Informatica Generale

21 aprile 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 $21 - 57$ rappresentando i numeri in *complemento a 2* su 8 bit. Mostrare che il risultato rappresenta effettivamente -36.
   (b) Qual è la rappresentazione di -45 in *complemento a due* su 8 bit?
   (c) Rappresentare in *esadecimale* il numero *ottale* 771.

(Svolgimento sul retro)

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

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

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

   (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 10 interi da tastiera, memorizzandoli in un array a,
   - controlli se i 10 interi sono in ordine crescente (cioè, ogni numero è minore o uguale al successivo) e in tal caso stampi la stringa `Crescente`,
   - in caso contrario, stampi la stringa `Non crescente`.

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

```
Numero 0:  3
Numero 1:  12
Numero 2:  12
Numero 3:  15
Numero 4:  15
Numero 5:  15
Numero 6:  15
Numero 7:  18
Numero 8:  1000
Numero 9:  2500


Crescente
```

(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,

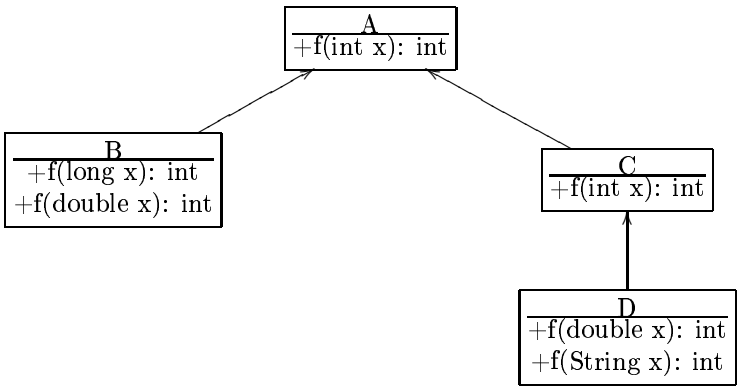
(c) stampi le sole frazioni maggiori di 0.8 (la classe Frazione dispone di un opportuno metodo toString()).

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

```
Inserisci un intero:  4

Numeratore:  5
Denominatore:  8

Numeratore:  5
Denominatore:  2

Numeratore:  -7
Denominatore:  3

Numeratore:  -2
Denominatore:  10

5/8
5/2
```

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

6. Considerate la seguente funzione $f : \{0, 1, 2, \dots\} \to \{0, 1, 2, \dots\}$:

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

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

7. Definite una classe, di nome Paziente, i cui oggetti rappresentano i pazienti ricoverati in una struttura ospedaliera. Ogni paziente è caratterizzato da un nome, un cognome, una data di ammissione, una diagnosi. Tutti questi attributi sono di tipo String, tranne la data di ammissione (che è di tipo java.util.Date). La classe deve avere:

   - un costruttore con quattro argomenti

     (nome, cognome, data di ammissione, diagnosi);

   - quattro metodi

     getNome(), getCognome(), getDataAmmissione(), getDiagnosi()

     che restituiscono il nome, il cognome, la data di ammissione e la disagnosi, rispettivamente;

   - un metodo toString();
   - un metodo **public boolean** stessaMalattia(Paziente p) che restituisce `true` se il paziente su cui è invocato ha la stessa diagnosi di quello che viene passato come argomento, `false` altrimenti.

   La classe Paziente dev'essere, inoltre, estesa da una classe PazieteDeceduto i cui oggetti rappresentano pazienti deceduti. Ognuno di essi, oltre alle caratteristiche di un paziente, deve anche avere un attributo che rappresenta la data di morte e un attributo booleano che indica se l'autopsia è già stata effettuata. La classe deve avere:

   - un costruttore con 5 argomenti

     (nome, cognome, data di ammissione, diagnosi, data di morte)

     che *assume che l'autopsia non sia stata effettuata*;

   - due metodi

     getDataMorte(), autopsiaEffettuata()

     che restituiscono rispettivamente la data di morte e se l'autopsia è stata effettuata oppure no;

   - un metodo toString().

(Svolgimento sul retro)