Lezione 2

Strumenti di sviluppo

Prima di inoltrarsi nello studio di Java, bisogna procurarsi tutti gli strumenti di sviluppo necessari. In questa lezione sono illustrate le soluzioni software di Sun Microsystems, che hanno il pregio di essere ufficiali, gratuite, frequentemente aggiornate e dettagliatamente documentate. In questo corso si farà riferimento alla versione 1.4 della piattaforma J2SE, poiché attualmente è quella maggiormente supportata dalle terze parti. Le istruzioni qui presentate potranno essere adattate ai nuovi rilasci senza troppe modifiche, durante i tempi a venire. Per installare ed utilizzare gli strumenti di sviluppo qui descritti, è necessaria una minima padronanza del sistema operativo impiegato e dei comandi in esso contemplati.

2.1 - Macchina virtuale, pacchetti base e strumenti di sviluppo

Java, dalla sua prima release ufficiale, si è evoluto rapidamente. Attualmente sono disponibili diverse piattaforme di esecuzione e sviluppo siglate dal marchio "Java 2". In questa parte del corso si fa riferimento alla piattaforma *Java 2 Standard Edition* (*J2SE*). Sun Microsystems emette periodicamente tutti gli strumenti software necessari per lo sviluppo e l'esecuzione del software Java su diversi sistemi operativi. La piattaforma J2SE si divide in tre parti:

1. **Macchina virtuale**. E' il componente necessario per l'esecuzione del bytecode. Senza di esso, il software Java non può essere eseguito.
2. **Pacchetti base**. Classi, interfacce e pacchetti sono argomenti che verranno esaminati nel dettaglio in questo corso. Per il momento, si pensi ai pacchetti base come a delle raccolte di funzionalità software di cui ogni programma Java solitamente necessita. All'interno dei pacchetti base della piattaforma J2SE, ad esempio, sono contenuti gli strumenti utili per la gestione delle finestre, per i calcoli matematici, per la lettura e la scrittura dei file, per il networking, per l'impiego delle date e degli orari, per la manipolazione delle immagini e così via. La macchina virtuale raccoglie queste funzionalità e le fornisce, dietro richiesta, al software in esecuzione. Per farla breve, la Java Virtual Machine ed i pacchetti base lavorano in concerto. Poiché tali raccolte di software vengono distribuite insieme alla macchina virtuale, come fossero una sua stessa parte, non è necessario replicarle ed includerle in ogni programma creato. In questa maniera, il software Java risulta sempre leggero (in termini di spazio), dinamico e mai superfluo o ridondante. Altri pacchetti di natura meno generica posso essere installati a parte, in qualità di estensioni. Non appena ci si addentrerà nell'ottica di Java questi argomenti appariranno più chiari.
3. **Strumenti di sviluppo**. Costituiscono l'insieme dei software rivolti ai programmatori, indispensabili per realizzare qualsiasi software Java. Il compilatore è certamente lo strumento di sviluppo più utilizzato, ma di certo non l'unico della collezione. Ad esso si affiancano altri software, utili per la verifica, la documentazione e la distribuzione del bytecode prodotto.

Le più recenti implementazioni della piattaforma J2SE possono essere gratuitamente prelevate dal sito Web di Sun Microsystems. La pagina di riferimento si trova all'indirizzo:

http://java.sun.com/j2se/

Si controlli la più recente release disponibile e si acceda alla pagina di download associata.

2.2 - JRE e SDK

La casa madre di Java fornisce software per Windows, Linux e Solaris (uno UNIX commerciale prodotto e distribuito dalla stessa Sun). Altri sistemi operativi sono supportati da terze parti, ed in questo caso bisogna far riferimento al sito Web dello specifico produttore. Le implementazioni per MacOS, ad esempio, vengono mantenute direttamente da Apple. Altri sistemi sono supportati dalle comunità Open Source.

Ad ogni versione di Java, sul sito di Sun, associati due distinti download, etichettati rispettivamente "JRE" e "SDK". Bisogna evidenziare la differenza che corre tra ogni JRE (*Java Runtime Environment*) ed il corrispondente SDK (*Software Development Kit*).

Le installazioni JRE contengono il solo runtime del linguaggio, costituito dalla macchina virtuale e dai pacchetti base della piattaforma. Queste distribuzioni sono rivolte agli utenti e non ai programmatori, giacché non includono gli strumenti di sviluppo. In poche parole, il runtime permette l'esecuzione del bytecode, ma con esso non si può sviluppare o modificare un software Java. Per avere il compilatore e tutti gli altri software di sviluppo, è necessario installare una distribuzione SDK. Questi pacchetti, che pesano spesso più di 50 MB, comprendono al loro interno il corrispondente JRE. Installando un SDK, quindi, si avrà immediatamente a propria disposizione tutto quello che serve tanto per sviluppare quanto per eseguire il software Java.

