

Fondamenti di Architettura e Programmazione

3 aprile 2006

Cognome **Babbi** Nome **Babbo**
Matricola 669966

- 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 $10 - 13$ rappresentando i numeri in *complemento a 2* su 7 bit. Mostrare che il risultato rappresenta effettivamente -3 .
 - (b) Qual è la rappresentazione di -5 in *complemento a 2* su 8 bit?
 - (c) La rappresentazione di 134 in base x è 342. Quanto vale x ?
 - (d) Rappresentare in esadecimale il numero ottale 342.

(Svolgimento sul retro)

(a)

10	0001010
-13	1110011
-3	1111101

(b) 11111011 (essendo $4_{10} = 100_2$)

(c) $3x^2 + 4x + 2 = 134$, cioè $3x^2 + 4x - 132 = 0$, che ha come sola soluzione positiva $x = 6$

(d) $342_8 = 011\ 100\ 010_2$ e, raggruppando le cifre 4 a 4, $1110\ 0010_2 = E2_{16}$.

2. Per ognuna delle seguenti espressioni booleane, fornite la tabella di verità, evidenziando eventuali *tautologie e contraddizioni*.

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

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

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

3. Dimostrare la seguente equivalenza logica sia mediante tabelle di verità sia mediante semplificazioni algebriche:

$$\neg x \vee \neg y \vee (\neg x \wedge \neg y) = \neg(x \wedge y)$$

(Svolgimento sul retro)

2.

x	y	z	$\neg x$	$\neg z$	$y \wedge \neg z$	$\neg x \vee (y \wedge \neg z)$
0	0	0	1	1	0	1
0	0	1	1	0	0	1
0	1	0	1	1	1	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	1	0	0	0	0
1	1	0	0	1	1	1
1	1	1	0	0	0	0

(a)

x	y	$x \vee y$	$\neg x$	$x \wedge (x \vee y)$	$(x \wedge (x \vee y)) \vee \neg x$
0	0	0	1	0	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	1	0	1	1

(b)

tautologia.

x	y	z	$\neg x$	$\neg x \vee y$	$\neg(\neg x \vee y)$	$\neg(\neg x \vee y) \wedge z$
0	0	0	1	1	0	0
0	0	1	1	1	0	0
0	1	0	1	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	1	0	0	1	1
1	1	0	0	1	0	0
1	1	1	0	1	0	0

(c)

3.

$$\neg x \vee \neg y \vee (\neg x \wedge \neg y) = \neg(x \wedge y)$$

x	y	$\neg x$	$\neg y$	$\neg x \wedge \neg y$	$\neg x \vee \neg y \vee (\neg x \wedge \neg y)$	$x \wedge y$	$\neg(x \wedge y)$
0	0	1	1	1	1	0	1
0	1	1	0	0	1	0	1
1	0	0	1	0	1	0	1
1	1	0	0	0	0	1	0

$$\neg x \vee \neg y \vee (\neg x \wedge \neg y) = \text{(commutativa)}$$

$$\neg y \vee \neg x \vee (\neg x \wedge \neg y) = \text{(assorbimento)}$$

$$\neg y \vee \neg x = \text{(de Morgan)}$$

$$\neg(\neg \neg y \wedge \neg \neg x) = \text{(doppia negazione)}$$

$$\neg(y \wedge x) = \text{(commutativa)}$$

$$\neg(x \wedge y).$$

4. 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 di dimensione n ;
- stampi i soli interi in A la cui rappresentazione esadecimale inizia con il carattere 7
(ricordiamo che il metodo statico `toString(int x, int b)` della classe `Integer` fornisce la stringa che rappresenta x in base b).

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

```
Quanti interi? 4
```

```
Intero 0: 122
Intero 1: 113
Intero 2: 170
Intero 3: 2679
```

```
122
170
```

(Svolgimento sul retro)

```
int n = in.readInt( "Quanti interi?" );
int a[] = new int[ n ];
for ( int i = 0; i < n; i++ )
    a[ i ] = in.readInt( "Intero " + i + ": " );
for ( int i = 0; i < n; i++ )
    if ( Integer.toString( a[ i ], 16 ).charAt( 0 ) == '7' )
        out.println( a[ i ] );
```

5. Considerate la classe `Cibo` i cui oggetti rappresentano cibi; questa classe ha un costruttore pubblico

```
public Cibo( String nome, int caloriePerEtto )
```

che costruisce un cibo con dato nome e calorie per etto. La classe ha i metodi pubblici

```
public int maxEttiAlGiorno()  
public boolean piuCaloricoDi( Cibo c )
```

il primo restituisce il numero massimo di etti di cibo che possono essere mangiati al giorno; il secondo determina se il cibo è più calorico di quello passato come argomento.

Scrivete un programma che operi come segue:

- (a) chieda all'utente di inserire un numero intero, diciamo N ;
- (b) dichiari un array di N cibi e lo riempia con cibi chieste all'utente;
- (c) stampi tutti i cibi di cui si possano mangiare più di 4 etti al giorno (per la stampa, assumete che la classe `Cibo` abbia un metodo `toString()` opportuno).

