

Informatica Generale — II Compitino

26 gennaio 2005

Nota. Ove necessario, nel seguito, assumeremo che `tastiera` e `video` siano due variabili di classe `ConsoleInputManager` e `ConsoleOutputManager` (rispettivamente), già dichiarate e inizializzate.

Rispondete alle seguenti domande.

1. 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 gli interi che terminano con la cifra “7”.

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

Quanti interi? **5**

Numero 0: **17**
Numero 1: **71**
Numero 2: **12**
Numero 3: **777**
Numero 4: **7**

17
777
7

(Svolgimento sul retro)

```
int n;  
n = tastiera.readInt( "Quanti interi? " );  
int a[] = new int[ n ];  
for ( int i = 0; i < n; i++ )  
    a[ i ] = tastiera.readInt( "Numero " + i + ": " );  
for ( int i = 0; i < n; i++ )  
    if ( a[ i ] % 10 == 7 )  
        video.println( a[ i ] );
```

2. Considerate la classe `Corso` i cui oggetti corrispondono a *corsi universitari*; questa classe ha un costruttore pubblico

- `public Corso(String nome, String docente, int cfu)`

che costruisce un corso universitario denominato `nome`, il cui docente titolare ha cognome `docente` e con numero di crediti `cfu`. La classe possiede i metodi

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

che forniscono, rispettivamente, il nome del corso, il cognome del docente titolare ed il numero di crediti.

Scrivete un programma che operi come segue:

- (a) chieda all'utente di inserire un numero intero, diciamo N ;
- (b) dichiari un array di N corsi e lo riempia con corsi inseriti dall'utente;
- (c) stampi i soli corsi tenuti da "Rossi" con numero di crediti maggiori o uguali a 6 (la classe `Corso` dispone di un opportuno metodo `toString()`).

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

Inserisci un intero: **4**

Nome: **Informatica Generale**
Docente: **Rossi**
Cfu: **12**

Nome: **Matematica Discreta**
Docente: **Verdi**
Cfu: **6**

Nome: **Interazione Uomo-Macchina**
Docente: **Rossi**
Cfu: **6**

Nome: **Metodi per il Trattamento dell'Informazione**
Docente: **Rossi**
Cfu: **5**

Corso: Informatica Generale; Tenuto da: Rossi; Cfu: 12

Corso: Interazione Uomo-Macchina; Tenuto da: Rossi; Cfu: 6

(Svolgimento sul retro)

```
int n;
n = tastiera.readInt( "Inserisci un intero: " );
Corso c[] = new Corso[ n ];
for ( int i = 0; i < n; i++ ) {
    String nome = tastiera.readLine( "Nome: " );
    String docente = tastiera.readLine( "Docente: " );
    int cfu = tastiera.readInt( "Cfu: " );
    c[ i ] = new Corso( nome, docente, cfu ); }
for ( int i = 0; i < n; i++ )
    if ( c[i].getCfu() >= 6 && c[i].getDocente().equals("Rossi" ) )
        video.println( c[ i ] );
```

3. Un prezzo in euro è una coppia di interi, in cui il primo rappresenta il numero di euro (e deve essere maggiore o uguale a zero) e il secondo il numero di centesimi di euro (e deve essere compreso fra 0 e 99). Le istanze della seguente classe rappresentano prezzi, usando come unico attributo il numero complessivo di centesimi (ad esempio, il prezzo 7,52 corrisponde a 752 centesimi). Completatene il codice riempiendo gli spazi.

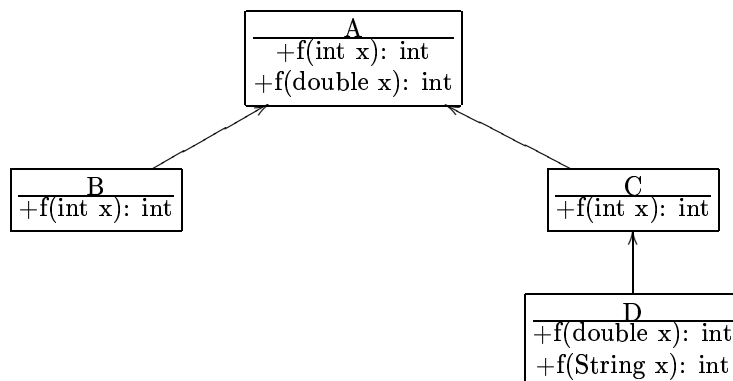
```
class Prezzo {  
    private int centesimiTotali;  
  
    public Prezzo(int euro, int centesimi) {  
  
        this.centesimiTotali = euro * 100 + centesimi;  
  
    }  
}
```

```
// Restituisce il numero di euro  
public int getEuro() {  
  
    return this.centesimiTotali / 100;  
  
}
```

```
// Restituisce il numero di centesimi di euro  
public int getCentesimi() {  
  
    return this.centesimiTotali % 100;  
  
}
```

```
// Restituisce true sse questo prezzo è maggiore di p  
public boolean isMaggiore(Prezzo p) {  
  
    return this.centesimiTotali > p.centesimiTotali;  
  
}  
}
```

