ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

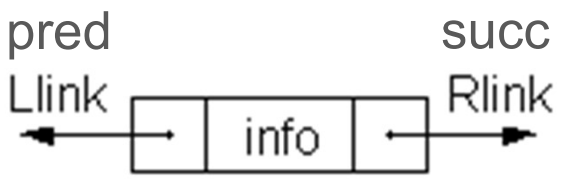
Arrays and lists 2 - Double linked lists

Algorithms and data structures

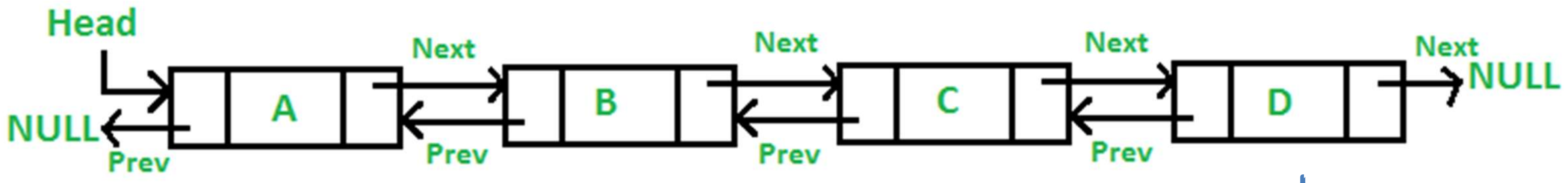
Exercise 2

Double linked lists

Llink/pred (left link) is a pointer to the predecessor



and, Rlink/succ (right link) is a pointer to the successor

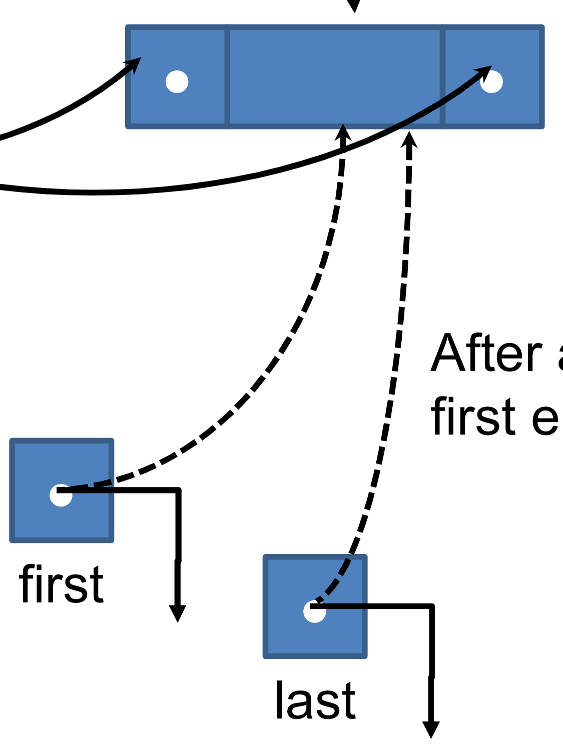


In Java an implementation where the first and the last node are stored is used

Double linked list - Java

```java
public class DLLNode<E> {
    protected E element;
    protected DLLNode<E> pred, succ;
    public DLLNode(E elem, DLLNode<E> pred, DLLNode<E> succ) {
        this.element = elem;
        this.pred = pred;
        this.succ = succ;
    }
}
```

```java
public class DLL<E> {
    private DLLNode<E> first, last;
    public DLL () {
        // kreiranje prazna lista
        this.first = null;
        this.last = null;
    }

}
```

