Programmazione

Appello del 25/01/2011

Esercizio 1 (6 punti)

Scrivere l'output del seguente programma Java

```
public class Main {
public static void main(String[] args) {
        // TODO code application logic here
        int [] A = {1,55,8,21,13,43,10,56};
        stampa(A);
        enigma(A);
        stampa(A);
    }

public static void stampa (int [] A ){
        for (int x:A) {
            System.out.print (x + " ");
        }
        System.out.println();
    }

public static void enigma (int A[]){
        for (int i=0; i< A.length; i++)
        {
            for (int j=A.length-2; j>=i ; j--)
            {
                if (A[j]>A[j+1])
                {
                    int temp=A[j];
                    A[j]=A[j+1];
                    A[j+1]=temp;
                }
            }

        }
    }
}
```

Esercizio 2 (6 punti)

Scrivere un metodo di classe ***coprimi*** che prende in input due interi a e b, e restituisce in output *true* se e solo se a e b non hanno divisori interi maggiori di 1 in comune.

Esercizio 3 (6 punti)

Scrivere un metodo di classe **Differenza** che presi in input due Array A e B del tipo generico T, restituisce in output un Vector<T> contenente gli elementi presenti nella differenza insiemistica tra A e B (ovvero gli elemeni di A che NON siano presenti in B).

[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.*

Esercizio 4 (6 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 pubblici che seguono.
- Aggiungere a Memoria un metodo **private int freeBlock(***int length***)** che restituisce la posizione iniziale di un blocco consecutivo di *length* locazioni libere. Se tale blocco non è presente, viene ritornato il valore convenzionale **-1**.
- Aggiungere a Memoria un metodo **public int *set****(byte [] values)* che memorizza consecutivamente in Memoria tutti i valori presenti nell'array *values.* Il metodo sfrutta il metodo privato **freeBlock** e ritorna la locazione in Memoria del primo valore memorizzato; viene sollevata un'opportuna eccezione se non c'è sufficiente spazio libero per la memorizzazione.
- Aggiungere a Memoria un metodo **public byte[] *get****(int start, int length)* che restituisce sottoforma di array i primi *length* valori memorizzati in memoria a partire dalla locazione start. Qualora almeno una di tali posizioni sia non assegnata, oppure inesistente, viene sollevata un'opportuna eccezione **java.lang.IndexOutOfBoundsException**.
- Aggiungere a Memoria un metodo **public void *reset****(int start, int length)* che elimina il contenuto di *length* posizioni a partire dalla posizione *start*.

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

Esercizio 5 (6 punti)

Si considerino le seguenti classi.

```
class A{
    int method(A a){
     return 1;
    }
    int method(B a){
     return 2;
    }
}

class B extends A{
    int method(A a){
     return 3;
    }
    int method(B a){
     return 4;
    }
 }
```

Si dica cosa viene stampato a video dal seguente codice:

```
      A a = new A();
      B b = new B();
      A c = new B();
      System.out.println(a.method(a));
      System.out.println(a.method(c));
      System.out.println(b.method(b));
      System.out.println(b.method(c));
      System.out.println(c.method(a));
```

```
        System.out.println(c.method(c));
```

Esercizio 6 (6 punti)

- Si scriva una classe astratta **Temperatura** per rappresentare valori di temperatura
- La classe deve avere un metodo **public abstract double toKelvin ();** che sarà implementato dalle classi che estendono **Temperatura** con lo scopo di fornire una valutazione della temperatura in gradi Kelvin.
- La classe deve inoltre implementare un metodo **public boolean LEQ (Temperatura b)** che ritorna true se e solo se la temperatura su cui è invocato è minore o uguale a quella di b. *[si sfrutti il metodo **toKelvin**]*
- *Si implementino infine le seguenti classi **concrete** che implementano* **Temperatura*, in ognuna delle quali bisogna implementare un costruttore che prende un double e il metodo toKelvin**:*
    - La classe **TemperaturaCelsius** che possiede una variabile d'istanza di tipo double che memorizza la temperatura in gradi centigradi, ed il cui costruttore prende in input il valore della temperatura in gradi centigradi.
    - La classe **TemperaturaFahrenheit** che possiede una variabile d'istanza di tipo double che memorizza la temperatura in gradi Fahrenheit, ed il cui costruttore prende in input il valore della temperatura in gradi Fahrenheit.

[Si ricorda che:
°K = (°F + 459,67) / 1,8 °K = °C + 273,15
]

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.