Programmazione Orientata agli Oggetti in Linguaggio Java

Classi e Oggetti: Conclusioni Parte b

versione 2.1

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

Sommario

- Strumenti
- Compilatore
- Macchina Virtuale
- Il Concetto di Classpath

Strumenti

- Java 2 Software Development Kit
  - ⇨collezione di strumenti per lo sviluppo Java
- In particolare
  - ⇨compilatore, javac
  - ⇨macchina virtuale, java
  - ⇨librerie (API di Java)
  - ⇨e molti altri

Compilatore

- Compilatore per Java
  - ⇨compila il codice sorgente in bytecode per la macchina virtuale
  - ⇨ne esistono vari
- Il compilatore della Sun
  - ⇨javac.exe
  - ⇨nella cartella bin di %JAVA_HOME%
  - ⇨utilizzo: javac <*percorso*>\<*classe*>.java

Compilatore

- Input per il compilatore
  - ⇨un file .java contenente il codice sorgente di una classe
- Output del compilatore
  - ⇨un file .class contenente il bytecode della classe relativa

Compilatore

- Cosa c'è nel file .class ?
  - ⇨codice oggetto nel linguaggio macchina della macchina virtuale Java
  - ⇨è possibile disassemblarlo per intuirne il funzionamento attraverso il disassemblatore standard di Java: javap

javap –c –l –verbose Circonferenza

Compilatore

- E il collegamento ?
  - ⇨nella programmazione procedurale, tipicamente il codice oggetto deve essere collegato una volta per tutte per generare l'eseguibile
  - ⇨in Java, viceversa, il collegamento è dinamico, e avviene a tempo di esecuzione
  - ⇨di conseguenza, prima dell'esecuzione è sufficiente solo la compilazione del codice

Compilatore

- Riferimenti ad altre classi
  - ⇨abbiamo detto che nel codice sorgente di una classe X è potenzialmente visibile qualsiasi altra classe Java Y
  - ⇨purché, però, durante la compilazione di X, il file del codice sorgente di Y sia disponibile al compilatore
  - ⇨per effettuare le verifiche di correttezza necessarie

Compilatore

ATTENZIONE
alle regole del compilatore

- Il processo di compilazione
  - un processo che si svolge secondo regole abbastanza complesse
  - che dipendono dalla relazione tra codice sorgente e posizione fisica dei file
- Esempio
  - il compilatore viene lanciato con il comando: javac circonferenzea\Principale.java
  - dalla cartella c:\codice (“cartella corrente”)

Compilatore

- Primo passo
  - il compilatore localizza il codice sorgente della classe da compilare utilizzando il percorso specificato sulla linea di comando a partire dalla cartella corrente
  - es: cerca un file Principale.java nella cartella circonferenzea di c:\codice
  - se trova il file, comincia a compilare il codice
  - altrimenti restituisce un errore

Compilatore

- La verifica dei riferimenti
  - supponiamo che durante la compilazione di un file .java venga trovato un riferimento ad un'altra classe
  - es: compilando Principale.java viene trovato un riferimento a Circonferenza oppure a java.lang.System
  - il compilatore comincia a cercare il bytecode della nuova classe per verificare il riferimento

Compilatore

- Le API del JRE
  - sono contenute nel file rt.jar ("runtime" jar)
  - nella cartella %JAVA_HOME%\jre\lib
  - sono sempre visibili sia per il compilatore, sia per la macchina virtuale che ne conoscono la posizione
- Il formato .jar
  - formato di java per la distribuzione di archivi complessi di classi

Compilatore

- File .jar
  - ⇨ archivio compresso contenente una collezione di classi Java che rispetta la struttura di cartelle dei package
  - ⇨ analogo di .zip/.tar per Java
  - ⇨ l'SDK fornisce uno strumento apposito per la manipolazione dei jar: jar.exe
  - ⇨ jar cvf <nomeFile>.jar <cartella>
    per comprimere il contenuto di una cartella
  - ⇨ jar xvf <nomeFile>.jar per decomprimere