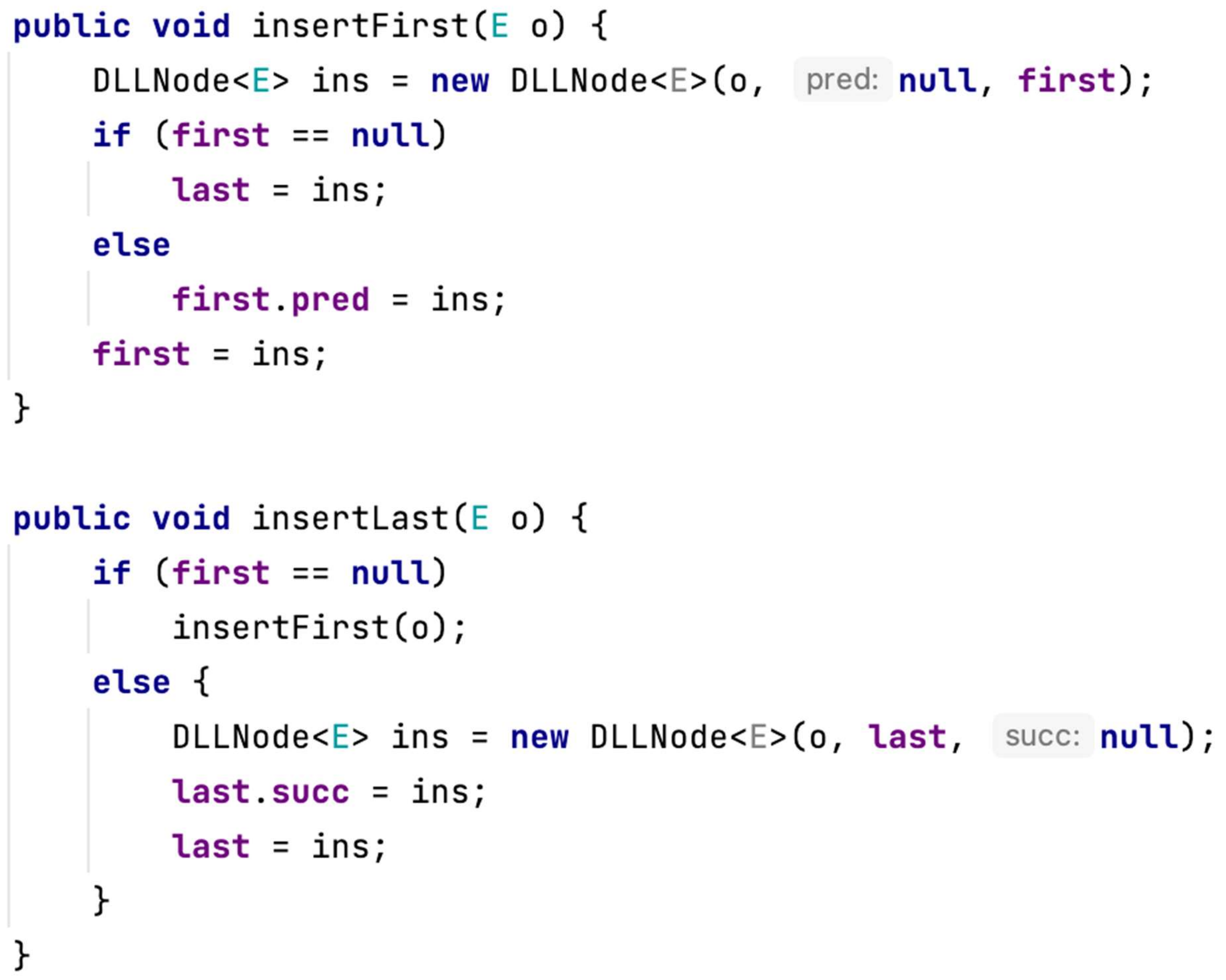


After addition of the first element

Double linked list - Java

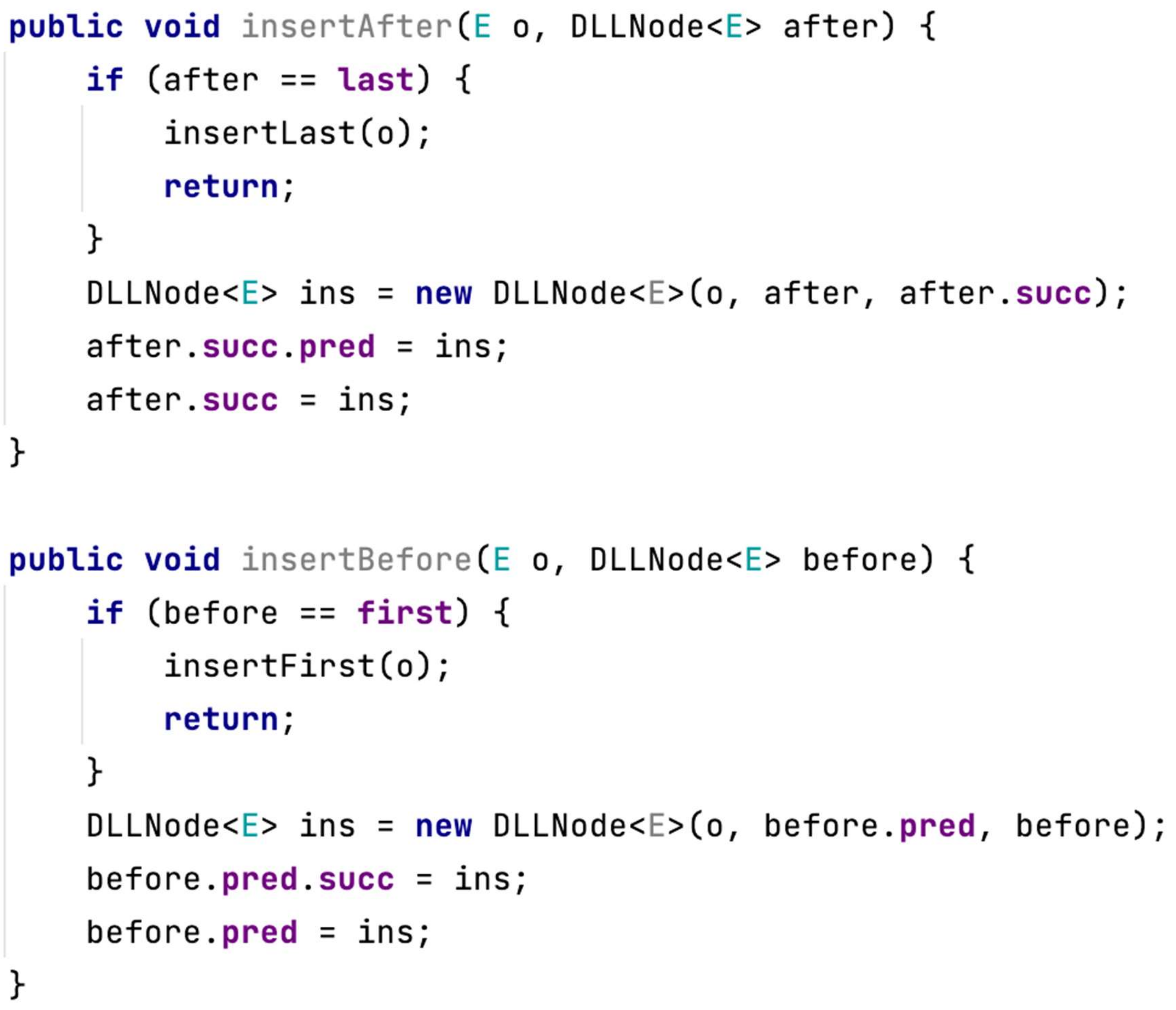
```java
public class DLL<E> {
  private DLLNode<E> first, last;
  public DLL (){
    // kreiranje prazna lista
    this.first = null;
    this.last = null;
  }
  public void insertFirst(E o)
  public void insertLast(E o)
  public void insertAfter(E o, DLLNode<E> after)
  public void insertBefore(E o, DLLNode<E> before)
  public E deleteFirst()
  public E deleteLast()
  public E delete(DLLNode<E> node)
  public DLLNode<E> find(E o)
  public DLLNode<E> getFirst()
  public DLLNode<E> getLast()
  public void deleteList ()
  public int getSize()
}
```

