Informatica Generale

14 febbraio 2005

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

- 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. Per ciascuna delle seguenti espressione booleane, dite su quante righe della tabella di verità il valore dell'espressione è *vero* ($\wedge$ = AND, $\vee$ = OR, $\oplus$ = XOR = OR ESCLUSIVO, $\neg$ = NOT):

   (a) $x \vee (\neg x \wedge (\neg x \wedge y))$: **3**

   (b) $x \wedge (y \vee \neg(x \vee y))$: **1**

   (c) $(\neg x \vee \neg y) \vee (z \oplus y)$: **7**

(Usate il retro per le tabelle di verità)

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

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

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

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

   - legga 10 stringhe da tastiera, memorizzandole in un array a,
   - stampi le sole stringhe la cui lunghezza è pari, ciascuna preceduta dalla sua lunghezza.

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

```
Stringa 0: Sempre
Stringa 1: caro
Stringa 2: mi
Stringa 3: fu
Stringa 4: quest
Stringa 5: ermo
Stringa 6: colle
Stringa 7: e
Stringa 8: questa
Stringa 9: siepe


6 Sempre
4 caro
2 mi
2 fu
4 ermo
6 questa
```

(Svolgimento sul retro)

```
String a[] = new String[ 10 ];
for ( int i = 0; i < 10; i++ )
   a[ i ] = in.readLine( "Stringa " + i + ": " );
for ( int i = 0; i < 10; i++ )
   if ( a[ i ].length() % 2 == 0 )
      out.println( a[ i ].length() + " " + a[ i ] );
```

3. Considerate la classe `Esame` i cui oggetti corrispondono a esami universitari; questa classe ha un costruttore pubblico

```
public Esame(String nome, int cfu)
```

che costruisce un esame con dato nome e numero di CFU (crediti), e i metodi

```
public int getCfu()
public String getNome()
```

che forniscono, rispettivamente, il numero di cfu e il nome dell'esame.

Scrivete un programma che operi come segue:

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

(b) chieda all'utente un altro numero intero, diciamo s;

(c) dichiari un array di n esami e lo riempia con esami inseriti dall'utente;

(d) stampi i nomi dei soli esami con almeno s CFU.

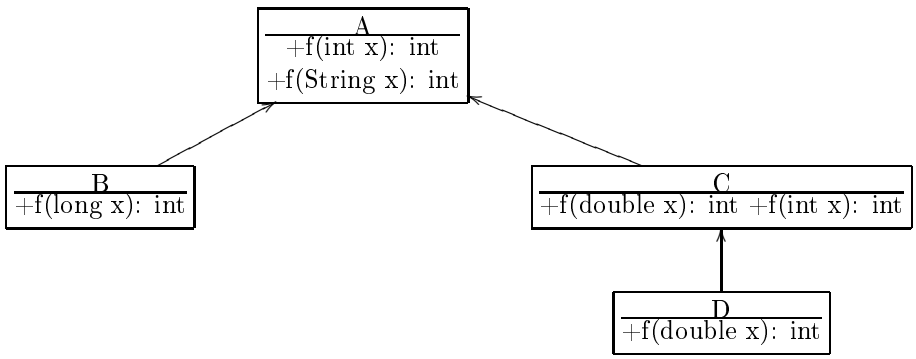
Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

```
Inserisci un intero:  4
Inserisci un altro intero:  5

Nome:  Informatica generale
Cfu:  12

Nome:  Taglio e cucito
Cfu:  3

Nome:  Barba e capelli
Cfu:  5

Nome:  Scaccolamento (corso di base)
Cfu:  1

Informatica generale
Barba e capelli
```

(Svolgimento sul retro)

```
int n = in.readInt( "Inserisci un intero: " );
int s = in.readInt( "Inserisci un altro intero: " );
Esame esame[] = new Esame[ n ];
for ( int i = 0; i < n; i++ ) {
   String nome = in.readLine( "Nome: " );
   int cfu = in.readInt( "Cfu: " );
   esame[ i ] = new Esame( nome, cfu );
}
for ( int i = 0; i < n; i++ )
   if ( esame[ i ].getCfu() >= s )
      out.println( esame[ i ].getNome();
```

4. 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 e la segnatura del metodo stesso:

- `b.f(3)`: **A f(int)**
- `b.f(3L)`: **B f(long)**
- `a1.f(3.0)`: **non compila**
- `b.f(3.0)`: **non compila**
- `c1.f(3+5)`: **C f(int)**
- `d.f(3+5+"")`: **A f(String)**

5. Assumete che `x` e `y` siano variabili `int`, e che il loro valore *prima di ciascuno dei seguenti assegnamenti* sia rispettivamente 1 e 2. Dite quale sarà il loro valore *dopo* gli assegnamenti indicati:

Assegnamento	`x`	`y`
`x+=y+1`	**4**	**2**
`x=++y`	**3**	**3**
`x=(y=x)+1`	**2**	**1**
`x=(x==1)? (x++==1? x:y) : x`	**2**	**2**
`x=y/3+y/2`	**1**	**2**
`x-=y`	**-1**	**2**

6. Scrivete il codice di una classe di nome `Libro` le cui istanze rappresentano i libri disponibili in una biblioteca. Ogni libro è caratterizzato da un titolo (una stringa), un autore (un oggetto istanza di una classe `Autore` che supponete sia già stata scritta), un numero di inventario (intero), un numero di copie (anch'esso intero) e un numero di copie attualmente in prestito.

   Dovete fornire i seguenti costruttori e metodi pubblici:

   - `Libro(String tit, Autore aut, int nInv, int copie)`: crea un libro con i dati forniti (assume che non ci siano copie in prestito);
   - `Libro(String tit, Autore aut, int nInv)`: come sopra, assumendo che ci sia 1 copia;
   - `void concediPrestito()`: concede una copia in prestito, e solleva una `IllegalStateExcepion` se tutte le copie sono in prestito;
   - `int copieDisponibili()`: restituisce il numero di copie disponibili (numero di copie meno numero di copie attualmente in prestito);
   - `boolean isDisponibile()`: restituisce true sse esiste almeno una copia disponibile;
   - `void copiaRestituita()`: viene invocato quando un utente restituisce una copia del libro.

(Svolgimento sul retro)

```
class Libro {
   String titolo;
   Autore autore;
   int nInv;
   int copie, prestate;

   public Libro(String tit, Autore aut, int nInv, int copie) {
      this.tit = tit;
      this.aut = aut;
      this.nInv = nInv;
      this.copie = copie;
      this.prestate = 0;
   }

   public Libro(String tit, Autore aut, int nInv) {
      this(tit, aut, nInv, 1 );
   }

   public void concediPrestito() {
      if ( copie == prestate )
         throw new IllegalStateException( "Tutte le copie sono in prestito!" );
      prestate++;
   }

   public int copieDisponibili() {
      return copie - prestate;
   }

   public boolean isDisponibile() {
      return copieDisponibili() > 0;
   }

   public void copiaRestituita() {
      prestate--;
   }
}
```