Programmazione Orientata agli Oggetti in Linguaggio Java

Test e Correzione: Conclusioni Parte a

versione 2.3

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Riassumendo
- La Piattaforma Java
  - ⇨Il Concetto di Classpath
  - ⇨La Classe it.unibas.utilita.Logger
  - ⇨Distribuzione delle Applicazioni
  - ⇨Lo Strumento jar

Riassumendo

- Alcune tecniche fondamentali
  - ⇨sviluppo di test di regressione
  - ⇨utilizzo di un sistema di logging
- Sviluppo di test di regressione
  - ⇨consente di automatizzare lo svolgimento dei test
  - ⇨è una misura essenziale per gestire l'evoluzione del codice
  - ⇨i test di regressione sono un "paracadute"

Riassumendo

- JUnit
  - ⇨framework di riferimento in Java per lo sviluppo di test di regressione
  - ⇨è necessario installarlo
  - ⇨è necessario conoscerne le regole per lo sviluppo dei test
  - ⇨classe di test, setUp, metodo di test, tearDown, asserzioni, testSuite

Riassumendo

- Asserzioni nei test e asserzioni difensive
  - ⇨in entrambi i casi hanno la funzione di rendere il codice capace di “autoverificarsi”
  - ⇨le asserzioni difensive nei metodi servono a garantire che siano soddisfatte le pre-condizioni prima dell’esecuzione dei metodi
  - ⇨le asserzioni nei test servono a verficare che siano soddisfatte le post-condizioni dopo l’esecuzione dei metodi

Riassumendo

- Utilizzo di un sistema di logging
  - ⇨consente di rendere più rapida la correzione del codice
  - ⇨consiste nell’aggiungere al codice istruzioni di registrazione di stringhe (“log records”)
  - ⇨a diversi livelli di priorità
  - ⇨su dispositivi potenzialmente diversi
  - ⇨le istruzioni possono essere abilitate o disabilitate facilmente

Riassumendo

- Attenzione
  - ⇨in entrambi i casi (test, logging) all'inizio viene richiesto un investimento
  - ⇨per scrivere codice aggiuntivo rispetto al codice che risolve le specifiche del problema
- Ma...
  - ⇨l'investimento si ripaga fortemente nel tempo
  - ⇨tanto che questo codice aggiuntivo deve essere considerato parte integrante dell'appl.

Riassumendo

- Logging e asserzioni
  - ⇨due strumenti collegati
  - ⇨le asserzioni (sia quelle difensive che quelle dei test) hanno la funzione di segnalare errori nel codice non appena si verificano >> rilevazione dell'errore
  - ⇨le istruzioni di logging viceversa hanno la funzione di aiutare a capire le cause dopo che si è manifestato un errore >> eliminazione dell'errore

La Piattaforma Java

- Tre aspetti significativi della piattaforma
  - ⇨utilizzo del classpath
  - ⇨proprietà di sistema e package java.util.logging
  - ⇨distribuzione delle applicazioni
  - ⇨utilizzo dello strumento jar

Il Concetto di Classpath

- In queste lezioni
  - ⇨utilizzo di un framework per lo sviluppo di test di regressione (JUnit)
  - ⇨utilizzo di un framework per il logging (java.util.logging o Log4J)
- In entrambi i casi
  - ⇨può essere necessario utilizzare package java sviluppati da terze parti

Il Concetto di Classpath

- In questi casi
  - ⇨è necessario risolvere problemi di visibilità delle classi
  - ⇨durante la compilazione
  - ⇨durante il collegamento dinamico
- Lo strumento di Java per la visibilità
  - ⇨il classpath
  - ⇨“percorso delle classi”

Il Concetto di Classpath

ATTENZIONE
alla regola di visibilità di Java

- Regola di visibilità in Java
  - ⇨è visibile esclusivamente quello che è raggiungibile attraverso il classpath
- Valore del classpath
  - ⇨il valore standard è . (cartella corrente)
  - ⇨può essere ridefinito attraverso la variabile di ambiente CLASSPATH

Il Concetto di Classpath

- Contenuto del classpath
  - ⇨lista di percorsi sul disco
  - ⇨corrispondenti alle cartelle di codice sorgente
  - ⇨e a file jar
- Nota
  - ⇨in ogni caso include %JRE_HOME%\lib\rt.jar
  - ⇨senza che sia necessario specificarlo esplicitamente

