

Fondamenti di informatica B (9 CFU)

Esame del 7 Gennaio 2011

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 progettare una parte di un sistema software per il controllo di robot. Un robot è rappresentato dalla classe `Robot` che ha i seguenti campi e metodi:

- `ArrayList<Sensors> sensors`: lista dei sensori di cui dispone il robot.
- `ArrayList<Actuators> actuators`: lista degli attuatori di cui dispone il robot.
- `Robot()`: costruttore di default.
- `void addSensor(Sensor s)`: aggiunge un sensore al robot (aggiungendolo alla lista).
- `void addActuator(Actuator s)`: aggiunge un attuttore al robot (aggiungendolo alla lista).
- `double sense(int i)`: metodo che legge il valore del sensore in posizione *i*. Se il valore di *i* non è ammissibile, allora il metodo lancia un'eccezione.
- `boolean act(int i, double v)`: metodo che imposta il valore *v* sull'attuatore in posizione *i*. Restituisce `true` se l'operazione va a buon fine e *i* è ammissibile, altrimenti restituisce `false`.

Le classi `Sensor` e `Actuator` sono classi astratte caratterizzate, rispettivamente, dal metodo astratto `double getValue()` che restituisce il valore letto dal sensore e `void setValue(double v)` che imposta il valore per l'attuatore (il metodo deve prevedere la possibilità che sia lanciata un'eccezione). Entrambe le classi hanno un campo che rappresenta il relativo stato (cioè il valore letto o impostato). Definire opportunamente campi e metodi come *private*, *protected* o *public* e un costruttore di default. La classe `Sensor` è estesa dalla classe

LightSensor. Per semplicità, si implementi il metodo `getValue()` assegnando allo stato un valore casuale $[0, 1]$ e restituendolo. La classe **Actuator** è estesa dalla classe **Wheel**. Il metodo `setValue(double v)` accetta un valore compreso nell'intervallo $[0, 1]$. In caso di valore esterno all'intervallo, il metodo lancia un'eccezione.

Si implementino le classi così come descritte (definendo opportunamente campi e metodi come *private*, *protected* o *public*) e si implementi anche un programma che mostri come utilizzare le classi e i metodi implementati. 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.
public E get(int index): Returns the element at the specified position
                        in this list.
```

```
java.lang
class Math
static double random(): Returns a double value with a positive sign,
                        greater than or equal to 0.0 and less than 1.0.
```

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 Alpha {
    private Rho r;

    public Alpha(Rho r) {
        this.r = r;
        System.out.println("Alpha," + r.toString());
    }
}

class Rho {
    protected int n;
    public Rho(){
        n = 1;
        System.out.println("Rho");
    }
    public String toString(){
        return "Rho:" + n;
    }
}

class Omicron extends Rho {
    public Omicron() {
        super();
        System.out.println("Omicron");
        n++;
    }
    public String toString(){
        return "Omicron:" + n;
    }
}

public class Esercizio2 {
    public static void main(String [] args){
        Omicron om = new Omicron();
        Alpha aa = new Alpha(om);
    }
}
```

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, un pulsante JButton e una casella JCheckBox. Si devono gestire due eventi diversi:

- Quando il cursore è sul campo `txt1` oppure sul campo `txt2`, alla pressione del tasto ENTER (della tastiera) si deve scrivere nel campo `dest` la concatenazione del contenuto di `txt1` e `txt2`.
- Quando si preme sul pulsante, lo stato modificabile (risp. non modificabile) del campo di testo `txt2` deve essere cambiato, **ma solo se la checkbox è selezionata**.

```
class MyPanel extends JPanel implements ActionListener{
private JTextField txt1,txt2,dest;
private JCheckBox ck;
public MyPanel() {
    super();
    txt1 = new JTextField(30); txt2 = new JTextField(30);
    dest = new JTextField(30); dest.setEditable(false);
    ck = new JCheckBox("On/Off");
    JButton b = new JButton("Ok");
    ...[1]...
}
public void actionPerformed(ActionEvent e) {
    ...[2]...
}
}
```

Documentazione di interesse:

```
public Object getSource():
```

The object on which the Event initially occurred.

```
public boolean isEditable():
```

Returns the boolean indicating whether this TextComponent is editable or not.

```
public boolean isSelected():
```

Returns the state of the button. True if the toggle button is selected, false if it's not.

Esercizio 4 (10/40)

a) Analizzare il seguente programma e indicare quali sono i valori stampati a tempo di esecuzione. Si motivi opportunamente la risposta.

```
#include <stdio.h>
#include <stdlib.h>

#define N 6

void mask(char *vec, int *bitstring, int len);

int k = 1;

int main()
{
    int *bitstring;
    int i, len;
    char mystring[N] = {'a', 'b', 'c', 'd', 'e', '\0'};

    bitstring = (int *) malloc(sizeof(int) * N);
    for (i = 0; i < N; i++) bitstring[i] = 1 - (i % 2);

    len = N;
    mask(mystring, bitstring, len);

    printf("%s\n", mystring);
    printf("k = %d\n", k);
    return 0;
}

void mask(char *vec, int *bitstring, int len)
{
    int i;
    i = k = 0;
    while (i < len-1){
        if (bitstring[len-i-1] != 0){
            vec[i] = 't';
            k++;
        }
        i++;
    }
}
```

b) Data la funzione:

```
int funzione(int x, int y)
{
    if (x - y <= 1) return x * y;
    else return funzione(x-1,y);
}
```

si scriva il risultato della funzione quando invocata come `funzione(4,2)` e si disegnino i corrispondenti record di attivazione. Indicare anche se la funzione è *tail ricorsiva* e motivare la risposta.