Fondamenti di Informatica A
Laboratorio 5
Programmazione in Java static

Danilo Pianini
danilo.pianini@unibo.it
Mirko Viroli
mirko.viroli@unibo.it

Alma Mater Studiorum—Università di Bologna

10 aprile 2013

Operazioni preliminari

- Accendere il computer
- Effettuare il log-in con le proprie credenziali istituzionali
  - Normalmente sono simili a:
    `nome.cognome@studio.unibo.it`
  - Speriamo che oggi funzioni :)
- Aprire Google Chrome, o altro browser
- Collegarsi al sito `http://bit.ly/finfa2013-e5`
- Scaricare l'esercitazione odierna
- Collegarsi al sito `http://bit.ly/finfa2013-e5-zip`
- Scaricare il file tar.gz contenente i sorgenti
- Decompattare l'archivio sul Desktop
  - In laboratorio, aperto lo zip, copiare e incollare i sorgenti sul desktop
  - Verrà utilizzato più avanti nell'esercitazione

Elementi base del file system

- I sistemi operativi odierni consentono di memorizzare permanentemente le informazioni su supporti di memorizzazione di massa (dischi magnetici, dispositivi a stato solido), unità ottiche (CD, DVD, Blu-Ray), memory stick, ecc...
- Le informazioni su questi supporti sono organizzate in files e cartelle:
  - i files contengono le informazioni;
  - le cartelle sono contenitori, all'interno contengono i files ed altre cartelle.
- La cartella più esterna, che contiene tutte le altre, è detta root. Essa è il livello gerarchico più alto nel file system.
  - In Windows, ciascun file system ha come root una lettera di unità (e.g. `C:`, `D:`).
  - In Unix (Linux, Mac OS, BSD, Solaris, etc), vi è una unica radice, ossia `/`
- La stringa che descrive un intero percorso dalla root fino ad un elemento del file system prende il nome di directory (e.g. `C:\Windows\win32.dll`, `/home/user/frameworkFS.jar`)

Manipolare il file system

L'utente può osservare e manipolare il file system:

- sapere quali files e cartelle contiene una cartella
- creare nuovi files e cartelle
- spostare file e cartelle dentro altre cartelle
- rinominare files e cartelle
- eliminare files e cartelle

Il software deputato ad osservare e manipolare il file system prende il nome di file manager.

- Su Windows, esso è "Esplora risorse" (`explorer.exe`)
- Su MacOS, il principale è "Finder"
- Su Linux (e Android) ne esistono diversi (Nautilus, Dolphin, Thunar, Astro...)

Provate ad effettuare le seguenti operazioni (utilizzando il file manager):

1. Creare due cartelle, di nome `f1` ed `f2`
2. Rinominare `f2` in `f12`
3. Spostare `f12` dentro `f1`

Il terminale

Il terminale consente di impartire comandi al computer tramite testo.

- Nell'antichità (in termini informatici) le interfacce grafiche erano sostanzialmente inesistenti, e l'interazione con i calcolatori avveniva di norma tramite interfaccia testuale
- Tutt'oggi, le interfacce testuali sono utilizzate:
  - è spesso più veloce eseguire un comando che effettuare operazioni con il mouse
  - operazioni complesse possono esser svolte con comandi semplici
  - non tutti i software sono dotati di interfaccia grafica
  - alcune opzioni di configurazione del sistema operativo restano accessibili solo via linea di comando
    - (anche su Windows: ad esempio il comando "magico" che cambia l'associazione dei file jar e vi fa funzionare il framework)

Struttura di un comando

Tutti i comandi condividono una struttura comune

`commandName parameters arguments`

- Solitamente i parametri vengono prefissi da un simbolo `-` o `--`, o (in ambiente Windows) `/`

Alcuni esempi (da capire, non da eseguire!)

