1.4 Ambienti

Il linguaggio Java, per quello che si è detto, è un **linguaggio portabile** (in quanto può essere eseguito in diversi ambienti hardware e software) e un **linguaggio ad oggetti puro**, in quanto si basa sul **paradigma di programmazione orientato agli oggetti** (OOL, Object Oriented Language). Due, inolre, sono delle tecnologie che permettono una maggiore gestione dei programmi:

1. **Garbage Collector**: tool di gestione della memoria, per motivi di sicurezza e di semplicità di programmazione, pulendo ed elimando locazioni di memoria inutilizzate;

2. **Multithreading**: tecnologia che consente la gestione dell'esecuzione contemporanea di più task all'interno della stessa applicazione. Il processore, tramite la tecnologia **time sharing** permette di gestire lo scheduling dei processi in modo tale da permettere un'esecuzione quanto più simultanea delle diverse task.

La tecnologia Java, infine, si avvale di tre importanti tool per la sua esecuzione:

1. **JDK** (Java Development Kit): un set completo di **API** (librerie di funzioni precostituite che consentono di eseguire operazioni automaticamente) e una serie di tool per lo sviluppo di applicazioni Java;
2. **JSE** (Java Standard Edition): pacchhetto base per lo sviluppo di normali applicazioni Java (che possono andare dagli **Applets** a **Desktop applications** con *Swing* o *Java FX* per l'interfaccia grafica). Esistono anche **JEE** (Java Enterprise Edition) per lo sviluppo di applicazioni server di grandi aziende e **JME** (Java Mobile Edition) per lo sviluppo di applicazioni mobile.
3. **JRE** (Java Runtime Environment), ambiente per l'esecuzione delle applicazioni Java tramite **JVM**, o Java Virtual Machine.

1.5 Linguaggi di programmazione

I linguaggi di programmazione si possono suddividere in due grandi categorie:

1. linguaggi di **basso livello** (**linguaggi macchina** e Assembly), che sono, quindi, linguaggi che il calcolatore è in grado di comprendere;
2. **linguaggi ad alto livello** o **linguaggi evoluti**, quindi più vicini al programmatore e che consentono l'uso di **nomi simbolici** che corrispondono a più set di istruzioni linugaggio macchina.

1.6 Programmi traduttori: compilatore e interprete

Di seguito si espone la definzione di **compilatore**:

COMPILATORE

Il **compilatore** è un programma che traduce un programma sorgente scritto in linguaggio ad alto livello in un programma oggetto in linguaggio macchina.
Esso, inoltre, svolge tre importanti funzioni di analisi:

1. **analisi lessicale**: consente di individuare le parole chiave e riservate del linguaggio di programmazione, ccosì come del vocabolario e permette di costruire la tabella dei simboli;
2. **analisi sintattica**: ossia il controllo della correttezza delle istruzioni al fine di verificare se essse sono scritte rispettando la sintassi del linguaggio Java, cioé le regole della grammatica;
3. **analisi semantica**: ovvero il controllo della validità del significato degli elementi in base al contesto (per esempio la presenza delle istruzioni dichiarative necessarie).

Pertanto il compilatore individua e segnala tutti gli **errori formali**: se vi sono errori, la traduzione non può avvenire. Se non ci sono errori, il compilatore passa alla fase di sintesi durante la quale viene generato il codice oggetto, di solito in formato rilocabile.
Sarà poi compito del **linker** trasformare il programma oggetto in programma eseguibile (con estensione .class), il quale verrà poi eseguito dalla macchina virtuale JVM grazie all'**interprete** che traduce le istruzioni del linguaggio ad alto livello in linguaggio macchina, una per una, al momento dell'esecuzione.

Durante il **runtime** vengono anche rilevati degli errori di esecuzione, dovuti non alla scorrettezza formale del codice sorgente, ma ad un uso scorretto del programma.
L'impiego di un compilatore o di un inteprete presenta una serie di vantaggi e di svantaggi:

Compilatore	**Interprete**
VANTAGGI	
maggior velocità di esecuzione	semplicità di messa a punto dei programmi poiché si lavora in ambiente interattivo
risparmio di memoria	maggior portabilità dei programmi
rilevazione di tutti gli errori formali	
consente la segretezza del programma sorgente, in quanto dopo essere stato compilato non è possibile risalire al codice di partenza	
non è necessario codice aggiuntivo	
SVANTAGGI	
tempi di creazione del programma più lunghi, per la ricerche di librerie e funzioni specifiche	maggiore occupazione di memoria
minore portabilità	l'esecuzione risulta più lenta
	non dà garanzia di correttezza sintattica
	non consente la segretezza del codice sorgente

Figura 1: Vantaggi e svantaggi del compilatore e dell'interprete

Osservazione: Si osservi che Java è **parzialmente compilato** e **interpretato**.

1.7 Soluzione di problemi con il calcolatore

In informatica, un **problema** è una qualsiasi situazione alla quale è necessario trovare una soluzione. Per la risoluzione di un problema è necessario analizzare diversi elementi

1. la situazione iniziale (quali sono i dati del problema): i dati che riguardano il problema sono tutte le informazioni disponibili all'inizio e quelle che si desiderano ottenere come soluzione del problema;
2. che cosa si desidera ottenere (risultati): le azioni che possono essere eseguite nel processo risolutivo sono le operazioni che il computer è in grado di eseguire; l'insieme delle istruzione costituisce l'**algoritmo**;
3. le risorse a disposizione, variabili a seconda dell'hardware sottostante.

1.8 Algoritmo e programma

Di seguito si espone la definizione di **algoritmo**:

ALGORITMO

Un algoritmo è un procedimento che permette di ottenere dei risultati (dati in uscita o di **output**) partendo da alcuni dati iniziali (dati di ingresso o di **input**).
L'algoritmo deve essere

1. **generale**: non deve risolvere un singolo caso particolare, ma tutta una serie di problemi dello stesso tipo;
2. **deterministico**: i risultati devono essere sempre gli stessi, partendo dagli stessi dati iniziali, cioè l'esito dell'esecuzione dell'agoritmo non deve dipendere da aleatorietà o effetti casuali;

Il **programma**, invece, si ottiene codificando l'algoritmo in un **linguaggio di programmazione**, cioè scrivendo le istruzioni dell'algoritmo secondo la sintassi del linguaggio scelto.