COGNOME: **NOME:** **MATRICOLA:**

Secondo Compitino di Programmazione per BioInformatica, 17-giu-2015

Esercizio 1

Si consideri la seguente classe per le liste:

```
public class List {
    private int head;
    private List tail;
    public List(int head, List tail) {
         this.head = head;
         this.tail = tail;
    }
}
```

Si aggiunga un metodo di istanza `public List getOrdered()` **ricorsivo**, che restituisca la lista ordinata in modo crescente a partire dalla lista `this`. La lista `this` non deve comunque essere modificata. Ad esempio, se `this` è `[3;2;4;8;2]`, allora `getOrdered()` deve restituire la lista `[2;2;3;4;8]`.

La soluzione deve essere *ricorsiva*: è **vietato** usare i costrutti **while, for o do while**. Il metodo `getOrdered()` può invocare altri metodi eventualmente aggiunti alla classei (l'utilizzo di costrutti while for e do while comporta la nullitá dell'esercizio).

Esercizio 2

Sia data la classe per le liste dell'**esercizio 1** si aggiunga il metodo di istanza `public String toString()` che restituisca sotto forma di stringa la lista `this`. La lista sotto forma di stringa dovrá essere: `[2;2;3;4;8]`.

(continua)

Esercizio 3

Si considerino le seguenti classi (ogni classe è in un file separato):

```
public class Xe {
    public int a = 0;
    private int b = 0;
    public Xe() { this.a = 100; }
    public void f(int a) {}
    public void g(long a, int b) {}
}

public class Ye extends Xe {
    public int b = 1;
    public void g(int c, long d) { this.b = super.b; }
}

public class XeYeMain {
    public static void main(String[] args) {
        Xe x = new Xe();
        Ye y = new Ye();
        Xe z = y;
        Ye w = (Ye)new Xe();
        x.g(2,3);
        y.g(2,3);
        x.f(5);
        y.f(6L);
        x = null;
        x.f(1);
        if (false) {
            y = null;
            y.f(0);
        }
        z.g(3L,4);
        int p = y.b - ((Xe)y).b;
    }
}
```

Si elenchino tutti gli errori generati dalla compilazione delle tre classi e (assumendo di commentare tutti i comandi che generano errori di compilazione) tutti gli errori che si verificheranno a run-time (non limitatevi a indicare il primo che si verifica, ma arrivate in fondo al sorgente!). Ogni errore individuato deve essere accompagnato da una breve descrizione del perché si verifica.

Soluzione 1

```
public class List {
    private int head;
    private List tail;

    public List(int head, List tail) {
        this.head = head;
        this.tail = tail;
    }
```

```
/* E' necessario scrivere un metodo ricorsivo che inserisca un elemento in modo
   Ordinato nella lista lincata */
    public void insertOrd(int val){
        if (this.head > val){ // il valore da inserire viene prima del corrente
            this.tail = new List(this.head, this.tail);
            this.head = val;
        }
        else if (this.tail == null) this.tail = new List(val, null);
        else tail.insertOrd(val);
    }


/* La soluzione vera e propria  che si appoggia sul metodo precedente */
    private void getOrdered(List l){
        if (this.tail != null) {
            l.insertOrd(this.tail.head);
            this.tail.getOrdered(l);
         }
    }

    public List getOrdered() {
        List l = new List(this.head, null); // Creo nuova lista con primo elemento
        getOrdered(l);
        return l;
    }
```

Soluzione 2

```
/**** Versione toString() ricorsiva ******
    private String convert(){
        if ( tail == null ) return "" + head;
        else return "" + head + ";" + tail.convert();
    }

    public String toString(){
       return "[" + convert() + "]";
    }
****************************************/

/**** Versione iterativa ****************/

 public String toString(){
       String s = "[" + this.head;
       List l = this.tail;
       while (l != null){
           s = s + ";" + l.head;
```

```
            l = l.tail;
        }
        s = s + "]";
        return s;
    }

     public static void main(String[] args) {
        List l = new List(3,new List(2,new List(4,
                    new List(5,new List(6,new List(8,
                    new List(2,new List(3,new List(7,
                    new List(8,new List(9,null)))))))))));

        List n = l.getOrdered();
        System.out.println(l);
        System.out.println(n);
    }
}
```

```
OUTPUT: [3;2;4;5;6;8;2;3;7;8;9] [2;2;3;3;4;5;6;7;8;8;9]
```

Soluzione 3

```
public class Xe {
    public int a = 0;
    private int b = 0;
    public Xe() { this.a = 100; }
    public void f(int a) {}
    public void g(long a, int b) {}
}

public class Ye extends Xe {
    public int b = 1;
    public void g(int c, long d) {
// Comp:    this.b =  super.b;  /* non accessibile perch privato*/
    }
}

public class XeYeMain {
    public static void main(String[] args) {
        Xe x = new Xe();
        Ye y = new Ye();
        Xe z = y;
/* Run time   Ye w = (Ye)new Xe(); Run time: Cast Errato */
        x.g(2,3);
//        y.g(2,3);   /* Comp: Ambiguo Xe:g(long, int) - Ye:g(int, long) */
        x.f(5);
//        y.f(6L);    /* Comp: Xe:f(int) non  si pu chiamare per long */
/*        x = null;
/* Run time:          Null pointer Exception (assegnato a null sopra) */
        x.f(1);
        if (false) {
            y = null;
            y.f(0);
        }
        z.g(3L,4);
//        int p = y.b;  //Comp: Mebro privato - ((Xe)y).b */;
        int p = y.b;
    }
}
```

Soluzione 4

```
public class XYmain{
    public static void main(String[] args) {
        X x = new X();
        Y y = new Y();
        X z = y;
                ...
        }
}
```

System.out.println(x.isX(z) + " " + y.isX(x));
`true true`
x.f(999);
`sono f(int a) di X 999 0`
x.f(888L);
`sono f(long a) di X 888 0`
y.f(777);
`sono f(int a) di X 777 0`
y.f(666L);
`sono f(long a) di Y 666 1`
z.f(555);
`sono f(int a) di X 555 0`
z.f(444L);
`sono f(long a) di Y 444 1`
z.g(-1);
`sono g(long a) di X -1 0`
y.g(-2);
`sono g(int a) di Y -2 1`