Fondamenti di Informatica A
Laboratorio 1
Valutazione e typing di espressioni

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

Alma Mater Studiorum—Università di Bologna

14 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-e1`
  - in “e1”, il carattere “1” è un uno, non una L minuscola!
- Scaricare l’esercitazione odierna

Verifica del funzionamento del Framework F-V

- 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 FrameworkFV
- Inserire nel campo “Espressioni da valutare” il testo $1 + 2$
- Premere il tasto “Valutazione”
- Verificare che il risultato sia “3”
- In caso di risultato diverso dalle attese, chiamare subito il docente o il tutor

Come esercitarsi con il Framework

- Nelle slides seguenti vi verranno proposti una serie di esempi di espressioni da valutare
- **Prima** di eseguirle nel programma, cercare **sempre** di anticipare quale sarà il risultato della valutazione
- Se quanto avete pensato corrisponde a quanto risponde il software, bene!
- Se non corrisponde, bene lo stesso, a patto che ne capiate la ragione
- Se ottenete un risultato inaspettato e non sapete giustificarlo, consultate il docente!
- Al termine dell'esercitazione, **effettuate il log out** dalla macchina che state utilizzando.

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 di errore, seguito dal codice Java reale che è stato invocato.

Operazioni su booleani

- `true`
- `false`
- `!false`
- `true & false`
- `true | false`
- `true | true & false | false` — Quale operatore è prioritario? In che ordine si valuta?
- `true & true | false & false`
- `true & (true | false) & false`

Operazioni su interi – elementi di base

- `1+2`
- `5*8`
- `10/2`
- `10/3` — Quanto fa? 3, $3.\bar{3}$ o 4?
- `10%3` — Classi di resto, come in Geometria
- `6/2*(1+2)` — Quanto fa? 6 o 9?
- `4+4*4+4*4-4/4+4*4`

Operazioni su interi – complemento a 2, overflow

In Java, gli interi negativi sono rappresentati in complemento a 2. Altri linguaggi, come C, consentono di dichiarare `unsigned` alcuni tipi di dato, ossia mai negativi. Java non offre questa possibilità. Calcolate **a mano** il complemento a due **prima** di eseguire!

- `8-7` — Trasformate 7 in complemento a 2, quindi sommate!
- `2147483647+1` — Cos'è successo??? Suggerimento: in Java gli `int` si rappresentano con **32bit**!
- `2147483647*2`
- `-2147483648-1`
- `-(-2147483648)`
- `-(-2147483648L)` — La `L` finale indica a Java di interpretare quel numero come tipo `long`. I `long`, a differenza degli `int`, hanno **64bit**.

Operazioni su numeri in virgola mobile

Java (così come C) consente di operare con numeri decimali. Ne esistono di due tipi `float`, rappresentati con **32bit**, e `double`, rappresentati con **64bit**.

- `1.0` — Aggiungere una parte decimale ad un numero lo fa interpretare a Java come `double`.
- `1+2.1` — Un'operazione fra un `int` e un `double` ha come risultato un double
- `10.0/3` — Confrontare il risultato con `10/3`
- `10d/3` — Posporre `d` ad un numero forza Java ad interpretarlo come `double`.
- `10f/3` — Posporre `f` ad un numero forza Java ad interpretarlo come `float`. Notare come i `float` siano meno precisi dei `double`, confrontando questo risultato con il precedente
- `15.45E24` — Supporto alla notazione scientifica

Precisione dei numeri in virgola mobile

Internamente, sia i double che i float, riservano alcuni bit alla rappresentazione dell'esponente, ed altri alla rappresentazione della parte significativa (mantissa): esattamente come i numeri scritti in notazione scientifica.

- `1e10+1e-5`
- `1e10+1e-6` — Troppe cifre significative per rappresentare correttamente il numero!
- `1e10+1e-7` — I numeri sono di ordine tanto diverso che a fronte della somma il secondo viene "mangiato".
- `1+1e-15`
- `1+1e-15*4` — La moltiplicazione viene fatta per prima, dunque non si ha perdita di precisione.
- `1+1e-15*3+1e-15`
- `4+1e-15+1e-15+1e-15+1e-15`

Operazioni su stringhe

Java consente di manipolare anche le stringhe con l'operatore "+". Questa caratteristica è assente o differente in molti altri linguaggi (in C ad esempio).

- `"a"+"b"`
- `"1"+"2"` — Le stringhe non vengono convertite in numeri automaticamente!
- `1+"2"` — La somma di un numero e una stringa, ha come risultato una stringa
- `1+2+"3"+4+5`
- `1+2+"3"+"4"+"5"`

Arrays

- `new int[]{}` — Tipo per `riferimento`!
- `new int[]{}.length`
- `new int[]{1, 2, 3}`
- `new int[10].length`
- `new boolean[]{true, true, false}[1]`
- `new String[]{“a”,“b”}[0]`

Operatori bitwise: shift

Java (come C) consente di operare anche a livello di bit.

- $1 << 0$
- $1 << 3$
- $1 << 32$
- $(1 << 2) << 3$
- $0xf << 4$
- $0xabcd << 8$
- $100 >>> 1$ — Non considera il segno!
- $-100 >> 1$
- $-100 >>> 1$ — Visto?

Operatori bitwise: & e |

Java (come C) consente di operare anche a livello di bit.

- `1 | 2`
- `1 | 4`
- `12345 & 54321`
- `-12345 & 54321`
- `12345 & -54321`
- `-12345 & -54321`
- `1<<7 | 1<<10`
- `0xffffffff & 0x0000ffff` — L'operatore & è spesso usato per "mascherare" una parte dati considerata non significativa
- `0xffffffff & 0xffff0000`
- `0xf<<12 | 0xf<<4`
- `1<<12 | 2<<12 | 4<<12 | 8<<12`

Errori sintattici

Provare queste espressioni ed osservare con cura il tipo di errore!

- `1+>3`
- `1+(2`
- `1e`
- `1e/5`
- `1)` — Quale errore è segnalato?
- `1t`
- `"a`

Provate a commettere altri errori di sintassi, e verificate il comportamento del framework ed il tipo di errore ritornato!

Errori semantici

Provare queste espressioni ed osservare con cura il tipo di errore!

- `1/0`
- `1.0/0` — I numeri in virgola mobile supportano ∞!
- `-1.0/0`
- `1%0`
- `1.0%0` — Cosa succede?
- `2147483648` — Come si più risolvere?
- `1+false`
- `a`
- `new int[]{true}`
- `new int[]{1}[a]`
- `new int[]{1, 2}[10]`
- `new int[]{1, 2}[-1]`
- `new int[]{1, 2}[2]` — Perché dà errore?

Esercizio riassuntivo

Considerando le priorità degli operatori viste a lezione, si risolva su carta e si provi la correttezza tramite il framework FV di:

- `5+7&8 | 9 | 10<<3-~7`
  - In Windows, il carattere "~" si ottiene tenendo premuto Alt e premendo in sequenza 1, 2, 6 nel tastierino numerico.

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