Il Concetto di Classpath

ATTENZIONE
alla ridefinizione del Classpath

- Una prima soluzione
  - ⇨cambiare il classpath di sistema
  - ⇨definire esplicitamente la variabile di ambiente CLASSPATH attribuendogli come valore .;%JUNIT_HOME%\junit.jar; dove %JUNIT_HOME% deve essere sostituito con il valore della cartella relativa (es: c:\Programmi\junit3.8.1)

Il Concetto di Classpath

- In questo modo
  - sto suggerendo al compilatore e alla macchina virtuale di cercare le classi nelle seguenti posizioni
  - %JAVA_HOME%\lib\rt.jar (sempre)
  - la cartella da cui vengono lanciati i comandi
  - il file jar %JUNIT_HOME%\junit.jar
  - posso aggiungere successivamente altre cartelle ed altri jar

Il Concetto di Classpath

- Un modo alternativo
  - per specificare il classpath
  - l'opzione –classpath di javac e di java
- Esempio
  - javac –classpath .;c:\junit\junit.jar prova*.java
  - java –classpath .;c:\junit\junit.jar prova.Principale

Il Concetto di Classpath

- Semantica
  - ⇨specifica un valore del classpath valido esclusivamente per una compilazione o una esecuzione specifica della macchina virtuale
  - ⇨prevale sul valore della variabile di ambiente CLASSPATH
  - ⇨utile nel caso in cui non sia opportuno modificare permanentemente il valore della variabile di ambiente

La Classe it.unibas.utilita.Logger

- Il codice della classe Logger
  - ⇨inizializza un logger chiamato it.unibas.utilita
  - ⇨stabilisce il livello per il logger
  - ⇨associa un handler al logger
- Due caratteristiche interessanti
  - ⇨accesso alle proprietà di sistema
  - ⇨blocco di inizializzazione statico

La Classe it.unibas.utilita.Logger

- Accesso alle proprietà di sistema
  - ⇨attraverso il metodo System.getProperty(String nomeProprietà)
  - ⇨consente di accedere ad una serie di proprietà predefinite
  - ⇨oppure proprietà specificate dall'utente sulla linea di comando attraverso l'opzione –D
  - ⇨es: java -Dit.unibas.utilita.LogLevel=finer <classe>

Esempi di Proprietà Predefinite

Key	Description of Associated Value
java.version	Java Runtime Environment version
java.vendor	Java Runtime Environment vendor
java.home	Java installation directory
java.vm.version	Java Virtual Machine implementation version
java.vm.vendor	Java Virtual Machine implementation vendor
java.class.path	Java class path
java.io.tmpdir	Default temp file path
java.compiler	Name of JIT compiler to use
os.name	Operating system name
os.version	Operating system version
file.separator	File separator ("/" on UNIX)
path.separator	Path separator (":" on UNIX)
line.separator	Line separator ("\n" on UNIX)
user.name	User's account name
user.home	User's home directory
user.dir	User's current working directory

La Classe it.unibas.utilita.Logger

- File .properties
  - ⇨file di testo contenente una serie di coppie nome=valore
- Caricamento di file .properties
  - ⇨può essere utilizzata la classe java.util.Properties
  - ⇨metodo void load(InputStream stream)
  - ⇨metodo String getProperty(String nome)

La Classe it.unibas.utilita.Logger

>> it.unibas.utilita.Logger

- L'altra particolarità
  - ⇨blocco di inizializzazione statico
- Sintassi
  - ⇨blocco di istruzioni racchiuse tra static{ ... }
- Semantica
  - ⇨viene eseguito all'atto del caricamento di una classe
  - ⇨consente di inizializzare lo stato del componente statico

Distribuzione delle Applicazioni

- Le applicazioni di questo modulo
  - ⇨sono composte di una serie di package
  - ⇨utilizzano le classi del package utilita.jar
  - ⇨che deve essere visibile nel classpath
- Come distribuire l'applicazione ?
  - ⇨in modo che sia facilmente eseguibile anche su una macchina non configurata

Distribuzione delle Applicazioni

- Una soluzione standard per Java
  - ⇨distribuire l'applicazione sotto forma di file .zip contenente tutti i componenti
  - ⇨il codice confezionato sotto forma di file .jar
  - ⇨le librerie necessarie sotto forma di file .jar disposte in una cartella apposita (es: lib)
  - ⇨uno "script" per l'esecuzione (es: file batch per Windows oppure file .sh per Linux)

>> indovinaIlNumero.zip
>> mediaPesata.zip

Distribuzione delle Applicazioni

- Procedura per confezionare il file .zip
  - ⇨creare un file .jar contenente i package dell'applicazione
  - ⇨risolvere il problema del classpath, in modo che eseguendo il codice del jar siano visibili le librerie
  - ⇨eseguire l'applicazione dal jar
- Soluzione
  - ⇨utilizzare lo strumento jar.exe

