

Fondamenti di informatica LB

Esame del 19 Luglio 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 progetti la parte principale di un'applicazione che realizza un'agenda per esami in un mese. L'esame generico è definito dalla classe astratta **Exam** così strutturata:

- Campi **String** `courseName`, **int** `time` e **int** `day` che rappresentano, rispettivamente, il nome dell'insegnamento, l'orario di inizio e il giorno del mese in cui si svolge l'esame.
- Costruttore con tre argomenti, rispettivamente per inizializzare i campi `courseName`, `time` e `day`. Qualora si provi a creare un esame con orario di inizio minore di 9 o maggiore di 18, deve essere generata un'eccezione.
- Metodo pubblico **int** `getDay()` che restituisce il giorno del mese in cui si svolge l'esame.

La classe **Exam** è estesa dalle classi **WrittenExam** e **OralExam**. La classe **OralExam** ha soltanto un costruttore per impostare i campi `courseName`, `time` e `day`. Invece, la classe **WrittenExam** ha anche un campo **int** `duration` che rappresenta la durata dell'esame. Tale campo dovrà essere inizializzato dal costruttore, che ha quattro parametri. Qualora la durata sia minore di 0 oppure la fine dell'esame (`time + duration`) sia maggiore di 18 deve essere generata un'eccezione.

L'agenda è realizzata dalla classe **Scheduler** così strutturata:

- Campo **int** `numberOfDays` che rappresenta il numero dei giorni del mese (da impostarsi tramite il costruttore).

- `ArrayList<Exam> exams`: campo che contiene la lista degli esami.
- Costruttore di default che inizializza `exams` e `numberOfDays`.
- `boolean add(Exam ex)`: metodo che permette di aggiungere un esame. Il metodo restituisce `false` se il giorno indicato nell'esame è minore di 1 o maggiore di `numberOfDays`, oppure se nel giorno si svolge già un altro esame della lista `exams`. In ogni altro caso aggiunge l'esame alla lista e restituisce `true`.

Si implementino le classi così come descritte e si implementi anche un programma che mostri come utilizzare le classi e i metodi implementati. Definire opportunamente campi e metodi come *private*, *protected* o *public*, se non specificato nel testo. Si riporta uno stralcio della documentazione di interesse:

```
java.util
Class ArrayList<E>
boolean add(E o): Appends the specified element to the end of 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.

```
class White {
    public void repaint(Brown g) {
        g.paint();
        System.out.println("Repainted in white");
    }
}

class Blue extends White {
    public void repaint(Green d) {
        d.paint();
        System.out.println("Repainted in blue");
    }
}

class Brown {
    public void paint() {System.out.println("Paint in brown");}
}

class Green extends Brown {
    public void paint() {
        super.paint();
        System.out.println("Paint in green");
    }
}

public class Esercizio2 {
    public static void main(String [] args){
        Blue obj1 = new Blue();
        Brown obj2 = new Green();
        obj1.repaint((Green)obj2);
        obj1.repaint(obj2);
    }
}
```

Esercizio 3 (8/40)

Si completi l'implementazione sotto riportata del pannello di una semplice applicazione la cui interfaccia grafica è costituita da due campi di testo `txt1` modificabile e `txt2` non modificabile e un bottone. Ogni **due pressioni del bottone**, nel campo `txt2` deve essere riportato ciò che è contenuto in `txt1`.

```
class MyPanel extends JPanel
    implements ActionListener {
    private JTextField txt1,txt2;
    private JButton b;
    private boolean act;

    public MyPanel() {
        super();
        txt1 = new JTextField(30);
        txt2 = new JTextField(30);
        ...[1]...
    }

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

Documentazione di interesse:

```
class JTextField
public String getText(): Returns the text contained in this TextComponent.
public void setText(String t): Sets the text of this TextComponent
                             to the specified text.
```

Esercizio 4 (10/40)

Si definisca il polimorfismo relativamente al *subclassing* e se ne fornisca un esempio.