>> rt.jar

Compilatore

- La risoluzione dei riferimenti
  - ⇨ il compilatore cerca il bytecode per verificare che la classe cercata esista e sia corretta
  - ⇨ è in grado di "ispezionare" il bytecode per verificare se le chiamate sono corrette
  - ⇨ se, per caso, invece del bytecode trova il codice sorgente (.java), lo compila per verificarne la correttezza
  - ⇨ quindi una compilazione può avviarne altre

Compilatore

- L'algoritmo di ricerca
  - ⇨come prima operazione, il compilatore completa il nome della nuova classe con il package corrispondente
  - ⇨tre possibili casi: (a) stesso package; (b) java.lang; (c) un package delle import
  - ⇨nell'esempio: Circonferenza viene completato come circonferenzea.Circonferenza, System viene completato come java.lang.System

Compilatore

- L'algoritmo di ricerca (continua)
  - ⇨a questo punto viene cercata la nuova classe
  - ⇨la ricerca viene effettuata in due posizioni
- Prima posizione
  - ⇨in tutte le sottocartelle della cartella corrente
- Seconda posizione
  - ⇨nell'archivio che contiene le classi della piattaforma (il file %JAVA_HOME%\jre\lib\rt.jar)

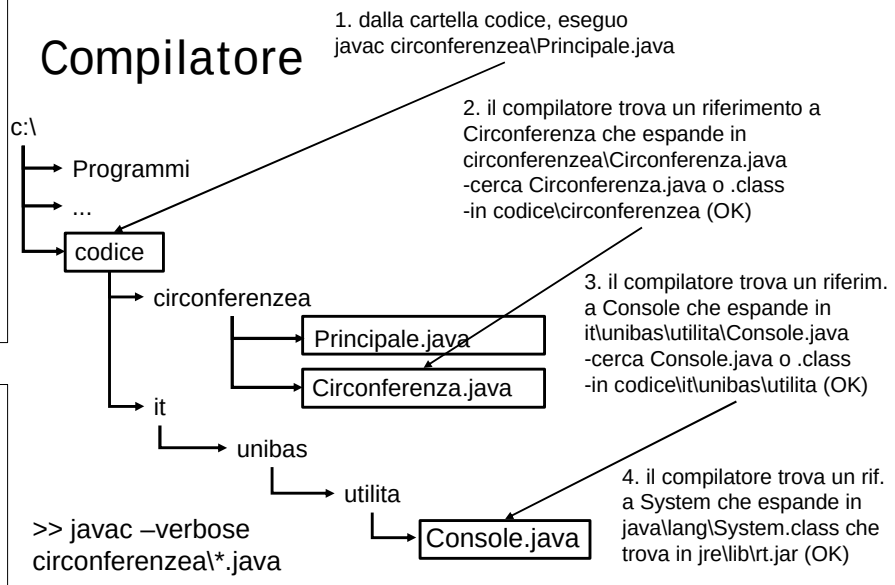
Compilatore

- Esempio
  - in Principale: riferimento a Circonferenza >> circonferenzea.Circonferenza >> cercata nella sottocartella circonferenzea della cartella corrente
  - in Principale: riferimento a it.unibas.utilita.Console >> cercata nella cartella it\unibas\utilita della cartella corrente
  - in Principale: riferimento a System >> java.lang.System >> cercata in rt.jar

