ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

# Низи и листи

## Алгоритми и податочни структури

## Аудиториска вежба 1

# Скелет на класа за низа во Java

```
public class Array<E>{
 private E data[];
 private int size;

 public Array(int capacity) {
        this.data = (E[]) new Object[capacity];
        this.size = 0;
 }

 public void insertLast(E o) {}
 public void insert(int position, E o) {}
 public void set(int position, E o) {}
 public E get(int position) {}
 public int find(E o) {}
 public int getSize() {}
 public void delete(int position) {}
 public void resize() {}

}
```

# Основни операции со низи

```
public void insertLast(E o) {
    if(size + 1 > data.length)
        this.resize();
    data[size++] = o;
}

public void insert(int position, E o) {
    // before all we check if position is within range
    if (position >= 0 && position <= size) {
        //check if there is enough capacity, and if not - resize
        if(size + 1 > data.length)
            this.resize();
        //copy the data, before doing the insertion
        for(int i=size;i>position;i--) {
            data[i] = data[i-1];
        }
        data[position] = o;
        size++;
    } else {
        System.out.println("Ne mozhe da se vmetne element na taa pozicija");
    }
}
```

# Основни операции со низи

```
public void set(int position, E o) {
    if (position >= 0 && position < size)
        data[position] = o;
    else
        System.out.println("Ne moze da se vmetne element na dadenata pozicija");
}

public E get(int position) {
    if (position >= 0 && position < size)
        return data[position];
    else
        System.out.println("Ne e validna dadenata pozicija");
    return null;
}
```

# Основни операции со низи

```
public int find(E o) {
    for (int i = 0; i < size; i++) {
        if(o.equals(data[i]))
            return i;
    }
    return -1;
}

public int getSize() {
    return size;
}
```

# Основни операции со низи

```
public void delete(int position) {
    // before all we check if position is within range
    if (position >= 0 && position < size) {
        // first resize the storage array
        E[] newData = (E[]) new Object[size - 1];
        // copy the data prior to the delition
        for (int i = 0; i < position; i++)
            newData[i] = data[i];
        // move the data after the deletion
        for (int i = position + 1; i < size; i++)
            newData[i - 1] = data[i];
        // replace the storage with the new array
        data = newData;
        size--;
    }
}
```

# Основни операции со низи

```
public void resize() {
    // first resize the storage array
    E[] newData = (E[]) new Object[size*2];
    // copy the data
    for (int i = 0; i < size; i++)
        newData[i] = data[i];
    // replace the storage with the new array
    this.data = newData;
}
```

# Користење на дефинираните методи

```
public static void main(String[] args) {
    Array<Integer> niza = new Array<Integer>( capacity: 4);

    niza.insertLast( o: 4);
    System.out.print("Nizata po vmetnuvanje na 4 kako posleden element: ");
    System.out.println(niza.toString());

    niza.insertLast( o: 7);
    niza.insertLast( o: 13);
    System.out.print("Nizata po dodavanje na 7 i 13 kako elementi: ");
    System.out.println(niza.toString());

    niza.insert( position: 1,  o: 3);
    System.out.print("Nizata po dodavanje na 3 kako element na pozicija 1: ");
    System.out.println(niza.toString());
```

# Користење на дефинираните методи

```
    niza.set(2, 6);
    System.out.print("Nizata po menuvanje na vrednosta na elementot na pozicija 2 vo 6: ");
    System.out.println(niza.toString());

    niza.delete( position: 0);
    System.out.print("Nizata po brishenje na elementot na pozicija 0 (prviot element): ");
    System.out.println(niza.toString());

    System.out.print("Na pozicija 2 vo nizata sega se naogja: ");
    System.out.println(niza.get(2));

    System.out.print("Brojot 3 sega se naogja vo nizata na pozicija: ");
    System.out.println(niza.find( o: 3));

    System.out.print("Sega na krajot goleminata na nizata e: ");
    System.out.println(niza.getSize());
}
```

# Задача 1.

- Нека се дадени две низи, кои треба да бидат со иста големина. Да се напише функција која ќе прави промени во двете низи така што доколку на дадена позиција тие имаат еднакви елементи, истите треба да се избришат и во двете низи.