Element insertion



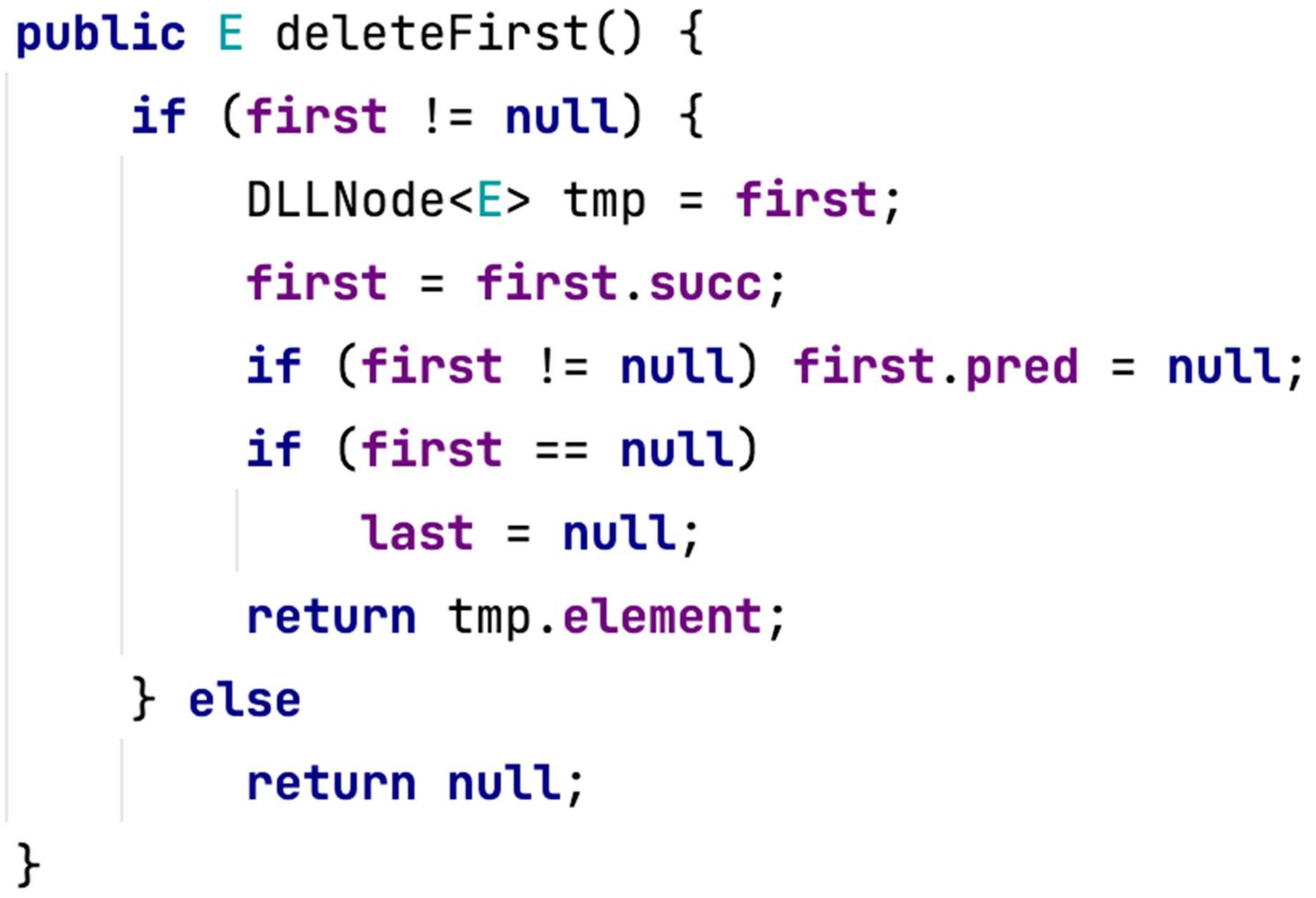
```java
public void insertFirst(E o) {
    DLLNode<E> ins = new DLLNode<E>(o, pred: null, first);
    if (first == null)
        last = ins;
    else
        first.pred = ins;
    first = ins;
}

public void insertLast(E o) {
    if (first == null)
        insertFirst(o);
    else {
        DLLNode<E> ins = new DLLNode<E>(o, last, succ: null);
        last.succ = ins;
        last = ins;
    }
}
```

Element insertion



```
public void insertAfter(E o, DLLNode<E> after) {
    if (after == last) {
        insertLast(o);
        return;
    }
    DLLNode<E> ins = new DLLNode<E>(o, after, after.succ);
    after.succ.pred = ins;
    after.succ = ins;
}

public void insertBefore(E o, DLLNode<E> before) {
    if (before == first) {
        insertFirst(o);
        return;
    }
    DLLNode<E> ins = new DLLNode<E>(o, before.pred, before);
    before.pred.succ = ins;
    before.pred = ins;
}
```

