Esercitazione 10: interfaccia Comparable e liste

Esercizio 1 (Interfaccia Comparable)**.** Si considerino le classi `Rubrica` e `Contatto` viste a lezione e fornite per comodità su Moodle insieme a questo testo.

Si modifichi la classe `Contatto` in modo che implementi l'interfaccia `Comparable<Contatto>`. Si sfrutti questo fatto per modificare la classe `Rubrica` in modo che esponga un metodo `sort()` che ordina l'array `contatti`. Si faccia poi in modo che, se `contatti` non è stato modificato dopo l'ultima `sort()`, il metodo privato `cercaIndice()` esegua una ricerca binaria sfruttando sempre il fatto che `Contatto` implementa `Comparable<Contatto>`.

Si estenda infine `RubricaDemo` per testare le modifiche fatte.

Esercizio 2. Date le classi

```
// Node.java
public class Node {
    private int elem;
    private Node next;

    public Node(int elem, Node next) {
        this.elem = elem;
        this.next = next;
    }

    public int getElem()
    { return elem; }

    public Node getNext()
    { return next; }

    public void setElem(int elem)
    { this.elem = elem; }

    public void setNext(Node next)
    { this.next = next; }
}

// MyList.java
public class MyList {
    private Node first; // Riferimento al primo nodo della lista

    public MyList() {
        this.first = null;
    }

    public void insert(int elem) {
        first = new Node(elem, first);
    }
```

```
    public String toString() {
        String res = "";
        for (Node p = first; p != null; p = p.getNext()) {
            res += p.getElem();
            if (p.getNext() != null) res += ", ";
        }
        return res;
    }
}
```

realizzare un metodo `modifica` di `MyList` che aggiunge ad ogni elemento della lista la somma degli elementi che lo precedono.

Alcuni esempi:

- prima: 5
- dopo: 5

- prima: 5, 3, 1
- dopo: 5, 8, 9

- prima: -1, 1, 0, 4
- dopo: -1, 0, 0, 4

Esercizio 3. Date le classi

```
// Node.java
public class Node {
    private int elem;
    private Node next;

    public Node(int elem, Node next) {
        this.elem = elem;
        this.next = next;
    }

    public int getElem()
    { return elem; }

    public Node getNext()
    { return next; }

    public void setElem(int elem)
    { this.elem = elem; }

    public void setNext(Node next)
    { this.next = next; }
}

// MyList.java
public class MyList {
    private Node first; // Riferimento al primo nodo della lista

    public MyList() {
```

```
        this.first = null;
    }

    public void insert(int elem) {
        first = new Node(elem, first);
    }

    public String toString() {
        String res = "";
        for (Node p = first; p != null; p = p.getNext()) {
            res += p.getElem();
            if (p.getNext() != null) res += ", ";
        }
        return res;
    }
}
```

realizzare un metodo `pushSomma` di `MyList` che inserisce in testa un nodo nuovo contenente la somma dei numeri positivi contenuti nella lista (0 se la lista è vuota).

Alcuni esempi:

- prima: (lista vuota)
- dopo: 0

- prima: 1
- dopo: 1, 1

- prima: 1, -2
- dopo: 1, 1, -2

- prima: 2, -2, 4, 7, -6, -4
- dopo: 13, 2, -2, 4, 7, -6, -4

Esercizio 4. Date le classi

```
public abstract class List {
    public abstract List insert(int n, int x);
}

public class Nil extends List {
    public List insert(int n, int x) {
        /* COMPLETARE */
    }

    public String toString() {
        return "";
    }
}

public class Cons extends List {
    private int elem;   // Elemento memorizzato
```

```
    private List next;  // Riferimento al nodo successore

    public Cons(int elem, List next) {
        this.elem = elem;
        this.next = next;
    }

    public String toString() {
        return elem + ", " + next.toString();
    }

    public List insert(int n, int x) {
        /* COMPLETARE */
    }
}
```

implementare il metodo `insert` in `Nil` e `Cons` che inserisce un elemento alla posizione specificata, dove 0 significa in prima posizione. Il metodo deve ritornare la lista modificata a seguito dell'inserimento. Gestire opportunamente il caso in cui la posizione `n` risulti negativa o maggiore della lunghezza della lista.

Alcuni esempi:

- `l.toString()`: (lista vuota)
- `l.insert(0, 5).toString()`: 5

- `l.toString()`: 3, 2
- `l.insert(1, -1).toString()`: 3, -1, 2

- `l.toString()`: 3, 2
- `l.insert(0, -1).toString()`: -1, 3, 2

- `l.toString()`: 1, 2, 3
- `l.insert(10, 9).toString()`: (ERRORE: non consentito)

Esercizio 5 (Altri esercizi sulle liste)**.** Ciascuno degli esercizi nella cartella `AltriEserciziSulleListe` è specificato in un file MainN.java ($N = 0, \ldots 7$). Ognuno di tali file contiene la consegna dell'esercizio e richiede la scrittura del corpo di un metodo (di cui è fornita la signature e un corpo vuoto).