# Задача 1 - решение

```
public class ChangeArrays<E> {
    public void compareAndChangeArrays(Array<E> niza1, Array<E> niza2) {
        if(niza1.getSize() != niza2.getSize()) {
            System.out.println("Nizite ne se so ista golemina!");
            return;
        }
        int size = niza1.getSize();
        int i = 0;
        while(i < size) {
            if(niza1.get(i).equals(niza2.get(i))) {
                niza1.delete(i);
                niza2.delete(i);
                size--;
            } else {
                i++;
            }
        }
    }
```

# Задача 1 - решение - main дел

```
public static void main(String[] args) {
    Array<String> niza1 = new Array<~>( capacity: 4);
    niza1.insertLast( o: "nb11");
    niza1.insertLast( o: "b1");
    niza1.insertLast( o: "b2");
    niza1.insertLast( o: "nb12");

    Array<String> niza2 = new Array<~>( capacity: 4);
    niza2.insertLast( o: "nb21");
    niza2.insertLast( o: "b1");
    niza2.insertLast( o: "b2");
    niza2.insertLast( o: "nb22");

    System.out.println("Nizite pred primenuvanjeto na funkcijata: ");
    System.out.println(niza1.toString());
    System.out.println(niza2.toString());
```

# Задача 1 - решение - main дел

```
Array<String> niza2 = new Array<~>( capacity: 4);
niza2.insertLast( o: "nb21");
niza2.insertLast( o: "b1");
niza2.insertLast( o: "b2");
niza2.insertLast( o: "nb22");

System.out.println("Nizite pred primenuvanjeto na funkcijata: ");
System.out.println(niza1.toString());
System.out.println(niza2.toString());

ChangeArrays<String> pom = new ChangeArrays<String>();
pom.compareAndChangeArrays(niza1, niza2);

System.out.println("Nizite po primenuvanjeto na funkcijata: ");
System.out.println(niza1.toString());
System.out.println(niza2.toString());

}
```

# Задача 1 - решение со ArrayList

```
public void compareAndChangeArrays(ArrayList<E> niza1, ArrayList<E> niza2) {
    if(niza1.size() != niza2.size()) {
        System.out.println("Nizite ne se so ista golemina!");
    }
    int size = niza1.size();
    int i = 0;
    while(i < size) {
        if(niza1.get(i).equals(niza2.get(i))) {
            niza1.remove(i);
            niza2.remove(i);
            size--;
        } else {
            i++;
        }
    }
}
```

Потребно е на почетокот на програмата (кодот) да ја импортирате класата ArrayList:

```
import java.util.ArrayList;
```

# Задача 1 - решение со ArrayList - main дел

```
ArrayList<Integer> niza3 = new ArrayList<Integer>( initialCapacity: 3);
niza3.add(10);
niza3.add(13);
niza3.add(7);

ArrayList<Integer> niza4 = new ArrayList<Integer>( initialCapacity: 3);
niza4.add(5);
niza4.add(13);
niza4.add(3);
```

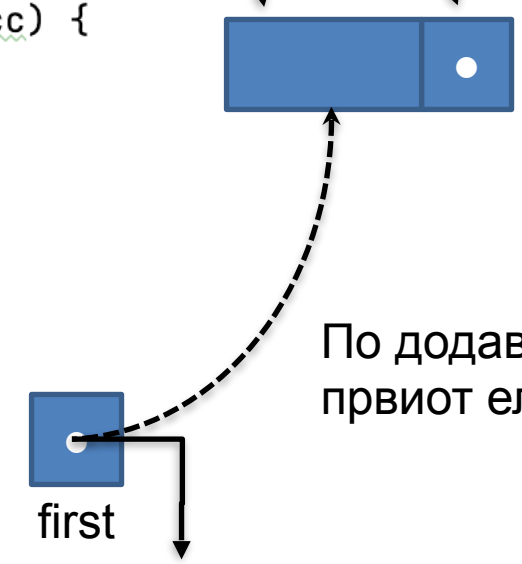
# Задача 1 - решение со ArrayList - main дел

```
System.out.println("Nizite pred primenuvanjeto na funkcijata: ");
System.out.println(niza3.toString());
System.out.println(niza4.toString());

ChangeArrays<Integer> pom2 = new ChangeArrays<Integer>();
pom2.compareAndChangeArrays(niza3, niza4);

System.out.println("Nizite po primenuvanjeto na funkcijata: ");
System.out.println(niza3.toString());
System.out.println(niza4.toString());
```