Element deletion

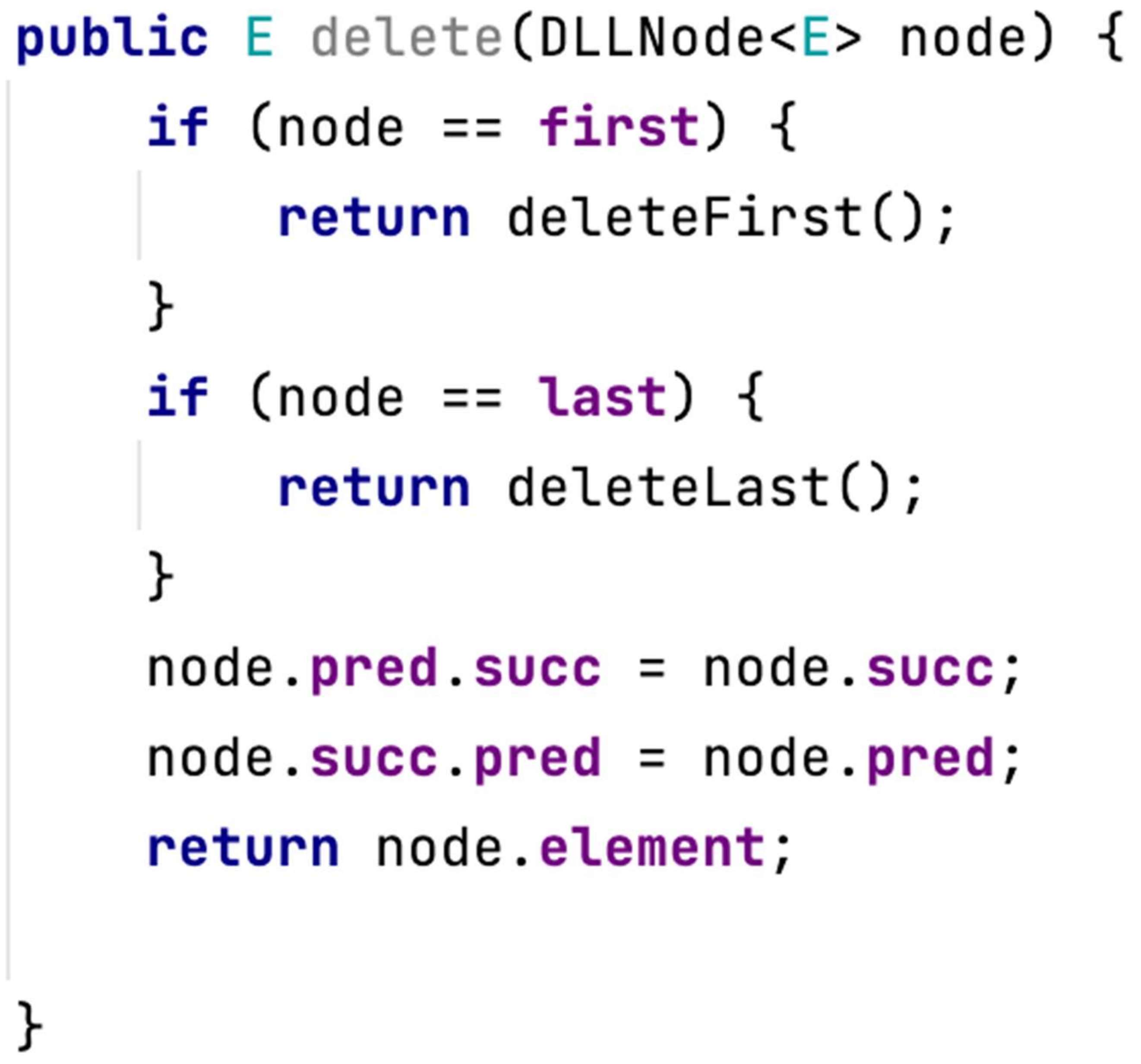


```
public E deleteFirst() {
    if (first != null) {
        DLLNode<E> tmp = first;
        first = first.succ;
        if (first != null) first.pred = null;
        if (first == null)
            last = null;
        return tmp.element;
    } else
        return null;
}
```

Element deletion

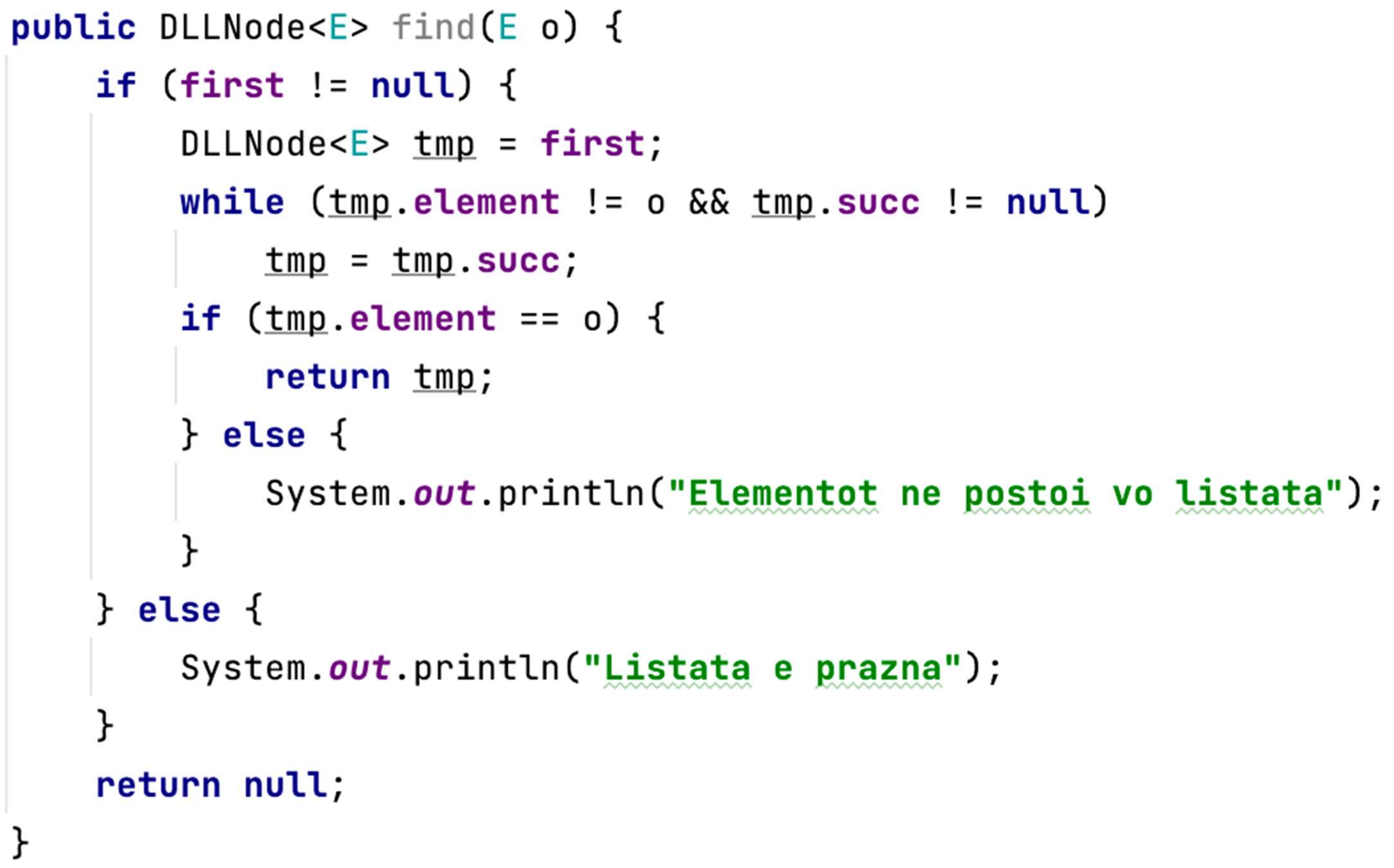
```
public E deleteLast() {
    if (first != null) {
        if (first.succ == null)
            return deleteFirst();
        else {
            DLLNode<E> tmp = last;
            last = last.pred;
            last.succ = null;
            return tmp.element;
        }
    } else
        return null;
}
```

