Appunti per l'esame 1Z0-811 Java Foundations Associate

Capitolo 1

Programmi e linguaggi di programmazione

Un programmatore istruisce un computer con delle istruzioni (che è il nostro codice), sono istruzioni in un linguaggio che il computer possa capire, questo è il motivo per cui un programmatore fa uso di un linguaggio di programmazione. Il computer di base è in grado di comprendere soltanto il "linguaggio macchina" (sequenze di 1 e 0 binarie), i linguaggi che usiamo oggi per programmare vengono definiti ad "alto livello" e sono ad esempio Java, Python, C, PHP, Javascript… Vengono definiti di alto livello, perché le istruzioni che si impartiscono al computer vengono espresse in linguaggio naturale anziché macchina. Linguaggio naturale inglese. Questo linguaggio naturale viene poi tradotto da una sorta di "translator" (l'interprete) in codice macchina perché il computer lo capisca. Chi traduce il linguaggio di programmazione in codice macchina può essere l'interprete o il "compilatore", il flusso è il seguente:

Programmatore → istruzioni ad alto livello (es. java code) → translator → linguaggio macchina (101010101) → "Hello World" sullo schermo del computer

Le Applicazioni

Sono programmi che permettono di usare il computer. Bisogna installarle sul computer per poterle usare. Le applicazioni che girano sul pc si chiamano "applicazioni desktop", poi ci sono delle classi di applicazioni cosiddette "mobile" che funzionano soltanto sugli smartphone, e poi ci sono anche applicazioni (ad es. Gmail) che vengono definite "applicazioni web" che si possono usare attraverso un browser sul pc. Le applicazioni si scrivono con i linguaggi di programmazione.

Api (application program interface)

Sono programmi che vengono esposti da un servizio attraverso una interfaccia che può permettere di usare il servizio ad un altro programma. Questa interfaccia viene per l'appunto definita api.

Capitolo 2

Che cosa è Java

Java è un linguaggio di programmazione e come tutti i linguaggi di programmazione ha le sue "keywords", sintassi e regole.

JDK→java development kit, come sappiamo il codice scritto in un linguaggio di programmazione deve essere tradotto per il computer, java per fare ciò si serve di istruzioni quali il compilatore, il debugger e il disassembler che aiutano a scrivere codice, tutti insieme questi strumenti sono chiamati il Java Development Kit.

JVM→ Java Virtual Machine, è possibile definirla un computer a sua volta. Se pensiamo che ogni dispositivo esegue un sistema operativo e che per ogni dispositivo se voglio eseguirci un programma devo adattarlo a quel sistema è proprio qui che entra in gioco la JVM. La JVM non è un computer fisico, ma è un software, che si esegue sopra un computer fisico. La JVM si prende cura di tradurre il nostro codice Java in un linguaggio macchina. La JVM è stata sviluppata per diverse architetture, però ce ne sono alcune che non sono supportate, soprattutto quelle che non hanno nulla in comune con i computer dekstop, ecco perché non possiamo eseguire applicazioni dekstop su dispositivo Ios o Android.

Bytecode → è il linguaggio macchina che capisce la JVM, se vogliamo eseguire programmi sulla JVM per prima cosa li dobbiamo "compilare" in istruzioni scritte in java bytecode, alla fine poi la JVM usa le funzionalità del sistema operativo per eseguire il bytecode.

Java Class Library→ libreria che comprende un sacco di funzionalità si chiama "Java Api's" tutto questo codice è organizzato in package.

JRE → Java Runtime Environment → si tratta della JVM e della collezione di classi di libreria precompilate, molti computer le hanno installate, ma se devo installarle le posso scaricare. Basata sul principio WORA (write once run everywhere).

N.B.

Le applicazioni java sono leggermente differenti a seconda del sistema operativo che abbiamo al di sotto, perché la JVM non ha dipendenze sul linguaggio java, si possono scrivere programmi in altri linguaggi e compilare in bytecode, ogni parte di java lavora in maniere indipendente questa è una delle ragioni di successo. La JVM è come un programma qualunque su computer e usa il sistema operativo sottostante.