# Едностранo поврзана листа

```
public class SLLNode<E> {
    protected E element;
    protected SLLNode<E> succ;

    public SLLNode(E elem, SLLNode<E> succ) {
        this.element = elem;
        this.succ = succ;
    }
}
```

```
public class SLL<E> {
    private SLLNode<E> first;

    public SLL () {
        //kreiranje prazna lista
        this.first = null;
    }
  ...
}
```



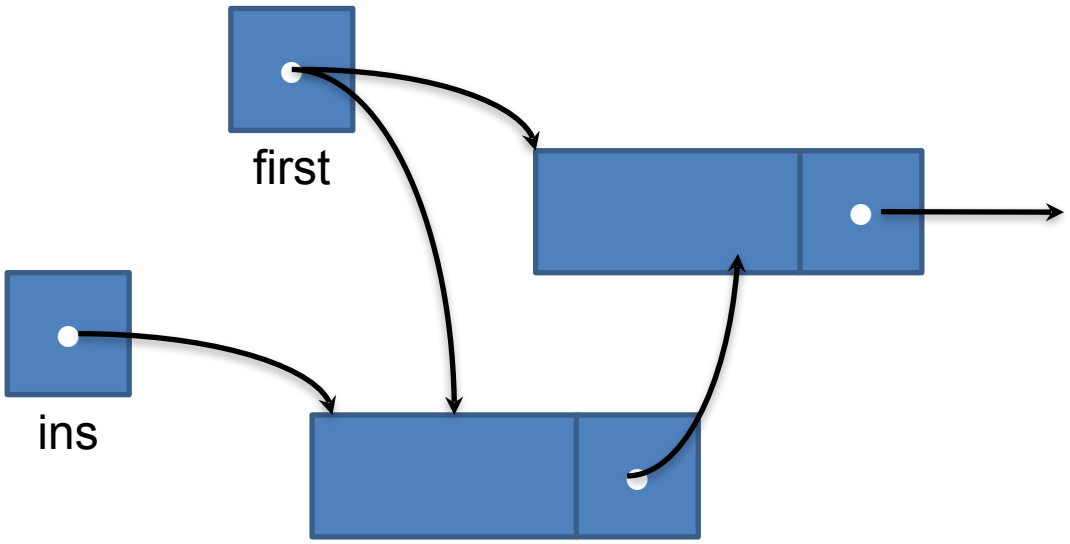

По додавање на
првиот елемент

# Едностранo поврзана листа во Java

```
public class SLL<E>{
    private SLLNode<E> first;

    public SLL () {
        // kreiranje prazna lista
        this.first = null;
    }

    public void insertFirst(E o)
    public void insertAfter(E o, SLLNode<E> after)
    public void insertBefore(E o, SLLNode<E> before)
    public void insertLast(E o)
    public E deleteFirst()
    public E delete(SLLNode<E> node)
    public SLLNode<E> find(E o)
    public int size()
    public void merge(SLL<E> in)
}
```