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 aa = new A(), ab = new B(), ac = new C(), ad = new D();
B bb = new B();
C cc = new C(), cd = new D();
D dd = new D();
```

Per ciascuno dei seguenti assegnamenti, dite se l'assegnamento è consentito, oppure se richiede (per poter essere compilato) un operatore di cast nel secondo membro (e in tal caso indicate l'assegnamento corretto), oppure se l'assegnamento non è consentito nemmeno con un cast (nel senso che anche con un cast produrrebbe un errore di compilazione o un'eccezione in esecuzione):

- `aa=bb`: **Ok**
- `bb=aa`: **No**
- `bb=ab`: `bb=(B)ab`
- `bb=dd`: **No**
- `ac=cc`: **Ok**

5. Facendo riferimento al precedente diagramma UML e alle variabili definite e inizializzate come sopra, considerate le seguenti invocazioni di metodo. Per ciascuna di esse dovete dire se è consentita e in caso affermativo dovete indicare il nome della classe che contiene il metodo che verrà effettivamente eseguito:

- `bb.f(3.4)`: **A: f(double)**

- `ab.f(3): B: f(int)`
- `ad.f(5): C: f(int)`
- `ad.f("pippo"): No`
- `dd.f("pippo"): D: f(String)`
- `cd.f(2): C: f(int)`
- `dd.f(3/4): C: f(int)`

6. Scrivete una classe di nome `Prodotto` le cui istanze rappresentano i prodotti in vendita in un supermercato. Ogni prodotto è caratterizzato da un nome (p.es. “Pannoloni Morbidoni Fluffy”), un prezzo unitario espresso da un `double` (p.es. 3.12), e un numero intero che rappresenta quantità disponibile (p.es. 1500).

La classe deve contenere i seguenti metodi e costruttori:

- `Prodotto(String nome, double prezzo, int disp)`: crea un prodotto con dato nome, prezzo unitario e quantità disponibile;
- `Prodotto(String nome, double prezzo)`: come sopra, ma assume che la quantità disponibile sia zero;
- `void acquista(int qta)`: aumenta la quantità disponibile del valore `qta`;
- `void vendi(int qta) throws IllegalStateException`: riduce la quantità disponibile del valore `qta`, e solleva un’eccezione se la quantità disponibile è inferiore (l’eccezione `IllegalStateException` è un’eccezione già esistente, con gli usuali costruttori);
- `double prezzoTotale(int qta)`: restituisce il prezzo totale relativo alla quantità `qta`;
- `boolean scorteFinite()`: restituisce `true` se le scorte di questa merce sono terminate (cioè, se la quantità disponibile è zero);
- `String toString()`: restituisce una stringa descrittiva.

(Svolgimento sul retro)

```

class Prodotto {
    private String nome;
    private double prezzoUnitario;
    private int disp;

    Prodotto( String nome, double prezzo, int disp ) {
        this.nome = nome;
        this.prezzoUnitario = prezzo;
        this.disp = disp;
    }

    Prodotto( String nome, double prezzo ) {
        this( nome, prezzo, 0 );
    }

    public void acquista( int qta ) {
        disp += qta;
    }

    public void vendi( int qta ) throws IllegalStateException {
        if ( disp < qta )
            throw new IllegalStateException();
        disp -= qta;
    }

    public double prezzoTotale( int qta ) {
        return prezzoUnitario * qta;
    }

    public boolean scorteFinite() {
        return disp == 0;
    }

    public String toString() {
        return nome + “ (” + prezzoUnitario + “, ” + disp + “)”;
    }
}

```