Java Platform → è la combinazione tra JVM, JRE e Dev Tools (compilatore ecc..), come base di partenza c'è la Java Standard Edition (Java SE) che contiene la JVM, La Standard Class Library e i DevTools, tutto questo è sufficiente per creare applicazioni stand alone. Per la parte Enterprise abbiamo Java Enterprise Edition (J2EE) che contiene la JVM, stardard and enterprise library e i dev tools, c'è poi anche Java Micro Edition(JME) per il mobile e i dispositivi elettronici.

La piattaforma Java è standardizzata per tutti i device. Esempio: se un device supporta Java SE allora supporta tutti gli elementi di Java SE, quindi se sviluppi un programma Java SE funzionerà su tutti i dispositivi che lo supportano.

N.B. J2EE ad oggi è Jakarta EE

Versioni di Java→ Java 1.0 è del 1996, Java 8 risale al 2014.

Java Community Process → JCP (1998) → organizzazione indipendente di utilizzatori Java. Controlla che java non venga alterato.

Caratteristiche di Java

- Indipendenza dalla piattaforma
- Performance Elevate
- Sicurezza
- Familiarità
- Semplicità
- Molteplici modalità di deploy
- Ecosistema

- Backward compatiblity
- Compilato → compilato in bytecode per mezzo del compilatore, il bytecode viene poi tradotto dalla JVM e questo fa si che Java sia un linguaggio compilato. I linguaggi compilati come Java hanno performance migliori a runtime di quelli interpretati come JavaScript o PHP.
- Varietà di soluzioni (jsp, jpa, ejb)
- Multithreading
- Distribuito
- Garbage collection
- Orientato agli oggetti puro
- Strutturato
- Fortemente tipizzato
- Gestione automatica della memoria
- Meccanismo delle eccezioni per il controllo degli errori

Capitolo 3

Le basi di Java

Installazione di java

Su microsoft Windows 10 di default java si va ad installare nel percorso C:\prograsm files\java. Se Java (jdk e/o Jre) non vengono configurati in maniera adeguata (variabili di ambiente) sul mio computer ricevo un messaggio di errore, quando cerco di richiamare il compilatore o la jre (tramite i comandi "javac" e "java"). Bisogna settare i PATH correttamente.

Note in merito ai cambiamenti di Oracle →la Oracle fino a jdk 10 ha ammesso il libero download della JDK, da JDK 11 è stato vietato. Per questo motivo è nato il gruppo Open JDK, quindi la Oracle possiede 2 build della JDK ufficiali:

- Java commerciale Oracle
- Open JDK

Path environment → dice al sistema operativo dove è localizzato l'eseguibile di Java. Aggiungere Java a questo percorso ci permette di arrivare automaticamente alla locazione degli eseguibili java. Possiamo verificare la presenza di java nel path digitando dal command:

c:\PATH

Usare un IDE → usare un ide è molto più che usare un editor di testo per scrivere codice java. I principali ide per java sono:

- Eclipse
- Netbeans
- IntelliJ

Poi ci sono ide più leggeri:

- Dr. Java
- BlueJ

E poi i classici editor di testo:

- Notepad++

JRE→include il minimo necessario per eseguire i programmi java e comprende la JVM, le classi della standard library (java api) e i loro files di configurazione ed estensione.

JDK→include tutti i tools richiesti per scrivere programmi java più il compilatore e un debugger. Inotlre contiene pure la jre infatti è molto più grosso della jre, se ad esempio abbiamo jdk 8 installato sul computer abbiamo anche la jre.

Componenti di un programma java → 1) codice sorgente 2) java standard library 3) librerie di terze parti.

Struttura di una classe Java→ un programma contiene di fatto le seguenti 3 cose:

- 0 oppure almeno 1 statement "package"
- 0 oppure almeno 1 statement "import"
- 0 oppure almeno 1 definizione di "class"

Esempio:

```
package com.abc;

import com.xyz;

class MyClass {……}
```

Il metodo MAIN → le classi java non sono file eseguibili nel vero senso della parola. Non sono dei file .exe. La JVM è un eseguibile, si esegue sempre la JVM. Noi passiamo come argomento il nome della classe che vogliamo che venga eseguita come argomento della JVM. Quando la JVM parte carica uno specifico metodo della classe e se non lo trova genera un errore, il metodo che viene cercato è il "metodo main()" ed ha una firma molto ben precisa.