`java -jar frameworkFS.jar`

- **Cosa fa?** Avvia una macchina virtuale java, chiedendo l'esecuzione del file "frameworkFS.jar"
- **Nome comando:** `java`
- **Parametri:** `-jar`
- **Argomenti:** `frameworkFS.jar`

`rm -r -f /home/user/uselessData /home/user/other`

- **Cosa fa?** Rimuove le directory `/home/user/uselessData` e `/home/user/other`, tutte le cartelle in esse contenute e tutti i file in esse contenute.
- **Nome comando:** `rm` (sta per remove)
- **Parametri:** `-f` (sta per "force"), `-r` (sta per "recursive")
- **Argomenti:** `/home/user/uselessData`, `/home/user/other`

Aprire un terminale in laboratorio

- In laboratorio, troverete il terminale (prompt dei comandi) clickando su Start $\Rightarrow$ Programmi $\Rightarrow$ Accessori $\Rightarrow$ Prompt dei comandi
- Metodo più rapido: Start $\Rightarrow$ Nella barra di ricerca, digitare `cmd` $\Rightarrow$ clickare su `cmd.exe`

Manipolare il file system dal terminale

Operazione	Comando Unix	Comando Win
Visualizzare la directoy corrente	`pwd`	`echo %cd%`
Eliminare il file `f` (non va con le cartelle!)	`rm f`	`del f`
Eliminare la directory `nd`	`rm -r nd`	`rd nd`
Contenuto della directory corrente	`ls -l`	`dir`
Avviare un eseguibile `p`	`./p`	`.\p`
Cambiare unità disco (passare a D:)	–	`D:`
Passare alla directory `nd`	`cd nd`	`cd nd`
Passare alla directory di livello superiore	`cd ..`	`cd..`
Spostare un file `f1` in `f2`	`mv f1 f2`	`move f1 f2`
Copiare il file `f` in `fc`	`cp f fc`	`copy f fc`
Creare la directory `d`	`mkdir d`	`md d`

Eseguire delle prove ed esser certi di aver compreso come utilizzare ogni comando. Per *cominciare* l'esame, in particolare, dovrete usare il comando `cd`: siate certi di aver capito cosa fa!

Piccole furberie da terminale

Autocompletamento

Sia Unix che Windows offrono la possibilità di effettuare autocompletamento, ossia chiedere al sistema di provare a completare un comando. Per farlo si utilizza il tasto "tab" (quello con due frecce orientate in maniera opposta, sopra il lucchetto). Si testi la funzione di autocompletamento sul proprio sistema operativo.

Memoria dei comandi precendenti

Sia Unix che Windows offrono la possibilità di richiamare rapidamente i comandi inviati precedentemente premendo il tasto "freccia su". I sistemi Unix supportano anche il lancio di comandi eseguiti in sessioni precedenti (se inviate un comando e riavviate il sistema, e al riavvio premete "freccia su", vi apparirà il comando precedente.

Interruzione di un programma

È possibile interrompere forzatamente un programma (ad esempio perché inloopato). Per farlo, sia su Windows che in Unix, si prema ctrl+c.

Utilizzo di `javac`

`javac` è il compilatore Java. Consente di tradurre un file di testo che contiene codice Java (tipicamente con estensione `.java`) in un file interpretabile dalla Java Virtual Machine (estensione `.class`)

Esempio d'uso

1. Si apra un terminale
2. Utilizzando il comando `cd` come descritto nella Slide 8, ci si posizioni nella cartella nella quale si è decompattato l'archivio
3. Utilizzando il comando `dir` (o `ls`, per chi usa Unix) come descritto nella Slide 8, si verifichi che la cartella contenga un insieme di files con estensione `.java`
4. Si lanci il comando `javac Empty.java`
5. Si verifichi che un file .class è stato creato nella cartella medesima

Utilizzo di `java`

Una volta che il software è stato compilato, può essere eseguito. Per eseguire un software Java, si invoca il comando `java`, che crea una istanza della Java Virtual Machine e le dà in pasto ciò che si desidera eseguire.

Esempio d'uso

1. Utilizzando il terminale precedente
2. Si lanci il comando `java Empty`
   - Chiede a Java di eseguire, se possibile, la classe `Empty`
3. Si ottiene un errore: lo si legga attentamente. Cosa significa?

Utilizzo di `java` - funzione `main`

Come fa la JVM a capire COSA eseguire?

