Informatica Generale

23 giugno 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 $12 - 75$ rappresentando i numeri in *complemento a 2* su 8 bit. Mostrare che il risultato rappresenta effettivamente -63.
   (b) Qual è la rappresentazione di -10 in *complemento a due* su 10 bit?
   (c) Rappresentare in *ottale* il numero *esadecimale* 15.

(Svolgimento sul retro)

2. Per ognuna delle seguenti espressioni booleane, fornite la tabella di verità e indicate in quanti casi esse assumono valore *vero*. Evidenziate inoltre eventuali tautologie e contraddizioni.

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

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

   (c) $(x \vee y \vee \neg z) \wedge \neg z$.

(Svolgimento sul retro)

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

   - legga un intero n da tastiera;
   - legga n interi da tastiera, memorizzandoli in un array a;
   - stampi quegli interi il cui valore è uguale all'indice (es., un valore uguale a 0 inserito nel posto 0, ecc.).

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

```
Quanti interi?  5

Numero 0:  1
Numero 1:  1
Numero 2:  3
Numero 3:  3
Numero 4:  15

1
3
```

(Svolgimento sul retro)

4. Considerate la classe Retta i cui oggetti corrispondono a delle rette sul piano cartesiano; questa classe ha un costruttore pubblico

```
public Retta( double m, double q )
```

che costruisce una retta con equazione $y = mx + q$. La classe ha il metodo pubblico

```
public boolean isPerp( Retta r )
```

che restituisce **true** se la retta su cui è invocato è perpendicolare alla retta r passata per argomento, **false** altrimenti.

Scrivete un programma che operi come segue:

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

(b) dichiari un array di N rette e lo riempia con rette i cui dati (coefficiente angolare m e segmento staccato q) sono inseriti dall'utente;

(c) stampi tutte le rette che *non* sono perpendicolari alla prima retta inserita (la classe Retta dispone di un opportuno metodo toString()).

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

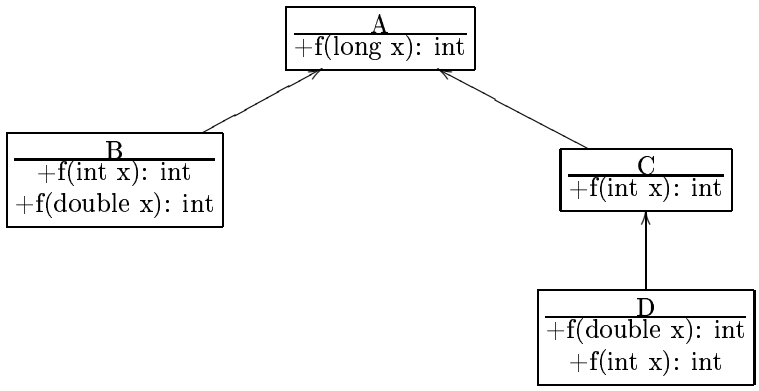
```
Inserisci un intero:  5

Retta #1:  m=4
Retta #1:  q=8
Retta #2:  m=-0.25
Retta #2:  q=6
Retta #3:  m=4
Retta #3:  q=3
Retta #4:  m=-0.25
Retta #4:  q=8
Retta #5:  m=0.12
Retta #5:  q=2

y = 4 * x + 3
y = 0.12 * x + 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 $f : \{0, 1, 2, \dots\} \to \{0, 1, 2, \dots\}$:

$$f(n) = \begin{cases} n - 1 & \text{se } n < 10 \text{ ed } n \text{ è pari} \\ f(n-1) + n & \text{altrimenti.} \end{cases}$$

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 Aereo, i cui oggetti rappresentano i velivoli posseduti da una compagnia aerea. Ogni velivolo è caratterizzato da un *nome*, un *tipo*, una *data* di acquisizione e un *numero* di voli effettuati. Il nome e il tipo sono di tipo String, la data di acquisizione è di tipo java.util.Date, il numero di voli è di tipo intero. La classe deve avere:

   - un costruttore con tre argomenti

     (nome, tipo, data di acquisizione);

     e *deve assumere che il numero di voli effettuati sia zero*;
   - quattro metodi

     getNome(), getTipo(), getDataAcquisizione(), getNumeroVoli()

     che restituiscono i valori dei rispettivi campi;
   - un metodo toString();
   - un metodo **public boolean** daRottamare() che restituisce **true** se il velivolo su cui è invocato ha effettuato più di 2000 voli, *false* altrimenti;
   - un metodo **public boolean** acquistatoPrimaDi(Aereo a) che restituisce **true** se il velivolo su cui è invocato è stato acquistato prima di a, **false** altrimenti (ricordate che la classe Date implementa l'interfaccia Comparable che prevede la presenza di un metodo compareTo con il seguente contratto: x.compareTo(y) ha un valore negativo se $x < y$, positivo se $x > y$).

   La classe Aereo va, inoltre, estesa da una classe AereoIntercontinentale i cui oggetti rappresentano aerei adatti a trasvolate intercontinentali. Un aereo intercontinentale *dev'essere rottamato quando ha effettuato più di 1000 voli*. Scrivete la classe AereoIntercontinentale implementando tutti e soli i metodi e costruttori necessari.

(Svolgimento sul retro)