- Il nome deve essere main()
- Come parametro riceve un array di stringhe
- Main è di tipo void
- Main deve essere dichiarato public e anche static

Esempio:

- public static void main(String[]args) {….}
- public static void main(String… args) {…}
- public static void main(String [] args) throwx Exception {…}

main può lanciare qualsiasi eccezione.

Esempi di dichiarazioni non valide del metodo main:

- static void main(…) // main non è dichiarato public
- public void main(…) // main non è dichiarato static
- public static void main(String[] a, String b) // main ha un solo parametro ed è un array di stringhe!!!

Compilare ed eseguire un programma Java → l'entry point di un programma java è sempre il suo metodo main(). Inoltre se il metodo main non si richiama ad altri metodi o ad altri oggetti è di fatto il programma java. Ma siccome java è un linguaggio orientato agli oggetti e ad alto livello allora non possiamo avere 1 metodo senza almeno 1 classe.

Il metodo è la "definizione di un comportamento". Un comportamento può essere incluso in una classe. Per creare un programma java di base allora dobbiamo per forza di cose crearci sempre una classe!

Step per compilare ed eseguire un programma:

- Individuare la cartella del codice sorgente
- Creare un file .java
- Compilare il file .java → javac MioFile.java
- Avviare il programma → java MioFile

N.B.

- Non possiamo avere overloading di metodi main()

Command line arguments → java TestClass a b c → args[0] = a, args[1] = b ecc…

Non si può passare il nome della classe java come argomento, perché proprio come per gli eseguibili non possiamo cambiare il nome della classe java. Il metodo main conosce già il nome della classe che lo contiene.

The end of main→ il metodo main è l'entry point delle nostre applicazioni java. Cosa succede quando il metodo main finisce? Dipende da cosa fa il metodo main() → se un programma è semplice terminato il main di solito termina e il tutto viene chiuso, ma siccome i programmi spesso sono composti da tantissime classi java, può succedere che il programma non termini. Si possono verificare molte attività in parallelo. La fine di un metodo main implica la fine di 1 sola attività, non vuol dire la fine di tutte le attività!! Java è in grado di eseguire attività in parallelo ad esempio servendosi dei threads.

Programmazione ad oggetti → Un pezzo di codice va pensato come un componente fisico. La gente deve sapere che cosa contiene ciò che compera!

API → application program interface → specifica una interfaccia per il proprio componente, l'interfaccia è una sorta di specifica con cui poter interagire con un certo componente. In poche parole è come il manuale delle istruzioni. Come sviluppatori di applicazioni noi dobbiamo conoscere l'interfaccia dei componenti che andremo a programmare. In java esiste la libreria e la sua relativa documentazione (javadoc). Quando installiamo la JRE essa include molte classi e pacchetti che si servono della Standard Java Class Library chiamata Java API -> questa contiene tutto ciò che serve di funzioni/metodi per costruire applicazioni, ciò fa risparmiare tempo nello sviluppo e non è necessario per noi conoscere in profondità il codice cui ci serviamo.

Principi della programmazione ad oggetti → la programmazione ad oggetti è un paradigma della programmazione che si ispira agli oggetti della vita reale. Se ci guardiamo intorno ogni cosa che vediamo è un oggetto, allo stesso modo quando sviluppiamo un programma in object oriented programming per noi ogni cosa è un oggetto, in java gli oggetti si modellano usando le "classi", l'interazione con gli oggetti "cambia lo stato" di questi che è rappresentato dai "campi" dell'oggetto (le variabili di istanza). Il processo che serve ad identificare gli argomenti da sviluppare come oggetti si definisce ASTRAZIONE.

Astrazione → è il processo che cattura i dettagli rilevanti di una entità o di un concetto. In Java una entità viene descritta nella classe. Quando un concetto è definito→ possiamo definire molte istanze di quel concetto (sono le istanze della classe).