MacOS e Java

Informazioni sugli strumenti Java integrati in MacOS possono essere reperite agli indirizzi:

http://www.apple.com/java/

http://developer.apple.com/java/

JDK e J2SE SDK

Il kit degli strumenti base per lo sviluppo di software Java attualmente distribuito da Sun Microsystems è chiamato J2SE SDK. In passato, Sun adoperò il nome JDK (*Java Development Kit*) per questo stesso genere di strumentazione. Talvolta, tanto nei newsgroup quanto in alcune documentazioni, si utilizza ancora la vecchia nomenclatura. E' dunque importante sapere che le sigle "J2SE SDK v1.4.2" e "JDK 1.4.2" identificano, di fatto, lo stesso kit.

2.3 - Installazione e configurazione del kit di sviluppo per Windows

L'installazione del kit di sviluppo per sistemi Windows è davvero semplice: avviato il programma di installazione, è sufficiente seguire le istruzioni presentate dalla procedura guidata. Completata l'installazione, è comodo annotare il percorso degli eseguibili del kit nella variabile di sistema chiamata *PATH*. La macchina virtuale, il compilatore e tutti gli altri strumenti di sviluppo, in questo modo, potranno essere richiamati senza dover digitare interamente il percorso che conduce alla cartella nella quale sono riposti. Si supponga, ad esempio, che il kit di sviluppo sia stato installato nella cartella *C:\j2sdk1.4.2_07*. Il compilatore, costituito dal programma *javac.exe*, si troverà al percorso *C:\j2sdk1.4.2_07\bin\javac.exe*. Se i sorgenti di un programma sono riposti in una locazione differente da quella in cui è conservato il compilatore, dal prompt dei comandi bisognerà digitare un tortuoso testo per tradurli in bytecode:

```
C:\j2sdk1.4.2_07\bin\javac.exe NomeSorgente.java
```

Annotando il percorso *C:\j2sdk1.4.2_07\bin* nella variabile di sistema *PATH*, invece, il medesimo risultato potrà essere ottenuto con il comando:

```
javac NomeSorgente.java
```

Per modificare il *PATH* del sistema è necessario osservare i seguenti passi:

1. Si individui l'esatto percorso della cartella che contiene il compilatore e gli altri strumenti del kit installato. Di norma, tale percorso ha una forma del tipo *C:\j2sdkX.Y.Z\bin*, dove *X*, *Y* e *Z* denotano la versione della distribuzione installata. Questa forma, naturalmente, varia in base a quanto specificato durante la procedura di installazione e in dipendenza dalla lettera associata al disco rigido impiegato per l'operazione.
2. Dal "Pannello di controllo" del sistema operativo si acceda alla voce "Sistema". Nella scheda "Avanzate" si ricerchi e si attivi un pulsante recante la dicitura "Variabili d'ambiente". Nell'elenco che riporta le variabili di sistema (da non confondere con la lista delle variabili utente), si ricerchi la voce *PATH* e se ne richieda la modifica. Nel caso non sia possibile individuarla, sarà necessario crearla sfruttando l'apposita opzione. Nel campo "Valore" associato alla variabile si introduca il percorso prima dedotto, separandolo con un punto e virgola dai dati eventualmente già presenti (**Figura 2.1**). Si confermi quindi ogni operazione effettuata, salvando le impostazioni digitate. Per portare gli intenti a compimento, è necessario utilizzare un profilo utente che abbia dei diritti di amministrazione.

Figura 2.2
Modifica della variabile di ambiente *PATH* su un sistema Windows XP. Se nel valore associato alla variabile sono già memorizzati dei dati, è necessario introdurre un punto e virgola prima del percorso da memorizzare. In caso contrario, il punto e virgola non serve.