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

```
Inserisci un intero: 4
```

```
Cibo #0, Nome = Crescenza
```

```
Cibo #0, KCal/h = 320
```

```
Cibo #1, Nome = Burro
```

```
Cibo #1, KCal/h = 716
```

```
Cibo #2, Nome = Crostata
```

```
Cibo #2, KCal/h = 505
```

```
Cibo #3, Nome = Vino
```

```
Cibo #3, KCal/h = 145
```

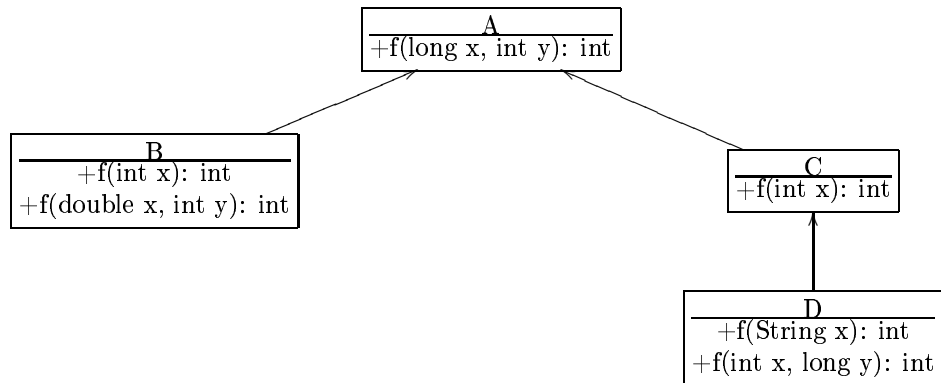
```
Crescenza (320 KCal/h)
```

```
Vino (145 KCal/h)
```

(Svolgimento sul retro)

```
int n = in.readInt( "Inserisci un intero: " );
Cibo a[] = new Cibo[ n ];
for ( int i = 0; i < n; i++ ) {
    String nome = in.readLine( "Cibo #" + i + ", Nome=" );
    int cal = in.readInt( "Cibo #" + i + ", KCal/h=" );
    a[ i ] = new Cibo( nome, cal );
}
for ( int i = 0; i < n; i++ )
    if ( a[ i ].maxEttiAlGiorno() > 4 )
        out.println( a[ i ] );
```

6. 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;
B b;
C c1, c2;
D d;
  
```

```

a1 = new A(); a2 = new B();
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:

- **a1.f(3,4): A: f(long,int)**
- **b.f(3+3L): No**
- **c1.f(3L,3): A: f(long,int)**
- **a2.f(3+"ciao"): No**
- **d.f(3,4L): D: f(int,long)**
- **c2.f(3+"ciao"): No**

7. Scrivete il codice di una classe di nome `Professore` le cui istanze rappresentano i docenti di una scuola media superiore. Ogni professore è caratterizzato dal *nome*, da una *materia* insegnata e da un *numero di ore* di insegnamento settimanali: il nome dovrà essere una stringa, il numero di ore un intero mentre la materia sarà istanza di un'opportuna classe `Materia` che potete assumere preesistente.

Dovete fornire i seguenti costruttori e metodi pubblici:

- `Professore(String nome, Materia mat, int ore)`: crea un nuovo professore con i dati specificati;
- `int oreAlMese(int sett)`: restituisce il numero di ore che il professore insegna in un mese, dato il numero di settimane in quel mese;
- `boolean stessaMateria(Professore p)`: restituisce true se il professore insegna la stessa materia di p;
- `int incrementaOre(int v)`: incrementa il numero di ore insegnate settimanalmente del valore v;

La classe `Professore` va, inoltre, *estesa* da una classe `ProfessoreDiLaboratorio` i cui oggetti rappresentano gli insegnanti di laboratorio: questi ultimi sono caratterizzati, oltre che dai dati che caratterizzano tutti i professori, anche dal nome del laboratorio di cui sono responsabili.

Oltre al costruttore, in tale classe va scritto il metodo `getLab()` che restituisce il nome del laboratorio di cui il professore è responsabile.

(Svolgimento sul retro)

```

class Professore {
    private String nome;
    private Materia mat;
    private int ore;

    public Professore(String n, Materia m, int o) {
        nome = n;
        mat = m;
        ore = o;
    }

    public int oreAlMese(int sett) {
        return ore * sett;
    }

    public boolean stessaMateria(Professore p) {
        return mat.equals(p.mat);
    }

    public void incrementaOre(int v) {
        ore += v;
    }
}

class ProfessoreDiLaboratorio extends Professore {
    private String lab;

    public ProfessoreDiLaboratorio(String n,
        Materia m, int o, String lab) {
        super(n, m, o);
        this.lab = lab;
    }

    public String getLab() {
        return lab;
    }
}

```