Fondamenti di Informatica A
Laboratorio 3
Basi di programmazione strutturata

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

Alma Mater Studiorum—Università di Bologna

27 marzo 2013

Operazioni preliminari

- Accendere il computer
- Effettuare il log-in con le proprie credenziali istituzionali
  - Normalmente sono simili a:
    `nome.cognome@studio.unibo.it`
- Aprire Google Chrome, o altro browser
- Collegarsi al sito `http://bit.ly/finfa2013-e3`
- Scaricare l'esercitazione odierna

Verifica del funzionamento del Framework F-S

- Sul desktop, troverete la cartella "framework"
- Fate doppio click sulla cartella
- Si aprirà una nuova finestra, all'interno della quale troverete tre software
- Fare doppio click sull'icona del FrameworkFS
- Inserire nel campo "Blocco da eseguire" il testo `println("Hello World");`
- Premere il tasto "Esecuzione"
- Verificare che il risultato sia "`Hello World`"
- In caso di risultato diverso dalle attese, chiamare subito il docente o il tutor

Come comportarsi di fronte agli errori

- Errare humanum est
  - Ed è anche molto meglio sbagliare in laboratorio che all'esame!
- È importante, di fronte agli errori, non scoraggiarsi, leggere **attentamente** il messaggio di errore, **capirlo** e studiare una contromisura.
- Java segnala gli errori in maniera piuttosto precisa...
- ...il framework, che ne usa una sottoparte molto ristretta, molto meno!
- In caso di errore, il framework vi mostrerà un messaggio che spiega cosa (il compilatore pensa che) sia successo, seguito dal codice Java reale che è stato invocato.

Come esercitarsi con il Framework

- Questo framework è abbastanza diverso dai due precedenti
- Serve ad abituarvi ad utilizzare un linguaggio imperativo
- Come in F-F, il blocco "Dichiarazione di funzioni" conterrà le funzioni che vorrete definire
- il blocco "Blocco da eseguire" potrà contenere una serie di statement, fra i quali due istruzioni (non esistenti in Java o C ma ammesse all'interno del frameworkFS)
  - `print(String s)`: scrive nel risultato la stringa `s`. Ad esempio, `print("ciao");` mostrerà nel risultato `ciao`.
  - `println(String s)`: scrive nel risultato la stringa `s`, quindi va a capo. Ad esempio, `print("ciao");` mostrerà nel risultato `ciao`, quindi andrà a capo.

Cosa fa il blocco seguente?

```
println("ciao");
print("ciao");
print(" ");
print("Lulù");
```

Formattazione

A partire da questo laboratorio, è **richiesto** di formattare correttamente il codice. Perché?

- Il codice correttamente formattato è più comprensibile agli altri che lo leggono
- Il codice correttamente formattato sarà più comprensibile a voi quando lo rileggerete dopo del tempo
- Esistono linguaggi (ad esempio Python, molto diffuso) che utilizzano la formattazione del codice per definire la **semantica** di un programma, quindi un codice mal formattato in quei linguaggi semplicemente **non funziona come atteso**.
- Meno importante in senso assoluto, ma importante per voi: all'esame, gli errori di formattazione verranno valutati. Non perdete punti per simili sciocchezze!

Formattazione, semplice esempio

```
int fatt(int n){if(n==0)
{return 1;}else{return n*fatt(n-1);}}
```

La funzione sopra, perfettamente funzionante in Java, non ha una struttura chiaramente comprensibile nonostante sia estremamente semplice. Si guardi la stessa meglio formattata:

```
int fatt(int n) {
   if(n==0) {
      return 1;
   } else {
      return n*fatt(n-1);
   }
}
```

Meglio, o no? Immaginate cosa possa succedere con programmi di qualche centinaio di righe di codice: se mal formattati, diventano impossibili da comprendere.

Uso base del framework

Si consideri la funzione `int fatt(int n)` prima introdotta.

- Si testi il funzionamento della funzione, stampando il risultato di `fatt(5)`. Per farlo, si ricorda l'utilizzo della funzione `void print(String s)`.
- Si scriva un blocco da eseguire che stampi, uno sotto l'altro, il risultato delle invocazioni della fuzione per input da 0 a 5
- Si scriva un blocco da eseguire che stampi, uno di seguito all'altro, separato da virgole, il risultato delle invocazioni della fuzione per input da 0 a 5 (il risultato atteso è `1, 1, 2, 6, 2, 120`).

Creazione di variabili

È possibile definire nuove variabili tramite l'assegnamento di due elementi:

- Un tipo di dato
- Un nome di variabile
- Esempi:
  - `int pippo;` Nuova variabile di tipo `int`, di nome `pippo`
  - `double pluto2;` Nuova variabile di tipo `double`, di nome `pluto2`
  - `int[] alda_merini;` Nuova variabile di tipo `int[]`, di nome `alda_merini`

Assegnamento di variabili

Una volta che una variabile è stata dichiarata, è possibile assegnarle un valore, tramite l'operatore =.