Element deletion



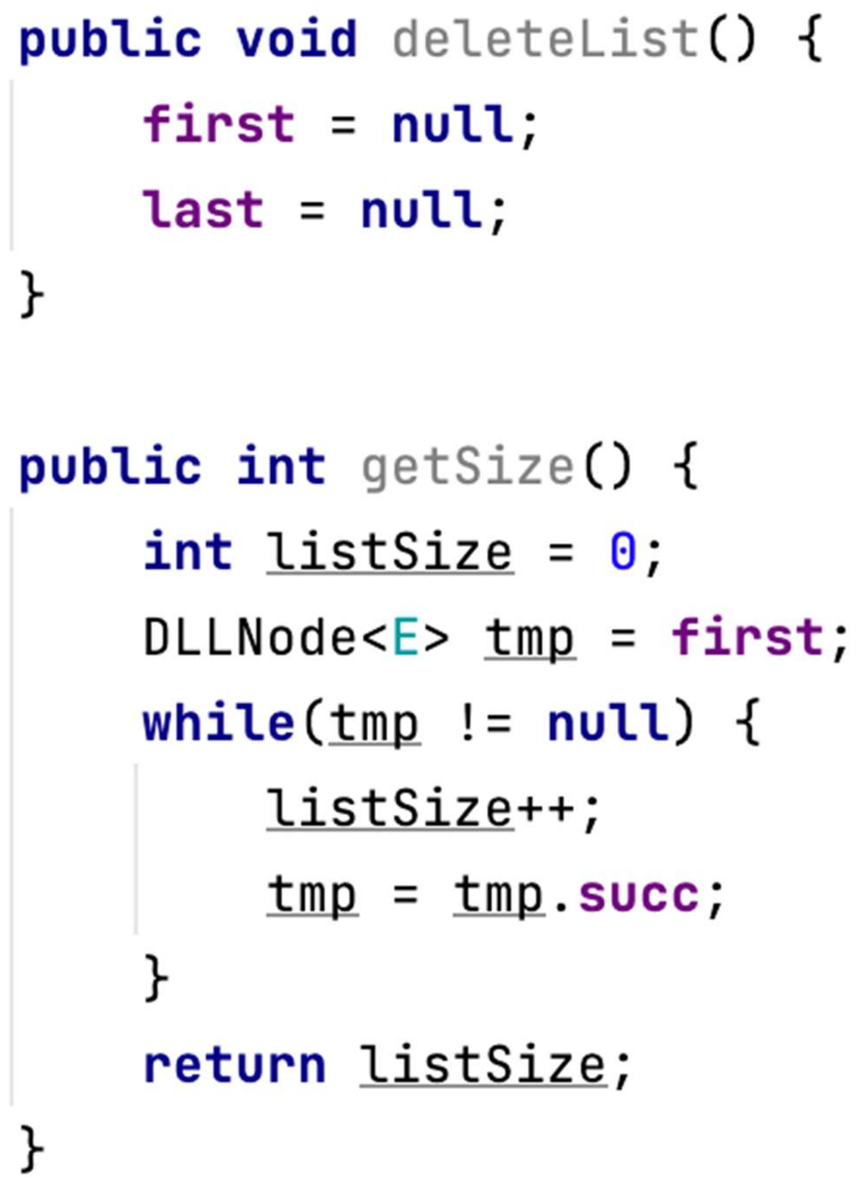
```java
public E delete(DLLNode<E> node) {
    if (node == first) {
        return deleteFirst();
    }
    if (node == last) {
        return deleteLast();
    }
    node.pred.succ = node.succ;
    node.succ.pred = node.pred;
    return node.element;

}
```

