Programmazione

Appello del 22/12/2010

Esercizio 1 (13 punti)

Scrivere l'output del seguente programma Java

```
public class Main {

public static void main(String[] args) {
    enigma(2);
    enigma(6);
    enigma(49);
    enigma(800);
    enigma(1024);
}

public static void enigma (int n){
        int i=2;
        while (i*i <= n){
            if (n%i==0){
                System.out.print(i + " ");
                n /= i;
            }
            else {
                i++;
            }
        }
        System.out.println(n);
    }
}
```

Esercizio 2 (5 punti)

L'implicazione logica ha la seguente tabella di verità:

b1\b2	True	False
True	True	False
False	True	True

Scrivere un metodo di classe ***implica*** che prende in input due booleani b1 e b2, e restituisce in output il booleano corrispondente a b1 implica b2. **Il metodo deve contenere unicamente il comando return.**

Esercizio 3 (13 punti)

Scrivere un metodo di classe **Unione** che presi in input due Array A e B del tipo generico T, restituisce in output un Vector<T> contenente l'unione degli elementi contenuti in A e in B, con l'accortezza che gli elementi non devono essere ripetuti nel Vector di output.

[per aggiungere elementi ad un Vector v si può usare il metodo v.Add(Object o)]

Esercizio alternativo (dà luogo ad una valutazione minore): *considerare il tipo int (ed Integer per il Vector) invece del tipo generico T.*

INIZIO SECONDO PARZIALE

Esercizio 4 (13 punti)

- Scrivere in Java una classe Memoria che simuli la memoria di un calcolatore, organizzata come un array di B byte (B è un **int** preso in input dal ***costruttore*** della classe che deve essere l'unico presente). Impostare tutti i campi nella classe in modo opportuno in modo che non sia possibile manipolare la memoria dall'esterno se non attraverso i metodi che seguono.
- Aggiungere a Memoria un metodo **public boolean *set****(int posizione, byte valore)* che memorizza il valore dato nella data posizione della memoria, se questa posizione è libera. Il metodo torna true se il valore è stato memorizzato, false altrimenti.
- Aggiungere a Memoria un metodo **public byte *get****(int posizione)* che restituisce il valore memorizzato nella data posizione di memoria, qualore questa posizione sia presente, altrimenti solleva un'opportuna eccezione **nullPointerException**.
- Aggiungere a Memoria un metodo **public void *reset****(int posizione)* che elimina il contenuto della posizione data.
- Aggiungere a Memoria un metodo **public int maxBlock()** che restituisce il numero massimo di posizioni consecutive libere in memoria.

[si consiglia di aggiungere come campo della calsse anche un array di booleani di dimensione B che tenga conto delle posizioni libere ed occupate della memoria.]

Esercizio 5 (9 punti)

Si considerino le seguenti classi.

```
class A{
    {
        stampa();
    }
    int x=9;
    {
        stampa();
    }
    public A(){
        x=10;
        stampa();
    }
    public A(int x){
        this();
        this.x=x*2;
        stampa();
    }
    private void stampa(){
        System.out.println(x);
    }
}

class B extends A{
    public B(int x){
        super(3*x);
    }
}
```

Si dica cosa viene stampato a video dalle seguenti creazioni di istanza.

```
  A a = new A(7);
  B b = new B(11);
```

Esercizio 7 (9 punti)

- Si scriva una classe astratta **Exp** per rappresentare espressioni aritmetiche di somma e moltiplicazione tra interi.
- La classe deve avere un metodo **public abstract int eval ();** che sarà implementato dalle classe che estendono **Exp** con lo scopo di fornire una valutazione dell'espressione.
- La classe deve inoltre implementare un metodo **public boolean LEQ (Exp b)** che ritorna true se e solo se l'espressione su cui è invocato è minore o uguale a b. *[si sfrutti il metodo **eval**]*
- *Si implementino infine le seguenti classi **concrete** che implementano Exp,* **in ognuna delle quali bisogna implementare un costruttore e il metodo eval***:*
    - La classe **Costante** che rappresenta un intero [già implementata]
    - La classe **Somma** che ha come campi due Exp
    - La classe **Prodotto** che ha come campi due Exp

Ad esempio, per creare l'espressione (3+5)*8 si dovrà poter scrivere

new Prodotto(new Somma(new Costante (3), new Costante(5)),new Costante(8))

Ecco l'implementazione della classe Costante:

```
public class Costante extends Exp {
    private int k;
    public Costante (int k){
        this.k=k;
    }
 public  int eval (){
     return k;
 }
}
```

```
public class Somma extends Exp {
 private Exp e1,e2;
...
...
...
}
```

Attenzione:

- Scrivere **subito** nome, cognome, matricola e numero del compito su OGNI FOGLIO.
- non è ammesso **per nessun motivo** l'uso di telefoni cellulari, calcolatrici, etc…
- **non** è possibile consultare appunti, libri, dispense.