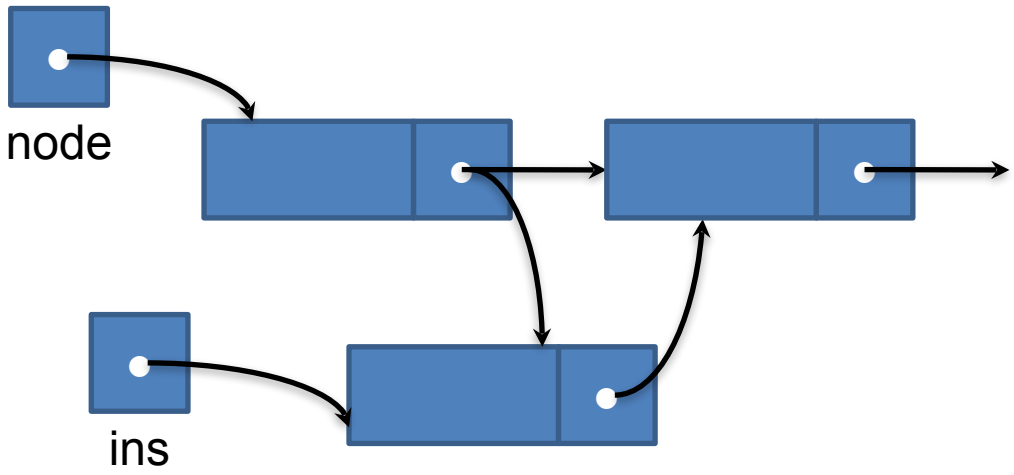
# Вметнување нов елемент - на почеток

```
public void insertFirst(E o) {
    SLLNode<E> ins = new SLLNode<E>(o, succ: null);
    ins.succ = first;
    //SLLNode<E> ins = new SLLNode<E>(o, first);
    first = ins;
}
```



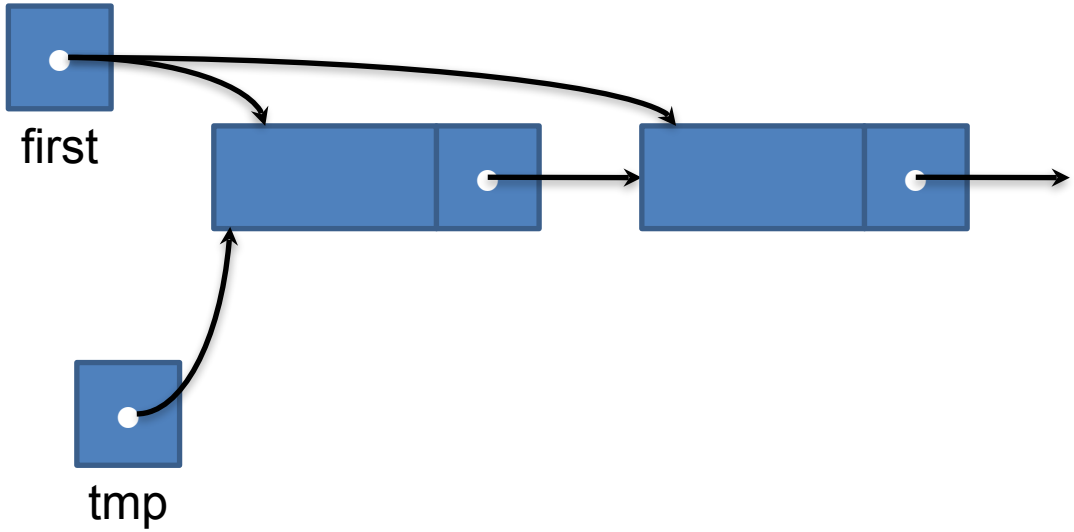
# Вметнување нов елемент - после зададен елемент

```
public void insertAfter(E o, SLLNode<E> node) {
    if (node != null) {
        SLLNode<E> ins = new SLLNode<E>(o, node.succ);
        node.succ = ins;
    } else {
        System.out.println("Dadenot jazol e null");
    }
}
```



