# Informatica Generale

11 febbraio 2004

Cognome ..... Nome .....
Matricola .....

- Usate il retro dei fogli per scrivere lo svolgimento degli esercizi di programmazione.
- Quando vi è richiesto di scrivere un programma, potete limitarvi al *corpo* del metodo main, assumendo se necessario che in e out siano due variabili di classe ConsoleInputManager e ConsoleOutputManager (rispettivamente), già dichiarate e inizializzate.

1. Nei seguenti esercizi, è essenziale riportare l'intero procedimento di soluzione.
   (a) Eseguire la sottrazione $14 - 45$ rappresentando i numeri in *complemento a 2* su 8 bit. Mostrare che il risultato rappresenta effettivamente $-31$.
   (b) Qual'è la rappresentazione di $-45$ in *modulo e segno* su 8 bit?
   (c) Rappresentare in ottale il numero binario 1100110.

(Svolgimento sul retro)

2. Individuare quali delle seguenti espressioni booleane sono delle *contraddizioni*, motivando la risposta ($\wedge$ = AND, $\vee$ = OR, $\oplus$ = XOR = OR ESCLUSIVO, $\neg$ = NOT):

   (a) $(x \wedge y) \wedge \neg(x \vee y)$,

   (b) $\neg x \wedge (y \vee \neg(x \wedge z))$,

   (c) $(\neg x \vee \neg y) \vee (x \oplus y)$,

(Svolgimento sul retro)

3. Scrivete un (frammento di) programma Java che operi come segue:

   - legga 8 interi da tastiera, memorizzandoli in un array $a$,
   - chieda all'utente di inserire due posizioni $i$ e $j$ (assumiamo che $0 \leq i \leq j \leq 7$),
   - inverta i numeri contenuti in $a$ tra le posizioni $i$ e $j$ (comprese), cioè: il numero all'$i$-esima posizione viene scambiato con quello alla $j$ esima posizione, il numero all'$(i+1)$-esima posizione viene scambiato con quello alla $(j-1)$-esima posizione, e così via,
   - stampi l'array risultante.

   Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

```
Numero 0:  3
Numero 1:  12
Numero 2:  44
Numero 3:  23
Numero 4:  -20
Numero 5:  11
Numero 6:  15
Numero 7:  6
Primo estremo:  2
Secondo estremo:  5

3
12
11
-20
23
44
15
6
```

(Svolgimento sul retro)

4. Considerate la classe Frazione i cui oggetti corrispondono a frazioni; questa classe ha un costruttore pubblico

```
public Frazione( int x, int y )
```

che costruisce una frazione di numeratore x e denominatore y, e i metodi

```
public int getNumeratore()
public int getDenominatore()
```

che forniscono, rispettivamente, il numeratore ed il denominatore della frazione.

Scrivete un programma che operi come segue:

(a) chieda all'utente di inserire un numero intero, diciamo $N$;

(b) dichiari un array di $N$ frazioni e lo riempia con frazioni inserite dall'utente;

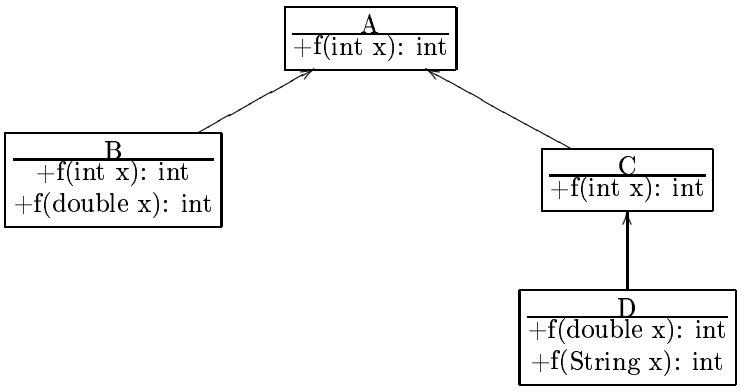
(c) stampi le sole frazioni *proprie*, cioè, le frazioni in cui il valore assoluto del denominatore è maggiore del valore assoluto del numeratore (la classe Frazione dispone di un opportuno metodo toString()).

Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

```
Inserisci un intero:  4

Numeratore:  5
Denominatore:  8

Numeratore:  4
Denominatore:  2

Numeratore:  -4
Denominatore:  2

Numeratore:  -2
Denominatore:  10

5/8
-2/10
```

(Svolgimento sul retro)

5. Considerate la seguente gerarchia di classi (rappresentata mediante un diagramma UML):




e ipotizzate che tutte le classi abbiano un costruttore pubblico senza argomenti. Assumete le seguenti definizioni e inizializzazioni:

```
A a1, a2, a3, a4;
B b;
C c1, c2;
D d;

a1 = new A(); a2 = new B(); a3 = new C(); a4 = new D();
b = new B();
c1 = new C(); c2 = new D();
d = new D();
```

Per ciascuna delle seguenti invocazioni di metodo, dire se essa è consentita e in caso affermativo indicare il nome della classe che contiene il metodo che verrà effettivamente eseguito:

- `b.f(3.4f)`: .....
- `b.f(3)`: .....
- `a1.f(3.0)`: .....
- `c1.f(3+5)`: .....
- `c1.f(3+"5")`: .....

6. Considerate la seguente funzione $f : \{1, 2, \dots\} \to \{1, 2, \dots\}$:

$$f(n) = \begin{cases} 1 & \text{se } n < 3 \\ 2 & \text{se } n = 3 \\ f(n-1) + f(n-2) & \text{altrimenti.} \end{cases}$$

Scrivete un metodo statico che realizzi $f$ (ignorate il problema di controllare che l'argomento passato al metodo sia maggiore di 0):

7. Definite una classe, di nome Studente, i cui oggetti rappresentano studenti universitari. Ogni studente è caratterizzata da nome, cognome e numero di matricola; tutti questi attributi sono di tipo String. La classe deve avere:

   - un costruttore con 3 argomenti (nome, cognome, matr);
   - tre metodi getNome(), getCognome() e getMatr() che restituiscono il nome, il cognome ed il numero di matricola, rispettivamente;
   - un metodo toString();
   - un metodo **public boolean** omonimo( Studente s ) che restituisce `true` se lo studente su cui è invocato è omonimo (per nome e cognome) dello studente che viene passato come argomento, `false` altrimenti.

   La classe Studente dev'essere, inoltre, estesa da una classe Tesista i cui oggetti rappresentano tesisti (studenti in tesi). Ogni tesista, oltre alle caratteristiche di uno studente, deve avere anche il cognome del professore suo relatore. La classe deve avere:

   - un costruttore con 4 argomenti (nome, cognome, matr, rel);
   - un metodo getRel() che restituisce il cognome del relatore;
   - un metodo toString();
   - un metodo **public boolean** inTesiCon( String r ) che restituisce `true` se il tesista su cui è invocato ha come relatore il professore il cui cognome è contenuto in r, `false` altrimenti.

(Svolgimento sul retro)