Double linked list rest operations



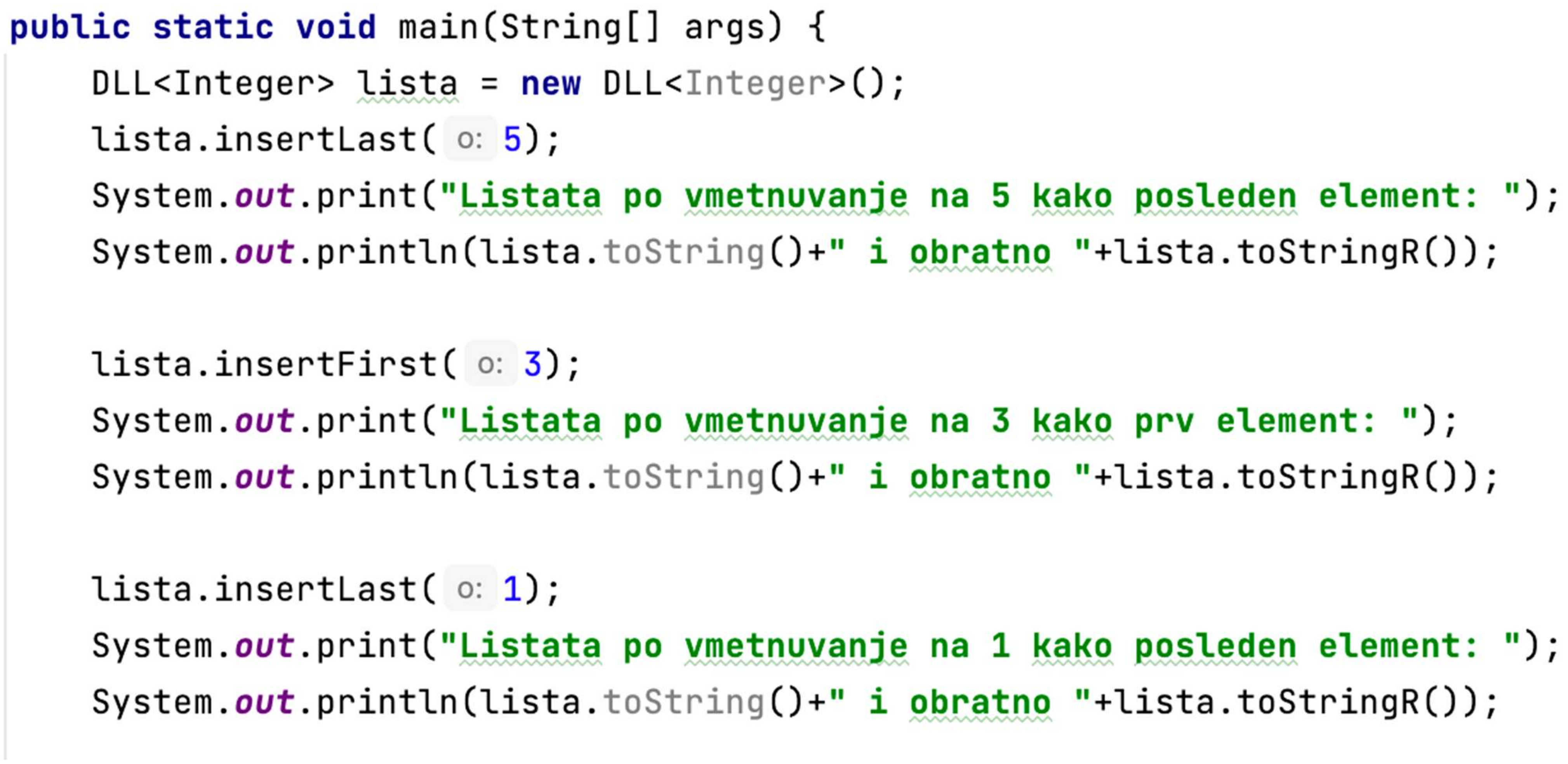
```java
public DLLNode<E> find(E o) {
    if (first != null) {
        DLLNode<E> tmp = first;
        while (tmp.element != o && tmp.succ != null)
            tmp = tmp.succ;
        if (tmp.element == o) {
            return tmp;
        } else {
            System.out.println("Elementot ne postoi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

Double linked list rest operations



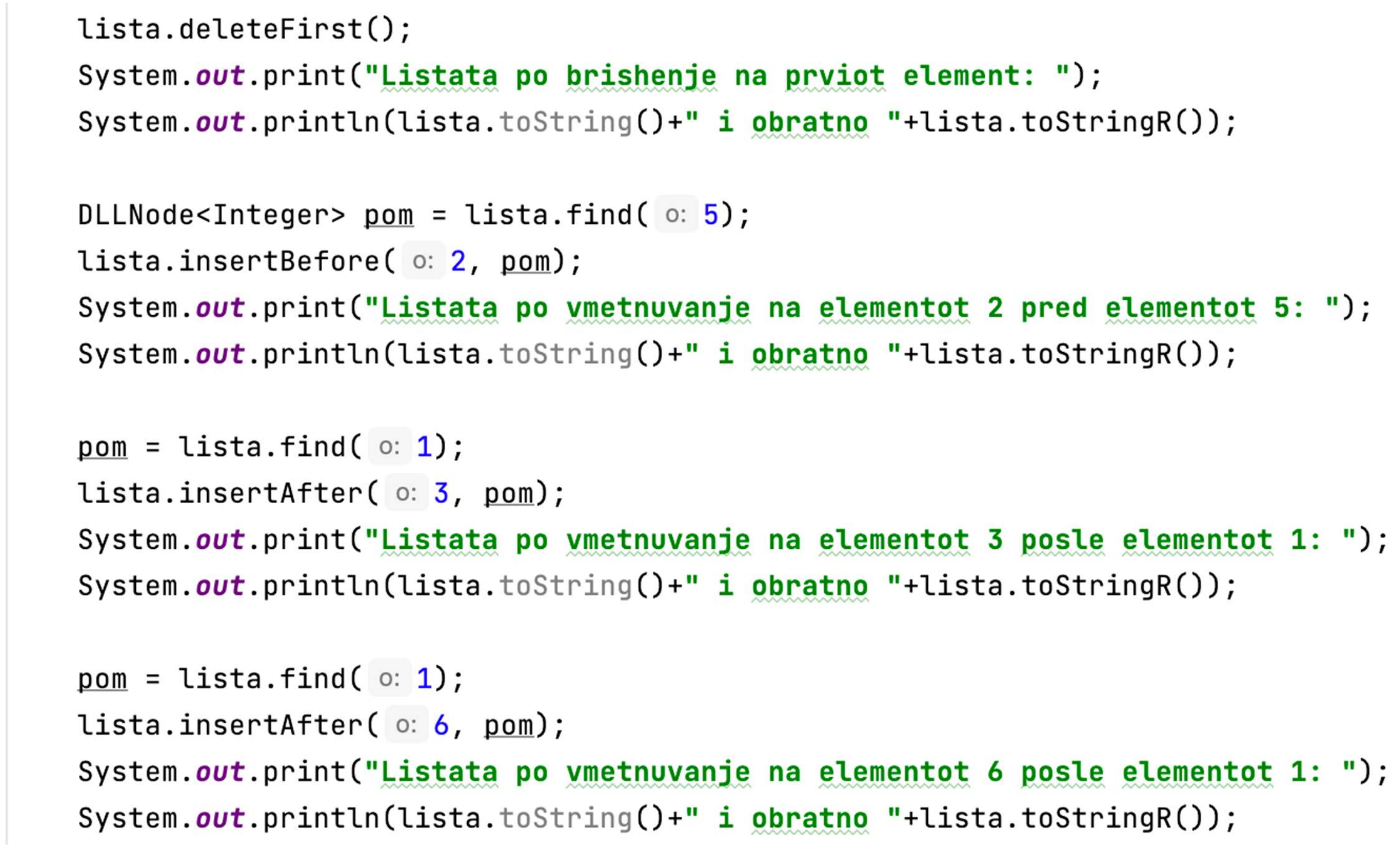
```java
public void deleteList() {
    first = null;
    last = null;
}

