Lezione 12

Pacchetti

In questa lezione sono trattati i package, che costituiscono una delle caratteristiche più innovative e peculiari di Java. I pacchetti, in sostanza, sono contenitori impiegati per raggruppare le classi e le interfacce, in maniera ordinata. Grazie ad essi, ciascuna classe trova posto in un comparto idoneo e coerente, dove ogni potenziale conflitto di nomi è preventivamente risolto. La lezione illustra i pacchetti, descrivendone prima la creazione e quindi l'utilizzo.

12.1 - Utilità dei pacchetti

Al mondo esistono milioni e milioni di programmatori Java, ognuno dei quali realizza i propri software suddividendoli in classi. Non è possibile stimare con accuratezza quante classi siano state create, in tutto il mondo, dal debutto di Java ad oggi. Bisogna poi ricordare che tutto il software Java è strutturato in maniera modulare: normalmente le classi possono essere condivise tra più applicazioni e tra più programmatori. La stessa piattaforma Java 2 comprende al suo interno decine di migliaia classi, realizzate per essere sfruttate in numerose occasioni distinte. Sun Microsystems, inoltre, non è l'unica azienda che produce e distribuisce classi Java: il mondo è pieno di programmatori e di altre aziende che fanno altrettanto. Raccolte di classi, idonee ai più disparati usi, possono essere liberamente vendute, acquistate o rese gratuitamente disponibili via Internet. Insomma, le classi di uso comune sono davvero tante. Una raccolta tanto vasta, se sprovvista di un meccanismo capace di tenerla sotto controllo, rischia il collasso. In particolare, la disponibilità di svariate migliaia di classi porta direttamente a due problemi: il disordine e la possibilità di omonimia tra due strutture distinte. Chi è stato il primo ad inventare una classe chiamata *Rettangolo*? Chi dovrebbe detenere il diritto di nominare una classe in questo modo, negando a tutti gli altri sviluppatori del mondo la possibilità di creare strutture omonime per non fare confusione?
Java risolve entrambi i problemi mediante i pacchetti. Un pacchetto, in breve, è una raccolta di classi (e di interfacce). Ogni pacchetto ha un nome, composto generalmente da più parti, che di norma identifica il produttore della classe e l'ambito di validità della medesima. I problemi di omonimia e disordine, con un colpo solo, sono stati risolti.
Le classi incluse nella piattaforma J2SE sono suddivise in pacchetti. I pacchetti base inclusi nella versione 1.4 della piattaforma J2SE sono ufficialmente centotrentacinque. Al loro interno trovano posto le oltre diecimila classi citate in precedenza, ordinatamente riposte e separate. Ad esempio, i pacchetti base di Java 2 ospitano due classi chiamate *List*. La prima è contenuta nel pacchetto *java.awt*, mentre la seconda è all'interno di *java.util*. Grazie ai pacchetti, rimane sempre possibile fare riferimento all'una o all'altra senza creare equivoci. Il nome completo della prima, infatti, è *java.awt.List*, mentre quello della seconda è *java.util.List*.

12.2 - Creazione di un pacchetto

Creare un pacchetto è davvero molto semplice. Inserendo l'istruzione *package* in cima ad un sorgente, si fa in modo che la classe in esso contenuta venga introdotta all'interno del pacchetto specificato:

```
package it.sauronsoftware.miopacchetto;

public class Persona {

  private String nome;
```

```
  private String cognome;

  public Persona(String nome, String cognome) {
    this.nome = nome;
    this.cognome = cognome;
  }

  public String getNome() {
    return nome;
  }

  public String getCognome() {
    return cognome;
  }

}
```

Tuttavia, la sola istruzione package non è sufficiente affinché la classe possa essere correttamente utilizzata. Java esige che le classi siano memorizzate all'interno di directory nominate esattamente come i pacchetti che le contengono. Dunque, il file *Persona.class* realizzato poco sopra dovrà essere inserito in un percorso di cartelle del tipo *it/sauronsoftware/miopacchetto*. A questo punto, ponendosi immediatamente fuori dalla directory impiegata per realizzare il pacchetto, diviene possibile utilizzare *Persona* al seguente modo:

```
class Test {

  public static void main(String[] args) {
    it.sauronsoftware.miopacchetto.Persona p =
    new it.sauronsoftware.miopacchetto.Persona("Mario", "Rossi");
    System.out.println(p.getNome());
    System.out.println(p.getCognome());
  }

}
```

Il nome completo della classe creata, dunque, è *it.sauronsoftware.miopacchetto.Persona*.