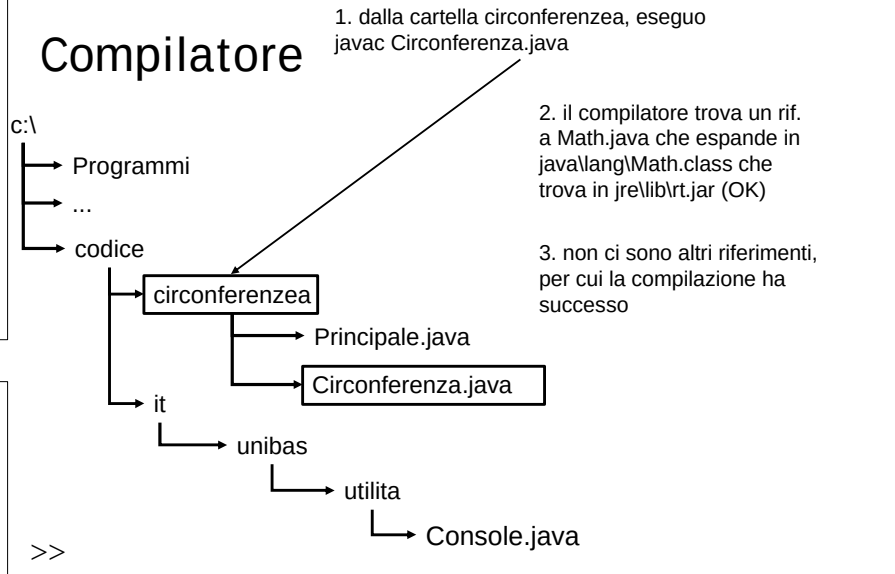
Compilatore

- Quindi
  - il funzionamento corretto della compilazione dipende da due fattori particolari
- Primo fattore
  - scelta corretta della cartella corrente da cui lanciare il compilatore
- Secondo fattore
  - corretta organizzazione delle classi all'interno delle sottocartelle e dei package

Compilatore



Compilatore

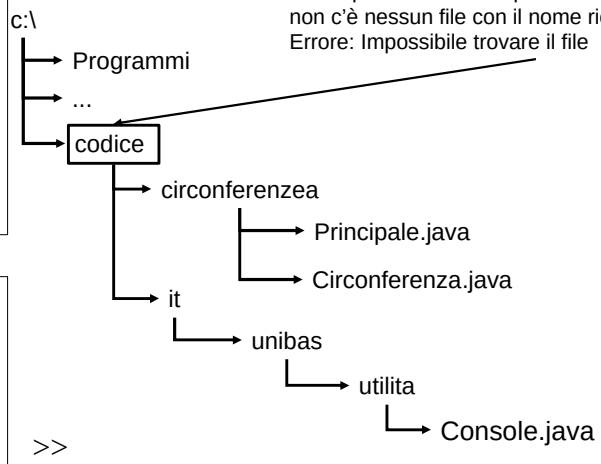


Compilatore

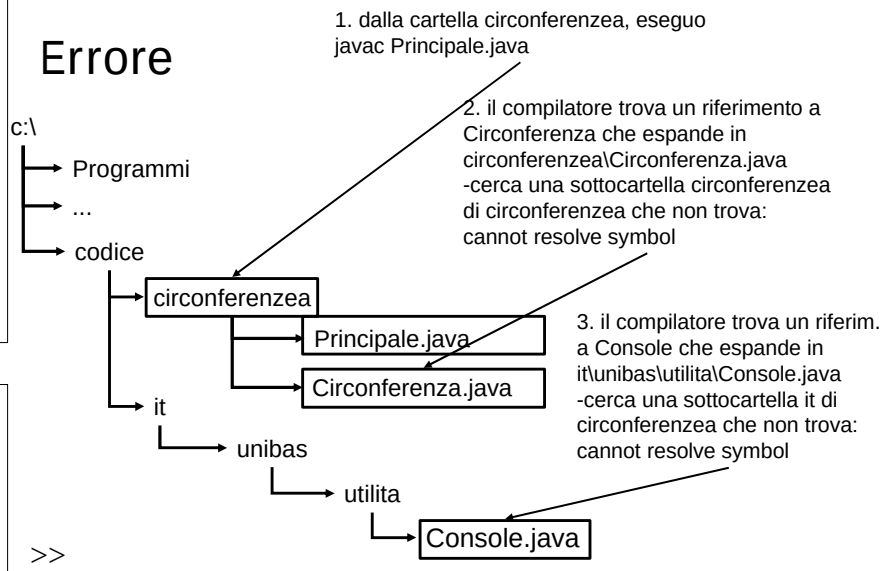
- Prima fonte di errori
  - ⇨impossibilità di localizzare il file di codice sorgente da compilare per via del percorso errato; errore del tipo “Impossibile trovare il file”
- Seconda fonte di errori
  - ⇨errori nel risolvere i riferimenti da una classe all’altra per via di errori nella posizione dei file; errori del tipo “cannot resolve symbol” durante la compilazione