- = in Java (ma anche in C, C#, C++ ed in una miriade di linguaggi simili) **non significa uguale**, ma **assegnamento**.
- Non confondere ==, operatore di uguaglianza che ritorna un boolean, con =, operatore di **assegnamento**.
- Analogamente al tipo di ritorno di una funzione, un assegnamento va a buon fine se il tipo che si sta provando ad assegnare è uguale, o convertibile per coercizione, al tipo della variabile.
- È **possibile** condensare in un'unica linea la dichiarazione di una variabile ed il suo assegnamento (e.g. `int s = 4;`). Anche se fatte nel medesimo statement, dichiarazione ed assegnamento sono comunque due cose distinte.

Creazione ed assegnamento di variabili

Si considerino le seguenti funzioni:

```
int fi(){
   return 2;
}
double fd(){
   return 1.1;
}
boolean fb(){
   return true;
}
```

Per ciascuno dei seguenti blocchi da eseguire, si intuisca quali danno errore e quali no, e di quelli che non devono ritornare errore, quale sia il risultato atteso. Di quelli che ritornano errore, si definisca il perché. Si usi il framework per verificare. In caso di non consistenza con i risultati attesi, si interpelli il docente.

Creazione ed assegnamento di variabili

```
int i;
i = 0;
print(i+"");
```

```
int i;
i = 1.1;
print(i+"");
```

```
int i;
print(i+"");
```

```
int i;
i = fi();
print(i+"");
```

Creazione ed assegnamento di variabili

```
double i;
i = 0;
print(i+""); // Perché non 0?
```

```
int i;
i = fd();
print(i+"");
```

```
boolean i;
i = fb() ^ i;
print(i+"");
```

```
int x;
int y = 9;
x = 10 - (y=(y-2)); // Che brutta cosa! Don't try this at home ;)
print("x = " + x + ", y = " + y);
```

Si riscriva l'ultimo caso in maniera tale che le operazioni siano eseguite in statement separati.

Operatore di incremento

In Java (come in C), un valore `int` (o `long`) può essere incrementato in tre modi differenti. Si consideri la dichiarazione di variabile `int i = 0;`:

- `i++`: prima valuta il valore di i, quindi lo incrementa.
- `++i`: prima incrementa il valore di i, quindi lo valuta.
- `i+1`: non modifica il valore di i, ma valuta la somma.

ATTENZIONE! Queste tre forme, sebbene svolgano una funzione analoga, sono **diverse**!
Considerando la seguente funzione:

```
int f(int i){
   return i;
}
```

Per ciascuno dei blocchi della slide seguente, si preveda l'output del programma. Si verifichi con il framework, quindi si interpelli il docente se il risultato è diverso da quanto previsto.

Operatore di incremento

```
int a = 0;
int b = f(a+1);
print("a = " + a + ", b = " + b);
```

```
int a = 0;
int b = f(a++);
print("a = " + a + ", b = " + b);
```

```
int a = 0;
int b = f(++a);
print("a = " + a + ", b = " + b);
```

Si costruiscano gli esempi analoghi per l'operatore di decremento, si preveda e si verifichi il loro funzionamento.

Costrutto `if`

Il costrutto `if` consente di effettuare una serie di operazioni solo a patto che una certa espressione booleana sia vera.

```
boolean f(int[] a, int n){
   if(a.length < 1) {
      return false;
   }
   return f(a, n, 0);

}
boolean f(int[] a, int n, int i){
   if(i >= a.length) {
      return false;
   }
   if(a[i] == n){
      return true;
   }
   return f(a, n, i+1);
}
```

- Cosa realizza la funzione `f(int[], int)` qui sopra?
- Dopo averlo capito, effettuare una serie di prove con il framework per verificare.

Costrutto `if - else`

Il costrutto `else` può essere utilizzato, dopo un `if`, per stabilire una parte di codice da eseguire **solo se** la condizione dell'`if` è falsa. è anche possibile utilizzare else if, per verificare ulteriori condizioni solo se il controllo degli if precedenti l'else sono falliti.

```
int f(int[] a) {
   return f(a, 0, 0);
}
int f(int[] a, int i, int s) {
   if (i >= a.length) {
      return s;
   }
   if (a[i] == 0) {
      s = 0;
   } else if (a[i] % 2 == 0) {
      s++;
   } else {
      s--;
   }
   return f(a, i + 1, s);
}
```

- Cosa realizza la funzione `f(int[])` qui sopra?
- Dopo averlo capito, effettuare una serie di prove con il framework per verificare.

Costrutto `for`

Il costrutto `for` consente di effettuare iterativamente delle operazioni fintanto che una condizione booleana è soddisfatta. Inoltre, consente di definire una operazione preliminare, ed una operazione da effettuare al termine di ciascuna iterazione.

```
int f(int[] a) {
   int s = 0;
   for (int i = 0; i < a.length; i++) {
      if (a[i] % 2 == 0) {
         s = s + a[i];
      }
   }
   return s;
}
```

- Cosa realizza la funzione `f(int[])` qui sopra?
- Quante iterazioni vengono effettuate quando si invoca `f(new int[]{2,1,5,9,7,3,5,154})`? Quante volte viene testata la condizione del for?
- Dopo averlo capito, effettuare una serie di prove con il framework per verificare.