- Un programma può contare di centinaia (ma anche migliaia, milioni) di funzioni. Come fa la Java Virtual Machine a capire cosa eseguire, ed in che ordine?
- La funzione che viene invocata è `static void main(String[])`
  - funzione di nome `main`
  - non ha alcun ritorno
  - prende in ingresso un array di `String`. Al momento dell'invocazione, gli argomenti passati al comando java dopo il primo (ossia, passati dopo la classe da eseguire) vengono inseriti nell'array di String e passati alla funzione `main`, che può eventualmente utilizzarli
- Quando si lancia `java ClassName`, quello che accade è che la JVM cerca la classe `ClassName`, cerca al suo interno la funzione `main`, quindi la esegue.

Scegliere un editor

A partire da questo laboratorio, si fa sul serio. Scriveremo finalmente dei programmi veri, fuori dall'ambiente limitante dei framework. Chiunque debba scrivere del codice, ha bisogno di un buon editor di testo. Alcune dei desiderata per un buon editor:

- Code highlighting: la capacità di evidenziare in maniera chiara le diverse parti di codice. Aiuta chi legge a capire cosa ci sia scritto.
- Code suggestions: la capacità di completare automaticamente nomi parzialmente scritti. Velocizza drammaticamente la scrittura e previene errori di typing.
- Supporto a differenti linguaggi: a volte uno stesso software è scritto con una combinazione di linguaggi. Un editor che è in grado di distinguere i diversi sorgenti e applicare il code highlighting correttamente a ciascun linguaggio può essere utilizzato per l'intero progetto.
- Possibilità di vedere e lavorare contemporaneamente più sorgenti.
- Funzionalità ancora più avanzate (organizzare i sorgenti, compilare automaticamente, eseguire dei test in modo automatizzato, etc.) consentono ad alcuni tool di passare da semplici editor di testo a veri e propri *Source Development Kit* (SDK)

In questo corso, dato lo scopo di insegnarvi l'informatica di base, non vi è consentito l'uso di SDK: lavoreremo con degli editor, utilizzando i comandi da terminale per compiere operazioni di compilazione ed esecuzione. Chi vorrà approfondire, venga a seguire il corso di Object Oriented Programming ad Informatica :)

Jedit

L'editor che abbiamo scelto per voi è JEdit, scritto interamente in Java.

- Interamente scritto in Java
- Supporta il code highlighting sia per Java che per C
- Consente di lavorare su più sorgenti contemporaneamente
- Esistono editor anche più avanzati, ma vi consigliamo Jedit
  - Anche perché all'esame, comunque, lo useremo

Compilare ed eseguire un programma Java

Utilizzare JEdit per vedere il contenuto di un sorgente

1. Aprite JEdit
2. Aprite in JEdit il file `ArgumentEcho.java` che si trova nella cartella dove avete scompattato i sorgenti
3. Osservate il codice: cosa realizza?
4. Utilizzando il comando `javac` adeguatamente, compilate il file
5. Utilizzando il comando `java` adeguatamente, eseguite il programma
6. Cosa succede se eseguite il seguente comando?
   - `java ArgumentEcho I love this course`

Costruire un programma Java data una funzione

Si sconsideri la seguente funzione:

```
static boolean isPrime(int n) {
  if (n == 2 || n == 3) {
    return true;
  }
  if (n < 2 || n % 2 == 0) {
    return false;
  }
  int i = 3;
  for (; i < n / 2 && n % i != 0; i+=2);
  return i == n/2 || i == n/2 + 1;
}
```

1. Si crei un nuovo file `PrimeDetector.java`
2. All'interno di `PrimeDetector.java`, si incolli la funzione data
3. Si scriva un test per la funzione `isPrime(int)`
4. Si scriva un main che esegua il test
5. Si compili il file `PrimeDetector.java`
6. Si esegua il programma
7. Considerando la funzione (esistente in Java) `int Integer.parseInt(String)` che, data una String con caratteri numerici (e.g. `"123"`), torna la sua rappresentazione come `int`; si modifichi il main per far sì che verifichi che il software verifichi e stampi a video se il primo argomento passato sia un primo o meno.