2.4 - Il primo programma

Completata l'installazione del kit di sviluppo adatto alla propria piattaforma, non resta che sperimentare il buon esito dell'operazione mediante un esempio pratico. Con un editor testuale qualsiasi (sotto Windows si può utilizzare il "Blocco note") si crei un file chiamato *CiaoMondo.java*, da posizionare in una cartella qualsiasi, contenente il seguente codice:

```
public class CiaoMondo {

	public static void main(String[] args) {

		System.out.println("Ciao, Mondo!");

	}

}
```

Si acceda al prompt dei comandi del proprio sistema operativo, ci si posizioni all'interno della cartella utilizzata e si richiami il compilatore alla seguente maniera:

```
javac CiaoMondo.java
```

In caso di errore, si controlli l'installazione dell'SDK, la configurazione del *PATH* di sistema e la corretta digitazione del codice Java, secondo il messaggio riscontrato.
Se tutto è andato a buon fine, il compilatore ha prodotto un nuovo file, nominato

CiaoMondo.class. I file CLASS sono eseguibili Java, costituiti da puro bytecode. E' dunque possibile eseguirli su un qualsiasi sistema dotato di un runtime Java. Con la macchina virtuale di Sun Microsystems, il comando necessario è il seguente:

java CiaoMondo

2.5 - Dal compilatore alla macchina virtuale

L'esempio esaminato nel paragrafo precedente mette in chiaro l'impiego di due strumenti essenziali: il compilatore, indispensabile per trasformare un sorgente in bytecode, e l'esecutore, che invia alla macchina virtuale il codice già compilato. Allo stesso tempo, l'esempio fornisce anche una panoramica sulla filosofia che permea lo sviluppo e la distribuzione del software Java.
I sorgenti di un programma non sono altro che una raccolta di file testuali cui viene data l'estensione *.java*. Per convenzione, ciascun file sorgente contiene una classe (un costrutto del tipo *class NomeClasse { ... }*), prendendo da questa il proprio nome. L'esempio esaminato è semplicissimo, pertanto l'intero programma è riposto nella sola classe *CiaoMondo*, contenuta nel file *CiaoMondo.java*.
Con Java, un problema articolato può essere scomposto in tanti problemi di entità minore, da risolvere uno ad uno mediante una breve classe dedicata. La programmazione orientata agli oggetti prende piede proprio da questo assioma. Un programma di natura complessa, quindi, sarà certamente costituito da più classi cooperanti, distribuite in diversi sorgenti. Si pensi, ad esempio, ad un software per gestire la contabilità di un piccolo ufficio: una classe potrebbe dedicarsi esclusivamente al recupero dei dati conservati su disco, un'altra al salvataggio delle modifiche e delle aggiunte effettuate dall'utente, un'altra ancora si potrebbe occupare esclusivamente dell'interfaccia grafica (finestre, pulsanti, menù e così via). I progetti di vaste dimensioni traggono grossi benefici da questo approccio. Ogni singola classe, inoltre, potrebbe trovare impiego in applicazioni distinte, incoraggiando la riusabilità del codice.
I pacchetti base della piattaforma J2SE contengono migliaia di classi pronte all'uso, che coprono le più generiche necessità di ogni software. In *CiaoMondo*, ad esempio, si è fatto uso di *System*, una delle classi fornite insieme alla macchina virtuale. Tra le tante funzionalità offerte, si è selezionata quella utile per inviare del testo al canale di output predefinito (tipicamente la console dei comandi).
La programmazione orientata agli oggetti permette lo sviluppo di librerie di classi personali, da sommare a quelle già disponibili nella piattaforma. Qualsiasi software Java, pertanto, si presenta come una collezione di classi cooperanti, alcune tratte dai pacchetti base della macchina virtuale, altre realizzate e distribuite dagli sviluppatori del programma stesso.
Il compilatore, come rimarcato più volte, traduce un sorgente Java in bytecode. Per attivarlo, bisogna posizionarsi con il prompt dei comandi nella cartella che contiene i sorgenti, per poi digitare:

```
javac NomeClasse.java
```

Quando un programma è costituito da più classi, non è necessario richiedere esplicitamente la traduzione di ognuna di esse: il compilatore è in grado di rilevare automaticamente le dipendenze, cambiando in bytecode tutti i sorgenti indispensabili per il funzionamento della classe di cui si è comandata la compilazione.
Il prodotto della compilazione è costituito da dei file con estensione *.class*, uno per ogni classe tradotta. Il contenuto dei file compilati è puro bytecode. Ogni bytecode è:

- **Riusabile**. Una medesima classe può essere impiegata in progetti distinti.
- **Dinamico**. Per utilizzare una classe all'interno di un software non è necessario disporre del suo sorgente. E' possibile impiegare direttamente il bytecode. *CiaoMondo* lo dimostra: la classe *System*, fornita insieme alla macchina virtuale, è già compilata, ed il suo sorgente non è necessario per tradurre, distribuire ed eseguire un software Java che ne faccia uso. Per quanto una classe possa essere parte integrante di un software, inoltre, ogni bytecode conserva un certo grado di indipendenza: è possibile aggiornare o correggere un programma sostituendo solo alcuni tra i file che lo costituiscono, senza per questo doverlo ricompilare (e ridistribuire) interamente. Diversamente avviene con altri linguaggi di programmazione, dove l'impiego di codice già compilato e la separazione di un programma in blocchi dinamici richiedono complessi artifici.
- **Portabile**. Ogni sistema dotato di una macchina virtuale può eseguire i file con estensione *.class*. Il software funzionerà in maniera identica ovunque lo si esegua (approfondimento in **Figura 2.2**).
- **Sicuro**. Le operazioni effettuate da un software Java, come già si è fatto presente nella prima lezione del corso, devono sempre essere approvate dalla macchina virtuale, in base al modello di sicurezza impiegato. Un bytecode di dubbia origine, pertanto, gode di meno privilegi rispetto ad un software ritenuto di fiducia dall'utente o dal sistema. Attraverso un'oculata gestione dei programmi eseguiti, è possibile dormire sonni tranquilli: nessuno potrà mai violare la sicurezza e la stabilità della macchina gestita.

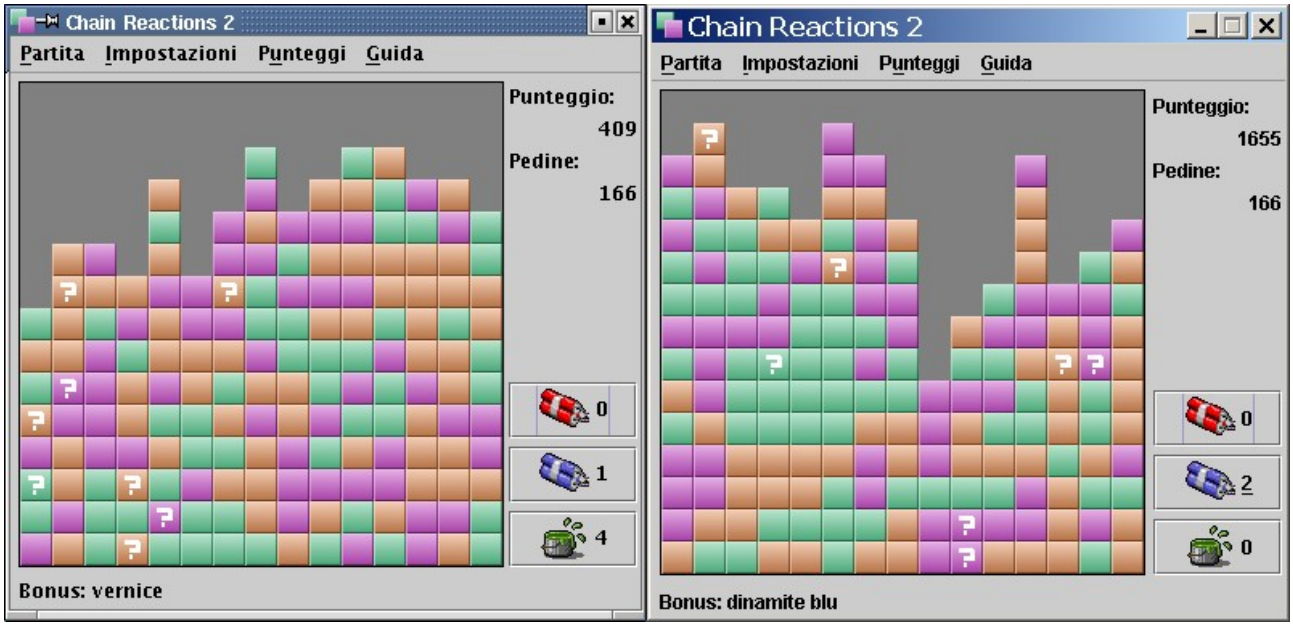

Figura 2.2
In figura è mostrato un programma Java eseguito su Linux (a sinistra) e su Windows (a destra). Per ottenere il risultato è stato sufficiente trasferire da un sistema all'altro i file che compongono il software, senza nessun ulteriore intervento. Il programma (un semplice giochino) si comporta in maniera identica sui due sistemi operativi impiegati per il test. Si riscontrano delle differenze minime nell'interfaccia, che si adatta automaticamente al motore grafico del sistema in uso. D'altro canto, non potrebbe avvenire diversamente.