Affinché l'esempio funzioni, è necessario organizzare i sorgenti nella seguente struttura di cartelle:

```
Cartella di base dell'applicazione
  |
  +- /it
  |    |
  |    +- /sauronsoftware
  |         |
  |         +- /miopacchetto
  |              |
  |              +- Persona.java
  |
  +- Test.java
```

Posizionandosi nella cartella di base dell'applicazione, è sufficiente compilare *Test.java* in maniera canonica:

```
javac Test.java
```

Il compilatore risolverà automaticamente le dipendenze, creando e disponendo automaticamente i file *.class* necessari all'esecuzione del programma. Al termine dell'operazione, quindi, avremo la seguente struttura:

```
Cartella di base dell'applicazione
  |
  +- /it
  |    |
  |    +- /sauronsoftware
  |         |
  |         +- /miopacchetto
  |              |
  |              +- Persona.java
  |              +- Persona.class
  |
  +- Test.java
  +- Test.class
```

Sempre restando nella directory di base dell'applicazione, il programma può essere eseguito in maniera classica:

```
java Test
```

Come è possibile osservare, esiste una corrispondenza fissa tra i pacchetti di Java e la maniera di distribuire sorgenti ed eseguibili nel file system del computer.

Una classe priva dell'istruzione *package* è automaticamente assegnata al pacchetto di default, che non ha nome. Il percorso su disco del pacchetto di default è costituito da tutte le locazioni espresse nella variabile d'ambiente *CLASSPATH*. Il pacchetto di default può essere liberamente utilizzato per lo sviluppo di software di entità piccola o medio-piccola. Quando si passa ad applicazioni più corpose, ad ogni modo, l'impiego dei pacchetti è certamente raccomandato e raccomandabile. Più un software è "spezzato" in pacchetti indipendenti e logicamente validi, più sarà facile mantenerlo, correggerlo, aggiornarlo ed evolverlo.

12.3 - Gerarchie di pacchetti

I pacchetti possono essere organizzati in maniera gerarchica, ossia possono essere nominalmente inseriti l'uno dentro l'altro. Il punto, anche in questo caso, lavora come carattere divisore:

```
package pacchetto1.pacchetto2.pacchetto3 ... ;
```

L'esempio dimostrato nel paragrafo precedente già faceva uso di questa caratteristica. La struttura di una gerarchia di pacchetti deve riflettersi sulla distribuzione delle classi lungo il file system. Ad esempio, le classi appartenenti al pacchetto

```
it.mariorossi.matematica
```

devono essere inserite nella directory al percorso

```
it\mariorossi\matematica (Windows)

it/mariorossi/matematica (Linux)
```

12.4 - Convenzioni per i nomi dei pacchetti

Gli identificatori validi per i pacchetti devono essere costituiti esclusivamente da lettere, rese in minuscolo. Sun Microsystems, inoltre, ha stabilito delle linee guida per aiutare lo sviluppatore nello scegliere i nomi ed i percorsi da assegnare ai propri pacchetti. Di norma, si suggerisce la creazione di pacchetti composti almeno da tre parti gerarchiche:

1. La prima parte deve essere un suffisso scelto tra i domini di primo livello validi in Internet (*com*, *net*, *org*, *it*, *uk*, *au* e così via). Questo permette di localizzare la posizione geografica o commerciale del produttore del pacchetto.
2. La seconda parte deve identificare l'autore o il distributore del pacchetto. Ad esempio, può essere usato il nome della software house che distribuirà il pacchetto, oppure il nome del suo autore.
3. Dalla terza parte in poi, si è liberi di scegliere gli identificatori a proprio piacimento, rispecchiando comunque l'organizzazione interna del proprio lavoro e mettendo in evidenza, nella maniera più semplice possibile, lo scopo e le funzionalità delle classi contenute nel pacchetto.

