Programmazione Orientata agli Oggetti in Linguaggio Java

Strumenti di Sviluppo: Introduzione

versione 1.0

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

Sommario

- Terminologia
- Ambienti di Sviluppo Integrati (IDE)
- Sistema di Costruzione
- In Questo Corso

Terminologia

- Processo di Sviluppo del Codice
  - ⇨scrittura del codice sorgente
  - ⇨compilazione
  - ⇨esecuzione dei test
  - ⇨correzione
- Al termine dello sviluppo
  - ⇨utilizzo
  - ⇨due possibilità: distribuzione o “deployment”

Terminologia

- Distribuzione
  - ⇨confezionamento del codice in un formato distribuibile; es: .jar oppure .zip
  - ⇨gli utenti scaricano e utilizzano
- Messa in opera (“deployment”)
  - ⇨predisposizione per il funzionamento
  - ⇨esempio: copia del jar di una libreria in una cartella visibile attraverso il CLASSPATH

Terminologia

- Differenza tra installazione e deployment
  - ⇨installazione viene intesa spesso come predisposizione per l'utilizzo interattivo su una macchina
  - ⇨è una forma particolare di deployment
  - ⇨deployment ha un'accezione più ampia (es: deployment di una libreria)
- Ma, in pratica...
  - ⇨i due termini sono usati come sinonimi

Terminologia

- Una metafora per descrivere il processo
  - ⇨il programmatore scrive il codice sorgente
  - ⇨e poi questo codice subisce una serie di trasformazioni
  - ⇨necessarie per "costruire" l'applicazione
- Analogo: fabbrica automobilistica
  - ⇨il codice sorgente è la materia prima
  - ⇨l'applicazione è l'automobile finita

Terminologia

- Due categorie di operazioni
  - ⇨predisposizione della materia prima
  - ⇨operazioni di costruzione
- Operazioni di costruzione (“build”)
  - ⇨compilazione
  - ⇨esecuzione dei test
  - ⇨predisposizione per la distribuzione
  - ⇨messa in opera (“deployment”)

Terminologia

- L’operazione “build”
  - ⇨deriva dalla terminologia tradizionale (C/C++)
  - ⇨si riferiva tradizionalmente al processo di compilazione e collegamento
- Nei linguaggi moderni
  - ⇨sostanzialmente build = compilazione
  - ⇨oppure, in un’accezione più sofisticata, build = compilazione ed esecuzione dei test

Terminologia

- Due categorie di build
  - ⇨build di sviluppo (“debug”)
  - ⇨build di rilascio (“release”)
- Build di sviluppo (“debug build”)
  - ⇨operazione di costruzione effettuata durante la fase di sviluppo e correzione
  - ⇨il compilatore produce informazioni di debug
  - ⇨il codice non viene ottimizzato
  - ⇨vengono compilati ed eseguiti i test

Terminologia

- Build di rilascio (“release build”)
  - ⇨operazione di costruzione effettuata durante la fase di rilascio e messa in produzione
  - ⇨il codice oggetto viene ottimizzato
  - ⇨non vengono incluse informazioni di debug
  - ⇨dalla distribuzione sono esclusi i test

Terminologia

- Strumenti necessari per la scrittura
  - ⇨editor e documentazione
- Strumenti necessari per la costruzione
  - ⇨compilatore
  - ⇨strumento per i test
  - ⇨debugger e/o sistema di logging
  - ⇨strumenti per la distribuzione
  - ⇨strumenti per la messa in opera

Terminologia

- Fino a questo punto
  - ⇨il processo è stato gestito con strumenti “poveri” e non integrati tra di loro
  - ⇨editor di testi con funzionalità di base
  - ⇨strumenti per la linea di comando (javac, java, jar)
  - ⇨strumenti aggiuntivi (debugger, test runner)
  - ⇨strumenti del sistema operativo (zip, copia)

Terminologia

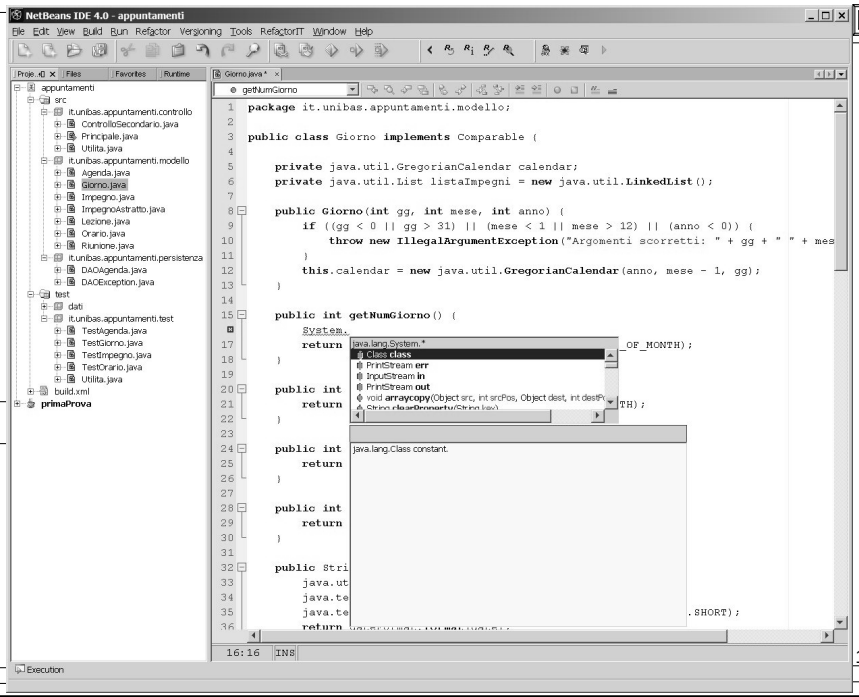
- In un contesto di sviluppo professionale
  - ⇨uno dei requisiti fondamentali per il programmatore è la produttività
  - ⇨è importante che gli strumenti siano accessibili rapidamente ed utilizzabili in modo efficace
- La soluzione tipica
  - ⇨utilizzare un ambiente di sviluppo integrato