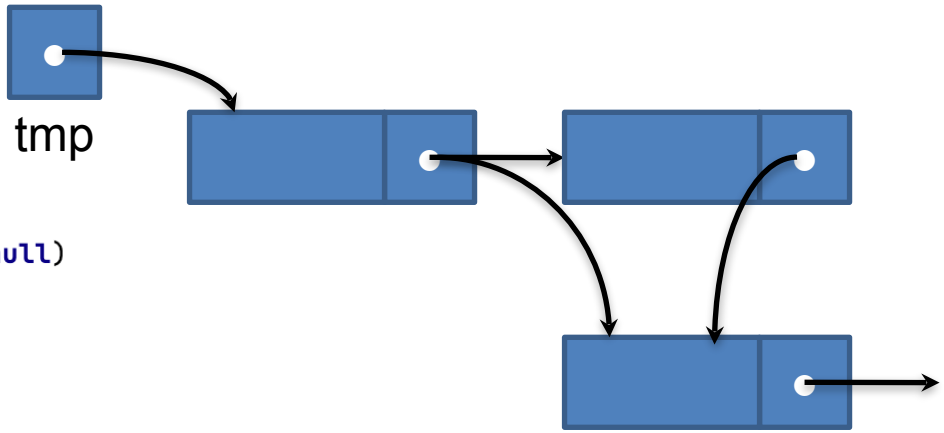
# Отстранување елемент - прв елемент

```
public E deleteFirst() {
    if (first != null) {
        SLLNode<E> tmp = first;
        first = first.succ;
        return tmp.element;
    } else {
        System.out.println("Listata e prazna");
        return null;
    }
}
```



# Отстранување елемент - специфичен зададен елемент

```
public E delete(SLLNode<E> node) {
    if (first != null) {
        SLLNode<E> tmp = first;
        if(first == node) {
            return this.deleteFirst();
        }
        while (tmp.succ != node && tmp.succ.succ != null)
            tmp = tmp.succ;
        if (tmp.succ == node) {
            tmp.succ = tmp.succ.succ;
            return node.element;
        } else {
            System.out.println("Elementot ne postoi vo listata");
            return null;
        }
    } else {
        System.out.println("Listata e prazna");
        return null;
    }

}
```



# Останати операции со едностранo поврзана листа

```
public int size() {
    int listSize = 0;
    SLLNode<E> tmp = first;
    while(tmp != null) {
        listSize++;
        tmp = tmp.succ;
    }
    return listSize;
}
```

# Останати операции со едностранo поврзана листа

```
public void insertBefore(E o, SLLNode<E> before) {

    if (first != null) {
        SLLNode<E> tmp = first;
        if(first==before){
            this.insertFirst(o);
            return;
        }
        //ako first!=before
        while (tmp.succ != before && tmp.succ!=null)
            tmp = tmp.succ;
        if (tmp.succ == before) {
            tmp.succ = new SLLNode<E>(o, before);;
        } else {
            System.out.println("Elementot ne postoi vo listata");
        }
    } else {
        System.out.println("Listata e prazna");
    }
}
```

# Останати операции со едностранo поврзана листа

```
public void insertLast(E o) {
    if (first != null) {
        SLLNode<E> tmp = first;
        while (tmp.succ != null)
            tmp = tmp.succ;
        tmp.succ = new SLLNode<E>(o, succ: null);
    } else {
        insertFirst(o);
    }
}
```

# Останати операции со едностранo поврзана листа

```
public SLLNode<E> find(E o) {
    if (first != null) {
        SLLNode<E> tmp = first;
        while (!tmp.element.equals(o) && tmp.succ != null)
            tmp = tmp.succ;
        if (tmp.element.equals(o)) {
            return tmp;
        } else {
            System.out.println("Elementot ne postoi vo listata");
        }
    } else {
        System.out.println("Listata e prazna");
    }
    return null;
}
```

# Останати операции со едностранo поврзана листа

```
public void merge (SLL<E> in){
    if (first != null) {
        SLLNode<E> tmp = first;
        while(tmp.succ != null)
            tmp = tmp.succ;
        tmp.succ = in.getFirst();
    }
    else{
        first = in.getFirst();
    }
}
```