Funzioni di libreria: da `String` a tipo primitivo

Di seguito verranno elencate delle funzioni di libreria Java che consentono di tradurre un valore in formato Stringa in un tipo di dato diverso

- `boolean Boolean.parseBoolean(String)`
- `double Double.parseDouble(String)`
- `float Float.parseFloat(String)`
- `int Integer.parseInt(String)`
- `long Long.parseLong(String)`

Funzioni di libreria: `Math`

Java offre un'ampia libreria con funzioni matematiche. È possibile computare (arco)seno, (ar)coseno, esponenziale, radici, elevamenti a potenza, logaritmi, numeri casuali, eccetera. Per una panoramica completa, si veda la pagina:
`http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html`
L'uso di queste funzioni è generalmente consentito

Funzioni di libreria: `Arrays`

Java offre alcune funzioni per manipolare gli array, all'interno della classe `Arrays`. Nel seguito, con `T[]` si indicherà un generico array. Le funzioni presentate sono infatti applicabili a qualunque tipo di array.

- `boolean Arrays.equals(T[], T[])` – Torna true se i due array hanno uguale contenuto (stessi elementi nelle stesse posizioni).
- `void Arrays.sort(T[])` – Modifica l'array passato ordinandolo in maniera ascendente
- `String Arrays.toString(T[])` – Torna una rappresentazione in forma di stringa dell'array. Molto utile per visualizzare il contenuto degli array.

L'uso di queste funzioni è generalmente consentito
ATTENZIONE! Per usare queste funzioni, occorre scrivere come prima linea di codice del sorgente:

```
import java.util.*;
```

Si veda in merito il sorgente `Import.java` del pacchetto.

Consegna dei prossimi esercizi

1. Si crei, per ciascun esercizio, un nuovo file `.java`
2. All'interno del file, si crei una classe con lo stesso nome del file
3. All'interno della classe, si implementi la funzione richiesta
4. Si scriva un test per la funzione
5. Si scriva un main che esegue il test e ne stampa il risultato
6. Una volta realizzato il main, se richiesto nell'esercizio, si aggiunga al main la gestione degli argomenti passati dall'utente
7. Si mostri il risultato al docente

RandomArray

Si crei il programma `RandomArray`, che, dato un numero intero `n` crea un array di `int` di lunghezza `n` e lo riempie con valori casuali compresi fra 0 e 1000, quindi lo mostra a video. Si ricorda che:

- la funzione `double Math.random()` torna un numero casuale compreso fra 0 ed 1. Questo può essere opportunamente trasformato in un numero casuale fra 0 e 1000.
- la funzione `String Arrays.toString(int[])` torna una rappresentazione come `String` del contenuto dell'array

Si modifichi il programma affinché:

- Se viene passato un argomento, esso deve essere `n`
- Se non viene passato alcun argomento, viene assunto `n = 10`

Per questo peculiare programma non è necessario scrivere un test.

Palindrome

Si crei il programma `Palindrome`, che, dato un numero intero `n`, crea un array di `int` che segue lo schema seguente:

- $0 \Rightarrow \{0\}$
- $1 \Rightarrow \{1,0,1\}$
- $2 \Rightarrow \{2,1,0,1,2\}$
- $5 \Rightarrow \{5,4,3,2,1,0,1,2,3,4,5\}$

Il programma deve essere realizzato in maniera *iterativa*. Si modifichi il programma affinché:

- Se viene passato un argomento, esso deve essere `n`
- Se non viene passato alcun argomento, viene assunto `n = 0`
- Si affianchi alla versione iterativa già realizzata una implementazione ricorsiva (ossia, si traduca la funzione realizzata in modo iterativo affinché usi la ricorsione).

PushZeroes - opzionale

Si crei il programma `PushZeroes`, che, dato un `int[]`, crea un nuovo array nel quale ad ogni 1 segue uno 0. Alcuni esempi:

- $\{\} \Rightarrow \{\}$
- $\{1,1,1,1\} \Rightarrow \{1,0,1,0,1,0,1,0\}$
- $\{1,0,1\} \Rightarrow \{1,0,0,1,0\}$
- $\{1,0,1,0\} \Rightarrow \{1,0,0,1,0,0\}$
- $\{0,1,2,3\} \Rightarrow \{0,1,0,2,3\}$
- $\{2,3,4,5\} \Rightarrow \{2,3,4,5\}$

Se eseguito, il programma deve eseguire un test e stampare:

1. Quali ingressi vengono dati alla funzione
2. Quale uscita dà la funzione
3. Il risultato del test (true o false)