Un programma, come si è appena detto, può essere costituito da più classi cooperanti. Affinché l'esecutore possa avviare il software, ad ogni modo, è necessario predisporre un punto di avvio all'interno di una delle classi che lo compongono. In *CiaoMondo*, ad esempio, è stato definito un blocco del tipo:

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

Questo rende *CiaoMondo* una classe avviabile. L'esecuzione del programma comincia proprio dal contenuto del blocco sopra riportato, etichettato "*main*". Le classi avviabili possono essere mandate in esecuzione digitando:

```
java NomeClasse
```

L'estensione *.class* non deve essere specificata.

I tanti argomenti snocciolati nel corso di questo paragrafo troveranno il giusto grado di approfondimento nelle lezioni successive. Per il momento, si prenda confidenza con le regole più basilari del linguaggio e della macchina virtuale, imparando come impiegare in maniera corretta il compilatore e l'esecutore.

2.6 - Altri strumenti di sviluppo

Diversi altri strumenti accompagnano il kit di sviluppo di Sun Microsystems. Tre di questi vanno subito citati, anche se non vi è necessità di impiegarli immediatamente:

- ***appletviewer***. E' uno strumento che consente la visualizzazione ed il test delle applet. Le applet sono programmi Java che possono essere eseguiti all'interno di un Web browser, come elementi dinamici di una pagina HTML.
- ***jar***. Gli archivi JAR sono utilizzati per racchiudere un programma (o una libreria di classi) all'interno di un unico file, compresso con la stessa tecnica adoperata per gli archivi ZIP. Questo facilita la distribuzione del software Java, altrimenti spezzato in una miriade di file distinti (un *.class* per ogni classe impiegata, più le immagini, i suoni, i file di configurazione e tutte le altre risorse necessarie). La dinamicità non viene lesa da questo formato, che si dimostra duttile e malleabile. Gli archivi JAR possono essere resi avviabili. In molti sistemi operativi (come Windows e MacOS), il programma contenuto in un archivio JAR avviabile può essere lanciato senza passare per la riga di comando, ricorrendo alla particolare combinazione impiegata dall'interfaccia grafica in uso (sotto la configurazione predefinita di Windows, ad esempio, basta un doppio clic del mouse sull'icona del file).
- ***javadoc***. Quando si sviluppa una libreria di classi che dovrà essere impiegata da altri programmatori, è necessario realizzare una documentazione che illustri dettagliatamente l'uso delle funzionalità offerte. Senza di essa, diventa davvero difficile capire come una certa classe possa essere impiegata o estesa. Lo strumento *javadoc* aiuta lo sviluppatore nel generare documentazione in formato HTML per le proprie classi. Gli stessi pacchetti base, prodotti da Sun Microsystems, sono stati documentati con questo strumento (si veda il paragrafo successivo).

2.7 - La documentazione dei pacchetti base

La documentazione ufficiale dei pacchetti base di Java è tra le più preziose risorse di cui può disporre uno sviluppatore. Qualsiasi corso, per quanto vasto ed approfondito, non può trattare nel dettaglio le migliaia di funzionalità disponibili, tra l'altro in continua espansione. Un punto di riferimento come la documentazione ufficiale, pertanto, diventa essenziale per lo sviluppo di software Java. Questo corso si pone l'obiettivo di far comprendere al lettore il pensiero ed i meccanismi che sono alla base del linguaggio e della sua piattaforma. Compresi questi, sarà possibile usufruire delle documentazioni in maniera veloce ed

efficace.

2.8 - Linea di comando, editor facilitati ed ambienti integrati

Lavorare con un comune editor di testo, compilando i programmi da riga di comando, può presto divenire inefficiente e noioso. Benché questa soluzione sia ideale durante la prima fase di ogni percorso didattico, presto si sentirà l'esigenza di impiegare un ambiente che velocizzi le distinte fasi dello sviluppo del software. Molti programmi, commerciali e non, vengono incontro alle esigenze dei programmatori Java. Bisogna distinguere due categorie di prodotti: gli editor facilitati e gli ambienti integrati.

Alla categoria degli editor facilitati appartengono quei programmi che aiutano il programmatore mediante l'uso dei colori nel testo digitato, l'indentazione automatica del codice e le scorciatoie che permettono la compilazione e l'esecuzione con pochi clic del mouse. Generalmente, i programmi di questo tipo possono gestire dei progetti di non grande entità. Gli editor facilitati sono spesso freeware o shareware. Una ricerca in rete metterà in luce le scelte disponibili per ciascun sistema operativo.