Errore

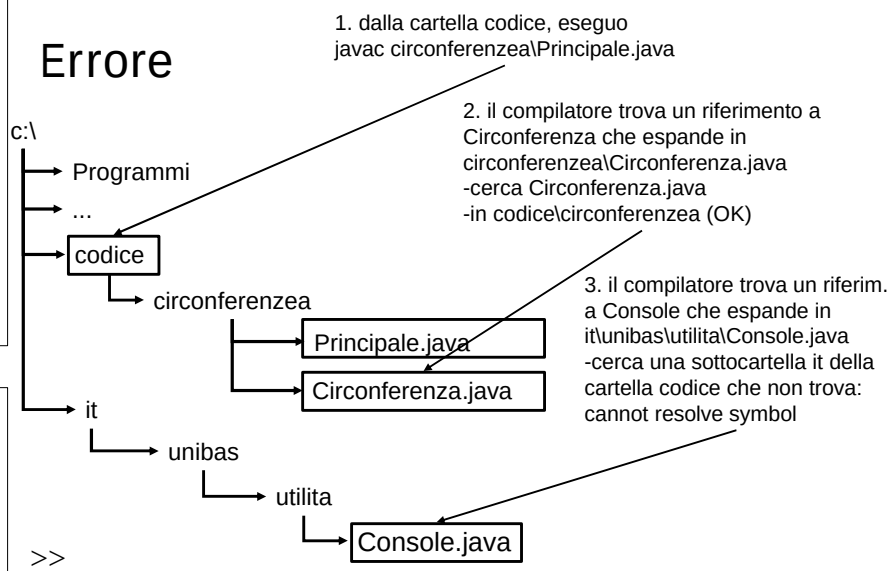
1. dalla cartella codice, eseguo
javac Principale.java (oppure javac .\Principale.java)
la compilazione fallisce perchè nella cartella
non c’è nessun file con il nome richiesto
Errore: Impossibile trovare il file



Errore



Errore



Compilatore

- Alcune annotazioni
  - ⇨è possibile compilare più di una classe contemporaneamente usando l'asterisco es: javac circonferenzea*.java
  - ⇨inoltre, questo algoritmo è una versione semplificata dell'algoritmo di ricerca di Java (lo approfondiremo successivamente)

Macchina Virtuale

- L'interprete della macchina virtuale
  - ⇨esegue il bytecode nella macchina virtuale
  - ⇨in realtà, tipicamente compila "al volo" il bytecode per il processore della macchina fisica
- Compilatore "Just In Time" (JIT)
  - ⇨compilatore che compila il codice in modo che sia eseguibile dalla macchina fisica a tempo di esecuzione

Macchina Virtuale

- Anche in questo caso
  - ⇨ne esistono vari
  - ⇨attenzione: qui le prestazioni sono determinanti
- L'interprete della Sun
  - ⇨java.exe
  - ⇨nella cartella %JAVA_HOME%\bin
  - ⇨uso: java <*nomeClasse*>

Macchina Virtuale

- Anche in questo caso ci sono regole
  - ⇨dovute al fatto che è necessario effettuare il collegamento dinamico del codice
  - ⇨molto simili a quelle della compilazione
- ATTENZIONE alla differenza
  - ⇨il compilatore è orientato ai file (gli argomenti sono nomi di file .java di codice sorgente)
  - ⇨la macchina virtuale è orientata alle classi; gli argomenti sono nomi di classi

Macchina Virtuale

- Esempi di istruzioni di compilazione
  - ⇨javac calcolatrice\Calcolatrice.java
  - ⇨javac it\unibas\utilita\Console.java
  - ⇨javac c:\codice\circonferenza*.java
- Esempi di istruzioni di esecuzione
  - ⇨java calcolatrice.Principale
  - ⇨java circonferenzea.Principale
  - ⇨java it.unibas.utilita.Console

Macchina Virtuale

ATTENZIONE
alle regole del collegamento dinamico

- Il processo di esecuzione
  - ⇨supponiamo che l'utente esegua la macchina virtuale su una classe; es:
    java circonferenzea.Principale da c:\codice
- Primo passo
  - ⇨caricamento del codice della classe da eseguire
  - ⇨il caricamento viene effettuato da un modulo opportuno chiamato ClassLoader

Macchina Virtuale

- ClassLoader
  - ⇨modulo della macchina virtuale che si incarica di localizzare e reperire il bytecode delle classi
- L'algoritmo del ClassLoader
  - ⇨simile a quello utilizzato dal compilatore
  - ⇨per localizzare la classe, il ClassLoader sfrutta la relazione tra nome della classe e struttura dei file su disco