Ambienti di Sviluppo Integrato

- Integrated Development Environment
  - ⇨in breve IDE
  - ⇨un'applicazione che raccoglie tutti gli strumenti necessari
  - ⇨consentendo di utilizzarli in maniera integrata
- Gli IDE più diffusi per Java
  - ⇨NetBeans
  - ⇨Eclipse

Ambienti di Sviluppo Integrato

- Funzionalità tipiche dell'IDE
  - ⇨supporto alla scrittura del codice
  - ⇨accesso integrato alla documentazione
  - ⇨supporto alle operazioni di costruzione
- Un esempio
  - ⇨NetBeans



Ambienti di Sviluppo Integrato

- In effetti
  - ⇨utilizzare un IDE è indispensabile per un programmatore professionista
  - ⇨l'IDE diventa la fabbrica per la costruzione
- In alcuni contesti
  - ⇨l'IDE viene considerato come coincidente con la piattaforma di sviluppo
  - ⇨es: Netbeans = Java
  - ⇨es: Visual Studio = .NET

Ambienti di Sviluppo Integrato

- In altri termini, secondo questa filosofia
  - ⇨imparare a programmare con la piattaforma vuol dire imparare ad usare l'IDE
- Grande svantaggio di questo approccio
  - ⇨il programmatore diventa dipendente dall'ambiente di sviluppo
  - ⇨non è autonomo nel caso in cui l'ambiente di sviluppo non sia disponibile

Ambienti di Sviluppo Integrato

- Limiti dell'IDE
  - ⇨non sempre l'IDE è disponibile sulla macchina di produzione; in questo caso il processo di costruzione non è replicabile
  - ⇨ogni programmatore ha il suo IDE e le sue impostazioni; in un gruppo di lavoro è difficile avere impostazioni comuni
  - ⇨l'IDE è uno strumento interattivo; non consente di costruire il codice automaticamente

Sistema di Costruzione

- Un approccio alternativo
  - ⇨utilizzare l'IDE per le attività di scrittura del codice sorgente
  - ⇨utilizzare un sistema di costruzione per le operazioni di costruzione
- Sistema di Costruzione
  - ⇨linguaggio per specificare operazioni di costruzione del codice
  - ⇨in modo che siano automatizzabili

Sistema di Costruzione

- Il sistema di costruzione standard di Java
  - ⇨Ant
- Ant
  - ⇨progetto open source della Apache Software Foundation – http://ant.apache.org
  - ⇨completamente scritto in Java
  - ⇨un linguaggio basato su XML per specificare operazioni di costruzione

>> ant utilita

Sistema di Costruzione

- Vantaggi di questo approccio
  - ⇨il processo di costruzione del codice è indipendente dall'IDE, dalla macchina, dallo sviluppatore e anche dalla piattaforma
  - ⇨il programmatore ha completo controllo su tutti i dettagli del processo di costruzione
  - ⇨il processo di costruzione è completamente automatizzato e replicabile rapidamente e frequentemente

In Questo Corso

ATTENZIONE
alla filosofia rispetto agli strumenti

- Utilizzeremo questo secondo approccio
  - ⇨l'IDE verrà utilizzato esclusivamente per le operazioni di scrittura del codice sorgente
  - ⇨Ant verrà utilizzato per tutte le operazioni di costruzione del codice
- La filosofia di sviluppo
  - ⇨"integrazione continua"

In Questo Corso

- Integrazione continua
  - ⇨automatizzazione del processo di costruzione
  - ⇨ruolo centrale dei test di regressione
  - ⇨"refactoring" continuo, ovvero miglioramento continuo della qualità del codice
- I progetti di riferimento
  - ⇨tutti i progetti visti fino a questo punto sono stati sviluppati con questo approccio

Riassumendo

- Terminologia
- Ambienti di Sviluppo Integrati (IDE)
- Sistema di Costruzione
- In Questo Corso

Termini della Licenza