Ad esempio, si supponga che Luigi Bianchi abbia realizzato una collezione di classi Java cooperanti che, insieme, offrono un'interfaccia di programmazione (*API*) valida per interagire in qualità di client con un server FTP. Tali classi dovrebbero essere racchiuse in un pacchetto del tipo:

```
it.luigibianchi.ftpclient
```

I pacchetti integrati all'interno della piattaforma J2SE sono gli unici cui è concessa la possibilità di seguire convenzioni differenti, semplicemente perché potrebbero esistere più implementatori di un medesimo pacchetto. Si ricorda che le specifiche di Java e della sua piattaforma sono aperte, e chiunque può realizzare una propria implementazione della macchina virtuale e delle sue librerie. Se ogni produttore marchiasse con il proprio nome i pacchetti base della piattaforma, si perderebbe l'omogeneità necessaria affinché un'applicazione Java possa essere eseguita su piattaforme diverse. Solitamente, quindi, i pacchetti base della piattaforma Java utilizzano i suffissi *java* e *javax*. Il primo identifica i pacchetti più basilari ed indispensabili, senza i quali poco avrebbe senso, mentre il secondo descrive le estensioni standard ufficialmente riconosciute da Sun Microsystems.

12.5 - Norme d'accesso e pacchetti

Come si è anticipato in precedenza (cfr. **Lezione 10**), i pacchetti hanno un certo influsso sull'accessibilità dei membri di una classe. I membri etichettati come *public* o *private* non variano il loro comportamento in dipendenza dei pacchetti: i primi restano visibili ovunque, mentre i secondi possono essere sfruttati esclusivamente dall'interno delle classi cui appartengono. I pacchetti dettano legge, invece, quando ad un membro non è associato alcuno specificatore di accesso, oppure quando lo stesso è etichettato come *protected*. Le possibilità di accesso ai membri di una classe è riassunto nel seguente specchietto, già incontrato in precedenza:

	public	***protected***	***niente***	***private***
Stessa classe	✔	✔	✔	✔
Sottoclasse stesso pacchetto	✔	✔	✔	✘

	public	*protected*	*niente*	*private*
Non sottoclasse stesso pacchetto	✔	✔	✔	✘
Sottoclasse altro pacchetto	✔	✔	✘	✘
Non sottoclasse altro pacchetto	✔	✘	✘	✘

Tabella 12.1
Specificatori d'accesso. Da dove si ha libero accesso ai membri di una classe?

I pacchetti, inoltre, determinano anche la visibilità delle classi. Una classe etichettata come *public* può essere sfruttata da qualsiasi posizione:

```
public class NomeClasse {

  // ...

}
```

Una classe priva di specificatore d'accesso, al contrario, può essere richiamata esclusivamente dall'interno del pacchetto al quale appartiene, rimanendo nascosta dal suo esterno.

12.6 - La parola chiave import

Supponiamo di voler realizzare un software che faccia frequente utilizzo della classe *MiaClasse*, conservata nel pacchetto *it.miopacchetto.mioargomento*. Ogni volta che si intende richiamare tale classe è necessario digitarne il nome in maniera estesa:

```
it.miopacchetto.mioargomento.MiaClasse
```

Questa pratica, ovviamente, è scomoda. Java, pertanto, fornisce la parola chiave *import*, che semplifica la stesura del codice:

```
import it.miopacchetto.mioargomento.MiaClasse;

public class Esempio {
  // ...
  public void mioMetodo() {
    //...
    MiaClasse mc = new MiaClasse();
    //...
  }
  // ...
}
```

Una volta che una classe è stata importata, è possibile far riferimento ad essa digitandone semplicemente il nome, escludendo il suo percorso completo. La parola chiave *import*, è necessario saperlo, compie esclusivamente questo lavoro, al solo scopo di facilitare la digitazione del codice. E' possibile scrivere software completi senza mai ricorrere alla clausola *import*, anche se non è conveniente farlo.

Per importare classi provenienti da più pacchetti, è sufficiente ricorrere ad una sequenza

di istruzioni *import*:

```
import it.miopacchetto.mioargomento.MiaClasse;
import java.io.InputStream;
import java.util.Vector;
...
```

Tutto il contenuto di un pacchetto può essere importato in un colpo solo:

```
import it.miomacchetto.mioargomento.*;
```

In questo modo, tutte le classi incluse nel pacchetto *it.miopacchetto.mioargomento* saranno automaticamente importate nel sorgente.

Il pacchetto più fondamentale della piattaforma Java 2 è *java.lang*. Al suo interno sono raccolte diverse classi basilari, come *String* e *System*. In tutti i codici sinora esaminati, ad ogni modo, ci si è sempre riferiti a queste due classi senza specificarne il percorso completo. Mai è stato scritto:

```
java.lang.String s = "Ciao";
java.lang.System.out.println(s);
```

Il pacchetto *java.lang*, infatti, è sempre importato automaticamente in ogni sorgente Java, senza che ci sia bisogno di digitare esplicitamente:

```
import java.lang.*; // Inutile, è implicito!
```