Lo Strumento jar

- jar.exe
  - ⇨strumento per la gestione di file .jar
  - ⇨eseguibile dalla linea di comando
- Caratteristiche di un file .jar
  - ⇨si tratta di un file compresso (simile a .zip)
  - ⇨la posizione delle classi deve rispettare le regole relative ai package di Java
  - ⇨il jar deve contenere un file di metainformazione detto "manifesto"

Lo Strumento jar

- Manifesto di un file .jar
  - ⇨file MANIFEST.MF contenuto in una cartella interna del jar META-INF
  - ⇨contiene una serie di coppie nome=valore che specificano informazioni descrittive sul jar
- Un manifesto minimale

```
Manifest-Version: 1.0
Created-By: 1.4.2 (Sun Microsystems Inc.)
```


Lo Strumento jar

- Per decomprimere un jar esistente
  - ⇨jar xvf <nomeFile>.jar
  - ⇨es: jar xvf utilita.jar
  - ⇨decomprime i file contenuti nel jar nella cartella corrente
  - ⇨riproducendo la struttura interna di cartelle contenuta all'interno del file .jar
  - ⇨crea la cartella META-INF con il relativo file di manifesto MANIFEST.MF

Lo Strumento jar

- Per creare un file jar
  - ⇨jar cvf <nomeFile>.jar <fileDaIncludere>
  - ⇨crea automaticamente un file di manifesto minimale
- File da includere
  - ⇨nome di file, anche con caratteri speciali (es: *.class)
  - ⇨oppure nome di cartella

Lo Strumento jar

- Esempio: mediaPesata
  - ⇨posizionandosi nella cartella padre della cartella it
  - ⇨jar cvf mediaPesata.jar it
- Per utilizzare un file di manifesto esistente
  - ⇨jar cvmf <fileManifesto>.mf <jar> <file>
  - ⇨es: jar cvmf metadati.mf mediaPesata.jar it
  - ⇨attenzione all'ordine in cui compaiono m ed f

Lo Strumento jar

- A che serve un manifesto esistente ?
  - ⇨consente di specificare informazioni aggiuntive per la macchina virtuale
  - ⇨nel caso che il jar sia “eseguito”
- Esecuzione di un jar
  - ⇨opzione –jar di java.exe
  - ⇨es: java –jar mediaPesta.jar

Lo Strumento jar

- Utilizzo del manifesto
  - ⇨la macchina virtuale consulta il manifesto per reperire due informazioni
- I informazione: classe eseguibile
  - ⇨nome della classe contenuta nel jar con il metodo main da cui avviare l’esecuzione
- II informazione: classpath da utilizzare
  - ⇨viene ignorato il classpath configurato sulla macchina, e considerato quello nel manifesto

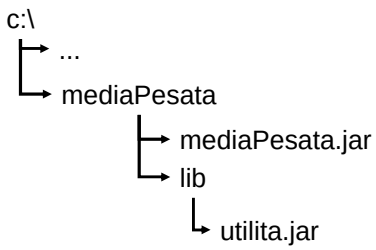
Lo Strumento jar

- Un esempio di manifesto completo
  - ⇨per mediapesata.jar

```
Manifest-Version: 1.0
Created-By: 1.4.2-b28 (Sun Microsystems Inc.)
Built-by: mecca
Implementation-Version: 1.0.51
Main-Class: it.unibas.mediapesata.Principale
Class-Path: lib/utilita.jar
```


Lo Strumento jar

- Attenzione al CLASSPATH
  - ⇨Class-Path: lib/utilita.jar
  - ⇨viene specificato in modo relativo rispetto alla posizione del jar
  - ⇨assume la presenza di una cartella lib allo stesso livello del file .jar



Lo Strumento jar

- Confezionamento dello zip
  - creazione della cartella radice (es: mediaPesata)
  - creazione del file .jar del codice con il manifesto preconfezionato nella radice (es: mediaPesata.jar)
  - creazione della sottocartella lib della radice contenente le librerie
  - creazione di uno script (es: mediaPesata.bat) eseguibile nella radice che contenga il comando java –jar <nomefile>.jar
  - compressione della cartella radice in un file .zip

Riassumendo

- Riassumendo
- La Piattaforma Java
  - Il Concetto di CLASSPATH
  - La Classe it.unibas.utilita.Logger
  - Distribuzione delle Applicazioni
  - Lo Strumento jar

Termini della Licenza