# Користење на дефинираните методи

```
public static void main(String[] args) {
    SLL<Integer> lista = new SLL<Integer>();
    lista.insertLast( o: 5);
    System.out.print("Listata po vmetnuvanje na 5 kako posleden element: ");
    System.out.println(lista.toString());

    lista.insertFirst( o: 3);
    System.out.print("Listata po vmetnuvanje na 3 kako prv element: ");
    System.out.println(lista.toString());

    lista.insertLast( o: 1);
    System.out.print("Listata po vmetnuvanje na 1 kako posleden element: ");
    System.out.println(lista.toString());
```

# Користење на дефинираните методи

```
lista.insertLast( o: 1);
System.out.print("Listata po vmetnuvanje na 1 kako posleden element: ");
System.out.println(lista.toString());

lista.deleteFirst();
System.out.print("Listata po brishenje na prviot element: ");
System.out.println(lista.toString());

SLLNode<Integer> pom = lista.find( o: 5);
lista.insertBefore( o: 2, pom);
System.out.print("Listata po vmetnuvanje na elementot 2 pred elementot 5: ");
System.out.println(lista.toString());

pom = lista.find( o: 1);
lista.insertAfter( o: 3, pom);
System.out.print("Listata po vmetnuvanje na elementot 3 posle elementot 1: ");
System.out.println(lista.toString());
```

# Користење на дефинираните методи

```
        System.out.println("Momentalna dolzina na listata: " + lista.size());

        System.out.print("Listata po prevrtuvanje: ");
        lista.mirror();
        System.out.println(lista.toString());

        pom = lista.find( o: 2);
        lista.delete(pom);
        System.out.print("Listata po brishenje na elementot 2: ");
        System.out.println(lista.toString());

        System.out.println("Momentalna dolzina na listata: " + lista.size());

        lista.deleteList();
        System.out.print("Pecatenje na listata po nejzino brishenje: ");
        System.out.println(lista.toString());
        System.out.println("Momentalna dolzina na listata: " + lista.size());

}
```

# Задача 2.

- Да се напише функција којашто за дадена едностранo поврзана листа од цели броеви ќе го врати бројот на парни елементи во истата.

# Задача 2 - решение

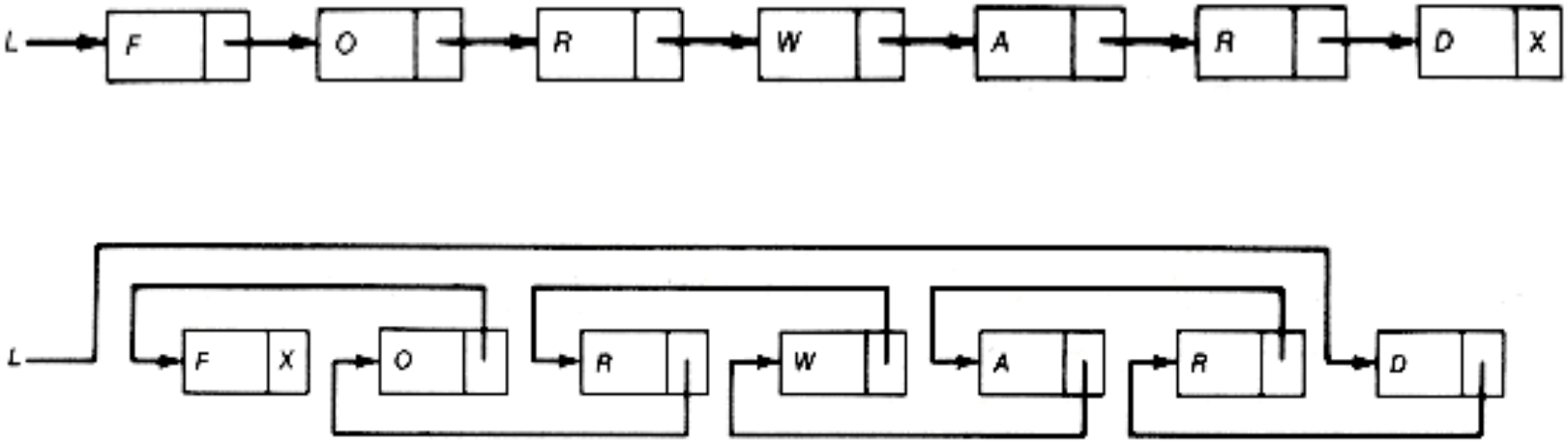
```
import java.util.Scanner;

public class EvenNumbersSLL {
    public static int evenNumbers(SLL<Integer> list) {
        SLLNode<Integer> tmp = list.getFirst();
        int res = 0;

        while(tmp!=null) {
            if(tmp.element%2==0) {
                res++;
            }
            tmp = tmp.succ;
        }

        return res;
    }
```

