

Fondamenti di informatica LB

Esame del 2 Settembre 2010

NOTE:

- Scrivere su ogni foglio utilizzato cognome, nome, numero di matricola (tutto in forma leggibile).
- Consegnare tutti i fogli ricevuti (anche il testo).
- Durante la prova non è consentita la consultazione di alcun tipo di materiale, né è consentito uscire.
- La prova è ritenuta sufficiente se almeno una parte significativa di ogni esercizio proposto è svolta correttamente. Il voto finale sarà espresso in ventisettesimi.
- Il tempo a disposizione per il compito è 2h 30'. Coloro i quali consegnano perdono il voto eventualmente conseguito in precedenza.

Esercizio 1 (12/40)

Si vuole realizzare un programma per gestire un negozio per vendite *online* di strumenti musicali. Il progetto deve essere strutturato come segue:

- Vi è una classe astratta **Item** che rappresenta un prodotto disponibile nel negozio.
- Le classi concrete **Guitar** e **Bass** estendono la classe **Item**.
- Una classe **Shop** che gestisce la vendita.

La classe **Item** ha i seguenti campi e metodi:

- **String model**: campo **protected** che rappresenta il modello dello strumento.
- **double price**: campo **protected** che rappresenta il prezzo dello strumento.
- **String getModel()**: metodo pubblico che restituisce il modello dello strumento.
- **double getPrice()**: metodo pubblico che restituisce il prezzo dello strumento.

La classe **Guitar** ha i seguenti campi e metodi:

- Campo privato **String type** che rappresenta il tipo della chitarra (per esempio “Steel-string acoustic guitar”)
- Costruttore, pubblico, con tre argomenti: due di classe **String** rappresentanti il modello e il tipo della chitarra e uno di tipo **double** rappresentante il prezzo (in Euro) della chitarra. Il costruttore deve lanciare un’eccezione **NegativePriceException** (da realizzare) nel caso si tenti di inizializzare uno strumento con un prezzo minore di 0.
- Ridefinire il metodo **toString()** in modo da restituire una stringa che riporti tutte le informazioni sullo strumento.

La classe **Bass** ha i seguenti metodi pubblici:

- Campo privato **int nStrings** che rappresenta il numero di corde del basso.
- Costruttore con tre argomenti: uno di classe **String** rappresentante il modello del basso, uno intero per il numero di corde e uno *double* per il prezzo. Analogamente al costruttore di **Guitar**, deve lanciare un’eccezione **NegativePriceException** nel caso si tenti di inizializzare uno strumento con un prezzo minore di 0.

La classe **Shop** **aggrega** un dato numero di oggetti di classe **Item** (non ha responsabilità di creazione degli oggetti) e li mantiene in una collezione realizzata come **ArrayList** di **Item**.

La classe **Shop** ha i seguenti metodi pubblici:

- Costruttore con un argomento intero che permetta di inizializzare un campo privato intero **maxItems** rappresentante il numero massimo di strumenti che il negozio può contenere.
- **boolean isAvailable(String model)**: restituisce **true** se nel negozio è presente uno strumento del modello indicato.
- **boolean addItem(Item in)**: aggiunge uno strumento **in** nel listino del negozio. Se il numero di strumenti presenti nel listino è inferiore a **maxItems**, **in** viene aggiunto alla lista e il metodo restituisce **true**. Se invece il numero massimo di strumenti è stato superato, **in** non viene aggiunto alla lista e il metodo restituisce **false**.
- **Item buy(String model, double money)**: permette di acquistare uno strumento di modello specificato, inserendo un quantità di denaro pari a **money**. Nel caso l’acquisto sia possibile (modello esistente e denaro sufficiente), il metodo restituisce il prodotto acquistato, aggiornando opportunamente la lista dei prodotti presenti nel negozio. Altrimenti restituisce **null**.

Si implementino le classi così come descritte e si implementi anche un programma di test. I costruttori delle classi `Guitar` e `Bass` possono anche essere realizzati estendendo un costruttore per `Item` che effettua le operazioni comuni. Inoltre, anche per l'implementazione degli altri metodi, è possibile servirsi di metodi ausiliari da realizzare come metodi privati. Si riporta stralcio della documentazione di interesse:

```
boolean add(E e): Appends the specified element to the end of this list.
```

```
boolean remove(Object o): Removes the first occurrence of the  
                           specified element from this list,  
                           if it is present.
```

```
int size(): Returns the number of elements in this list.
```

Esercizio 2 (10/40)

Si analizzi il codice seguente e si riporti l'output prodotto dall'esecuzione del programma. Le risposte devono essere opportunamente commentate e motivate.

```
abstract class Alpha {
    public Alpha() { System.out.println("Alpha"); }
    public void foo(Alpha a) { System.out.println("Euler"); }
}

class Beta extends Alpha {
    public Beta(){
        super();
        System.out.println("Beta");
    }
    public void foo(Beta a) { System.out.println("Gauss"); }
}

class Gamma extends Beta {
    public Gamma(){
        super();
        System.out.println("Gamma");
    }
    public void foo(Alpha a) { System.out.println("Riemann"); }
}

public class Esercizio2 {
    public static void main(String [] args){
        Beta b = new Beta();
        Gamma g = new Gamma();
        b.foo(g);
        b.foo((Beta)g);
        b.foo((Alpha)g);
        g.foo((Alpha)b);
    }
}
```

Esercizio 3 (8/40)

Si completi l'implementazione sotto riportata del pannello di una semplice applicazione la cui interfaccia grafica è costituita da tre campi di testo e un gruppo di due bottoni radio.

Quando il cursore è sul campo `txt1`, alla pressione del tasto ENTER si deve scrivere nel campo `dest` la concatenazione del contenuto di `txt1` e `txt2`. Inoltre, quando è selezionato il bottone radio *Active* (risp. *Not active*) il campo di testo `txt2` deve essere reso modificabile (risp. non modificabile). Si ottenga questo comportamento, gestendo gli eventi generati dai bottoni radio (eventi di tipo *ActionEvent*);

```
class MyPanel extends JPanel implements ActionListener{
    private JTextField txt1,txt2,dest;
    private ButtonGroup bg;
    private JRadioButton [] rb;

    public MyPanel() {
        super();
        txt1 = new JTextField(30);
        txt2 = new JTextField(30);
        dest = new JTextField(30);
        dest.setEditable(false);
        bg = new ButtonGroup();
        rb = new JRadioButton[2];
        rb[0] = new JRadioButton("Active");
        rb[1] = new JRadioButton("Not active");
        rb[0].setSelected(true);
        ...[1]...
    }

    public void actionPerformed(ActionEvent e) {
        ...[2]...
    }
}
```

Documentazione di interesse:

```
public Object getSource():
    The object on which the Event initially occurred.
```

Esercizio 4 (10/40)

Si illustrino sinteticamente le proprietà principali dell'ereditarietà tra classi.