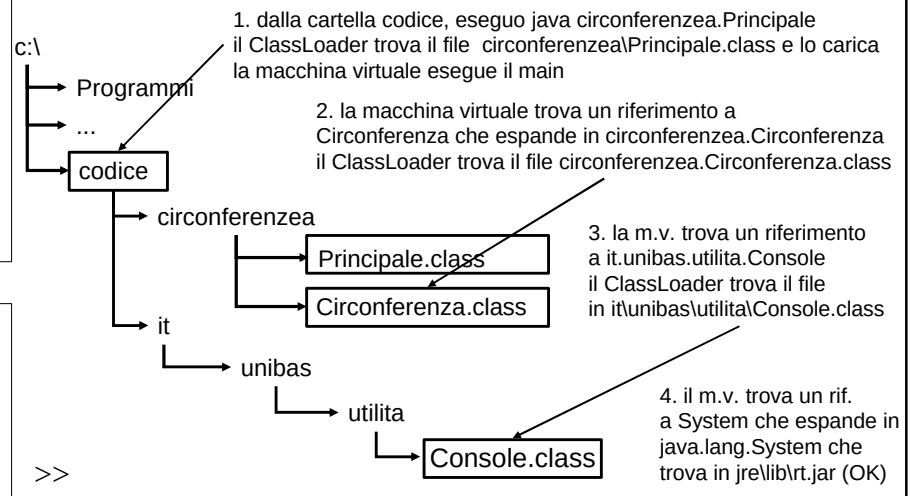
Macchina Virtuale

- Esempio
  - ⇨la macchina virtuale chiede al ClassLoader di trovare circonferenzea.Principale per avviare l'esecuzione
  - ⇨per prima cosa il ClassLoader risale dal nome della classe, completo di package, al file .class che contiene la classe
  - ⇨la ricerca avviene secondo le stesse regole del compilatore: nelle sottocartelle della cartella corrente e nel file rt.jar di %JAVA_HOME%\jre\lib
  - ⇨se il ClassLoader trova il file lo carica, altrimenti restituisce un errore (NoClassDefFoundError)

Macchina Virtuale

- Il processo di esecuzione (continua)
  - ⇨a questo punto la macchina virtuale comincia l'esecuzione dal metodo main
  - ⇨se non c'è metodo main, restituisce un errore
  - ⇨durante l'esecuzione del codice della classe, la macchina virtuale incontra riferimenti ad altre classi (nome package e nome della classe)
  - ⇨a questo punto si avvia il processo di collegamento dinamico
  - ⇨la macchina virtuale chiede al ClassLoader di ritrovare e caricare il file secondo le stesse regole

Macchina Virtuale



Strumenti

- Di conseguenza
  - ⇨dal punto di vista sintattico, le classi “visibili” in una classe java sono tutte le classi raggiungibili dal compilatore durante la compilazione e durante la macchina virtuale durante l’esecuzione
- Concetto dinamico di visibilità
  - ⇨dipende dalla struttura delle cartelle al momento della compilazione e dell’esecuzione

Il Concetto di Classpath

ATTENZIONE
al concetto di
Classpath

- Classi visibili in un’applicazione Java
  - ⇨classi delle API standard (in rt.jar)
  - ⇨tutte le classi che durante la compilazione e l’esecuzione sono raggiungibili a partire dal classpath
- Classpath di Java
  - ⇨insieme di percorsi per la ricerca delle classi
  - ⇨collezione di riferimenti a cartelle del disco e a file .jar

Il Concetto di Classpath

- Il Classpath standard
  - ⇨la cartella corrente da cui viene eseguito il compilatore/macchina virtuale
  - ⇨ovvero: la cartella .
  - ⇨ma può essere modificato aggiungendo ulteriori cartelle e file .jar (>>)
- Cambiamento del Classpath di Java
  - ⇨richiede la definizione di una variabile di ambiente denominata CLASSPATH

Riassumendo

- Strumenti
- Compilatore
- Macchina Virtuale
- Il Concetto di Classpath

Termini della Licenza