Analisi di funzioni date

Controllando ad occhio il codice delle seguenti funzioni, intuire quale algoritmo implementano, e solo dopo verificare utilizzando il framework il loro funzionamento, facendo qualche prova.

```
double f1(int[] a ){
   int acc = 0;
   for (int i=0;i<a.length;i++) {
      acc += a[i]; /* Operatore di incremento!
         Sarebbe meglio non usarlo normalmente. */
   }
   return (double) acc / a.length;
}
```

- Quale informazione si vuole avere sull'array in input?
- Perché si effettua un'operazione di cast prima di ritornare il risultato?

Analisi di funzioni date

```
boolean f2(int[] a) {
    int i = 0;
    for (; i < a.length && a[i] == 1; i++);
    return i == a.length;
}
```

- Quale grammatica riconosce?
- Perché `int i = 0` non è dichiarata come prima istruzione del `for`?
- Qual è il significato del controllo finale?
- Quante cicli vengono effettuati all'interno del `for` se viene invocata `f2(new int[]{1,1,1,1,1,1,0,1,1,2})`?

Analisi di funzioni date

```
int f3(int[] a) {
   int count = 0;
   for (int i = a.length - 1; i>=0; i--){
      if (a[i]>0) {
         count++;
      }
   }
   return count;
}
```

- Si esegua passo-passo su carta l'invocazione `f3(new int[]{-5,-7,5,4,3})`
- Quale funzione è realizzata?

Come si crea una funzione?

Non esiste (in generale) un metodo meccanico che vi consenta di realizzare l'algoritmo che desiderate. Esiste però una serie di operazioni che potete effettuare per semplificare la creazione di nuove funzioni.

1. Resistete all'istinto di scrivere subito il codice: i linguaggi di programmazione (tutti), sono meno malleabili del linguaggio naturale. Se partite scrivendo subito codice, alla difficoltà dello studio dell'algoritmo si aggiunge quella del linguaggio.
2. Capite con molta attenzione cosa vi viene richiesto per risolvere il problema. Accertatevene anche osservando l'eventuale test, o scrivendone uno se non viene fornito. Per ciascun possibile input della funzione, dovete essere in grado di associare un output.
3. Ragionate sul procedimento mentale che vi porta risolvere il problema, cercando di estrarre, sempre in linguaggio naturale, i passi che vi portano dall'input all'output.
4. Una volta ottenuto l'algoritmo in linguaggio naturale, lo si deve tradurre in Java.
5. A questo punto, in caso di errori, di esecuzione, si deve eseguire passo-passo su carta ciò che si è realizzato, cercando di capire quando il risultato diverge da quanto atteso.

Esempio di algoritmo in linguaggio naturale

L'algoritmo in linguaggio naturale non potrà esser dato in pasto ad una macchina, ma una volta scritto deve consentire a qualunque essere umano, dato un input, di ottenere il vostro stesso output.

- Esempio di come potrebbe apparire un algoritmo in linguaggio naturale:
  1. Dato un array, lo scorro fintanto che vedo "1" oppure "2",
  2. una volta terminato, verifico che vi sia un altro valore, che deve essere "3", e deve essere l'ultimo elemento dell'array. Se è così, torno vero.
  3. se non è così, allora torno falso.
- Quale grammatica riconosce?
- Lo si traduca dal linguaggio naturale a Java.
- Si mostri al docente il risultato.

Creazione guidata di funzioni

Si desidera costruire il riconoscitore per la grammatica $0\{1\}$. Un possibile algoritmo in linguaggio naturale potrebbe essere:

1. Dato un array, verifico che abbia almeno un elemento, e che il primo elemento sia zero. Se non è così, torno falso.
2. Se è così, scorro l'array a partire dal secondo elemento, e proseguo fintanto che vedo degli "1".
3. Se, quando ho finito di scorrere, ho analizzato tutti gli elementi dell'array, torno true. Altrimenti torno false.

- Si costruisca la funzione Java corrispondente.
- Si scriva una funzione di test che ne verifichi il corretto comportamento.
- Si mostri al docente il risultato.
- Analogamente, si costruisca il riconoscitore per la grammatica $0\{1\}2$ e si scriva un test. Si mostri al docente il risultato.

Creazione di una funzione

Si desidera costruire una funzione che, dato un array `a` ed un intero `k`, calcoli la somma degli ultimi `k` valori di `a`.

- Si scriva in linguaggio naturale un algoritmo che risolve il problema.
- Si mostri al docente l'algoritmo.
- Si costruisca la funzione Java corrispondente.
- Si scriva una funzione di test che ne verifichi il corretto comportamento.
- Si mostri al docente il risultato.

Al termine dell'esercitazione

- Ricordarsi di effettuare il logout
  - Dal menu Start, selezionare "Disconnetti"
- Una volta disconnessi:
  - Se siete del primo turno, lasciate le macchine accese, di modo che i vostri colleghi del turno successivo siano agevolati
  - Se siete del secondo turno, spegnete le macchine