# Задача 2 - решение - main дел

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Vnesete go brojot na elementi vo listata:");
        int n = sc.nextInt();

        SLL<Integer> list = new SLL<>();
        System.out.println("Vnesete gi elementite na listata (celi broevi):");
        for(int i=0;i<n;i++) {
            list.insertLast(sc.nextInt());
        }

        System.out.println("Brojot na parni elementi vo vnesenata lista e: " + evenNumbers(list));
    }
}
```

# Задача 3.

- Да се напише функција што ги превртува сите врски во едностранo поврзана листа.



# Задача 3 - решение

```
public void mirror() {
    if (first != null) {
        //m=nextsucc, p=tmp,q=next
        SLLNode<E> tmp = first;
        SLLNode<E> newsucc = null;
        SLLNode<E> next;

        while(tmp != null){
            next = tmp.succ;
            tmp.succ = newsucc;
            newsucc = tmp;
            tmp = next;
        }
        first = newsucc;
    }
}
```

# Задача 3 - решение - main дел

```
public static void main(String[] args) {
    SLL<String> lista = new SLL<String>();
    lista.insertLast( o: "ovaa");
    lista.insertLast( o: "lista");
    lista.insertLast( o: "kje");
    lista.insertLast( o: "bide");
    lista.insertLast( o: "prevrtena");
    System.out.println("Listata pred da bide prevrtena: " + lista.toString());
    lista.mirror();
    System.out.println("Listata otkako e prevrtena: " + lista.toString());
}
```

# Задача 4.

- Нека се дадени две едностранo поврзани листи чии јазли се сортирани во растечки редослед. Да се напише функција која ќе ги спои двете листи во една така што резултантната листа да е сортирана. Сортирањето е подредување со слевање.

# Задача 4 - решение

```java
public class JoinSortedLists<E extends Comparable<E>> {

    public SLL<E> join(SLL<E> list1, SLL<E> list2) {
        SLL<E> rezultat = new SLL<E>();
        SLLNode<E> jazol1 = list1.getFirst(), jazol2 = list2.getFirst();
        //SLLNode<E> jazol2 = list2.getFirst();

        while (jazol1 != null && jazol2 != null) {
            if (jazol1.element.compareTo(jazol2.element) < 0) { //jazol1<jazol2
                rezultat.insertLast(jazol1.element);
                jazol1 = jazol1.succ;
            } else {
                rezultat.insertLast(jazol2.element);
                jazol2 = jazol2.succ;
            }

        }

        if (jazol1 != null) {
            while (jazol1 != null) {
                rezultat.insertLast(jazol1.element);
                jazol1 = jazol1.succ;
            }
        }

        if (jazol2 != null) {
            while (jazol2 != null) {
                rezultat.insertLast(jazol2.element);
                jazol2 = jazol2.succ;
            }
        }

        return rezultat;
    }
}
```

# Задача 4 - решение - main дел

```
public static void main(String[] args){
    SLL<String> lista1 = new SLL<String>();
    lista1.insertLast( o: "Ana");lista1.insertLast( o: "Bojana");lista1.insertLast( o: "Dejan");
    SLL<String> lista2 = new SLL<String>();
    lista2.insertLast( o: "Andrijana");lista2.insertLast( o: "Biljana");lista2.insertLast( o: "Darko");

    JoinSortedLists<String> js = new JoinSortedLists<String>();
    System.out.println(js.join(lista1, lista2));

}
```