Incapsulamento → è la tecnica che ci permette di nascondere l'implementazione e i dettagli di una entità. Una entità deve esporre il "cosa" e non il "come". In java si usano particolari "modificatori di accesso" per implementare l'incapsulamento.

Ereditarietà → è il processo che ci permette la creazione di entità specializzate estendendo entità esistenti, è una relazione "is a…" tra 2 entità. L'ereditarietà permette il riuso del codice estendendo quello che abbiamo già.

Polimorfismo → il polimorfismo permette che la stessa entità possa esibire comportamenti differenti sulla base dei contesti. Il polimorfismo è la diretta conseguenza dell'ereditarietà.

Java è un linguaggio ad oggetti, ma nonostante sia ad oggetti in alcuni casi viene usato in forma procedurale, però non è una buona idea perché proprio per le sue caratteristiche è più difficile usarlo come linguaggio procedurale.

Descrivere gli oggetti → descriviamo gli oggetti attraverso gli attributi e la proprietà/comportamento viene definito metodo.

Creare gli oggetti → si usa la parola chiave NEW, esempio: Car c = new Car();

Lanciare un programma java → inizia tutto con il suo metodo main() e lanciamo il programma attraverso la JSVM che esegue il metodo main. Il metodo main è per questa ragione definito l'entry point di una applicazione java.

Relazioni tra classi, oggetti e referenza → un oggetto è una istanza di una classe, la classe è quindi un template nel quale gli oggetti vengono creati. Una classe definisce gli oggetti, quando si creano gli oggetti usiamo la parola chiave "new". Per accedere ad un oggetto bisogna sapere la sua locazione di memoria, dopo che conosciamo la locazione di memoria possiamo accedere ai campi dell'oggetto. L'indirizzo è custodito in una variabile di referenza. Ci sono delle differenze fondamentali su come java tratta le "variabili referenza" e quelle che non lo sono cioè le "variabili primitive".

- Nel caso di una varibile INT (primitiva) la JVM usa il valore contenuto nella variabile così come è.
- Per una stringa la JVM sa come è definita come un riferimento in memoria e usa il valore contenuto per arrivare alla sua locazione di memoria
- In java non c'è modo di manipolare l'esatto valore contenuto in una variabile di referenza, lo posso fare in C/C++ ma non in java, perché i creatori di Java hanno deciso di non consentire l'accesso diretto alla memoria per motivi di sicurezza, ecco perché Java viene definito un linguaggio più sicuro del C/C++.
- Per assegnare riferimento basta copiare il valore in un altro, esempio: String str1 = str2; // str1 e str2 si riferiscono allo stesso oggetto.
- Una variabile referenza che di fatto non punta a nessun oggetto di fatto è = NULL. Di contro una variabile primitiva non si riferisce mai ad un oggetto

Esempi:

String str = null; // OK

Int n = 0; // OK

String str = 0; // non compila!!

Int n = null; // non compila!!

Statico e istanza → abbiamo detto che usiamo una classe per astrarre le entità del mondo reale e ci istanziamo gli oggetti nelle classi per creare i nostri programmi, però abbiamo detto che in java si può usare anche la programmazione procedurale, quindi per raggiungere uno scopo possiamo usare più modalità, teniamo conto comunque che in java a differenza del C/C++ ogni pezzo di codice deve per forza di cose rientrare in una classe sempre. Possiamo aggirare questo ostacolo usando la parola chiave "static". In inglese statico è qualcosa che non cambia mai, però in java per questo scopo hanno inventato la parola chiave "final", in Java statico è più qualcosa che dice che "appartiene alla classe", invece che ad una istanza di una classe. Possiamo usare qualcosa di statico senza creare oggetti della classe. In una classe java posso quindi avere:

- Metodi statici
- Variabili statiche (dette variabili di classe)

"Static" non è una caratteristica della programmazione ad oggetti, perché i campi statici non appartengono agli oggetti bensì all'intera classe. Un metodo statico appartiene all'intera classe e vi si può accedere ad esso anteponendo il nome della classe stessa. L'opposto di statico è l'istanza stessa. Quando avviamo la JVM non crea l'oggetto, ma invoca il metodo main() che viene passato come argomento a linea di comando.