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

Defined methods usage



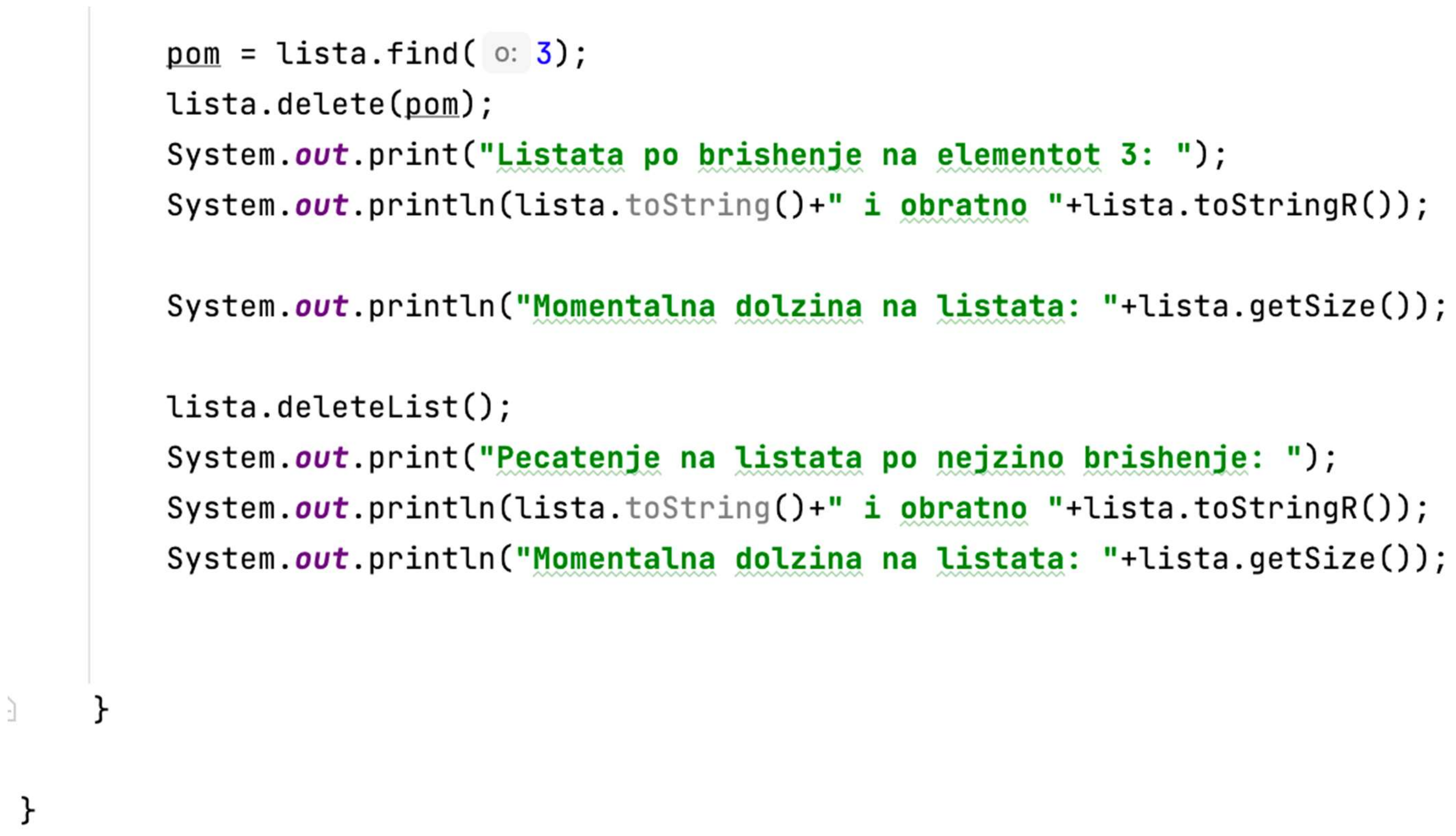
```
public static void main(String[] args) {
    DLL<Integer> lista = new DLL<Integer>();
    lista.insertLast( o: 5);
    System.out.print("Listata po vmetnuvanje na 5 kako posleden element: ");
    System.out.println(lista.toString()+" i obratno "+lista.toStringR());

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

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

Defined methods usage



```
lista.deleteFirst();
System.out.print("Listata po brishenje na prviot element: ");
System.out.println(lista.toString()+" i obratno "+lista.toStringR());

DLLNode<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()+" i obratno "+lista.toStringR());

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()+" i obratno "+lista.toStringR());

pom = lista.find( o: 1);
lista.insertAfter( o: 6, pom);
System.out.print("Listata po vmetnuvanje na elementot 6 posle elementot 1: ");
System.out.println(lista.toString()+" i obratno "+lista.toStringR());
```

Defined methods usage

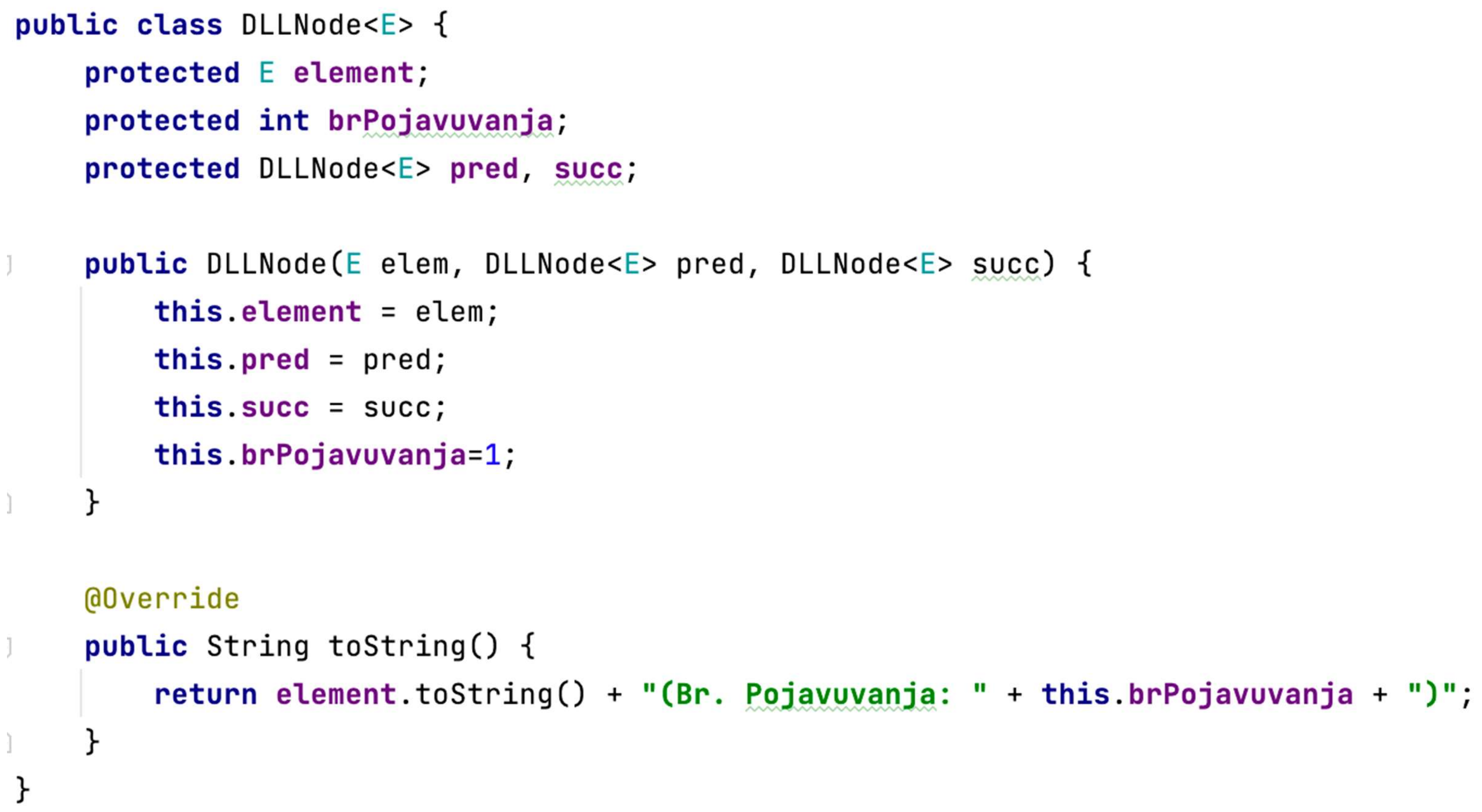


```
        pom = lista.find( o: 3);
        lista.delete(pom);
        System.out.print("Listata po brishenje na elementot 3: ");
        System.out.println(lista.toString()+" i obratno "+lista.toStringR());

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

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


    }

}
```

Problem 1.

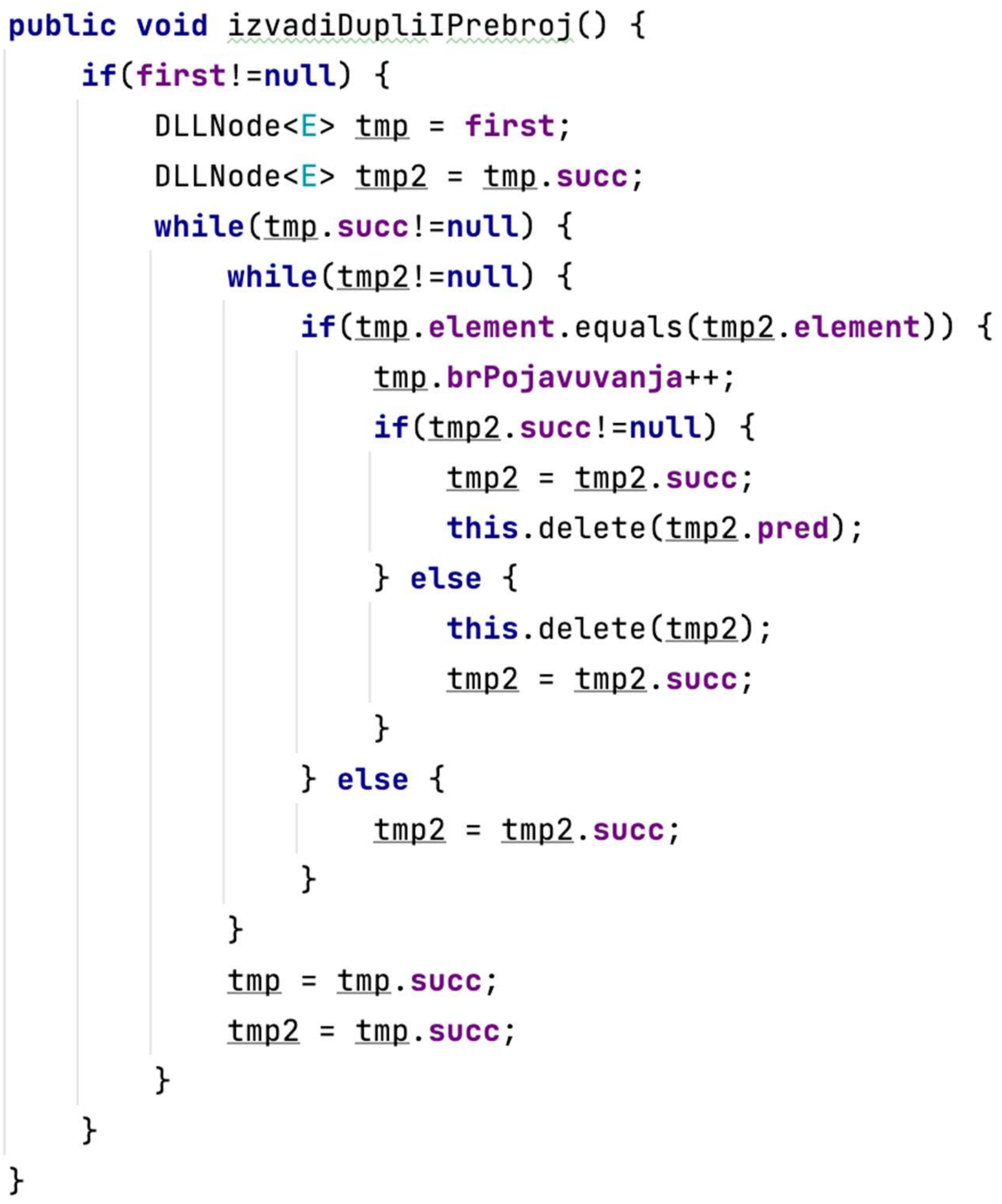
- Write a program for an arbitrary double linked list all repeating nodes to be deleted. In addition, each node of this list, in addition to the object, also to contain additional information: the number of repetitions of the given node.

Problem 1 - solution



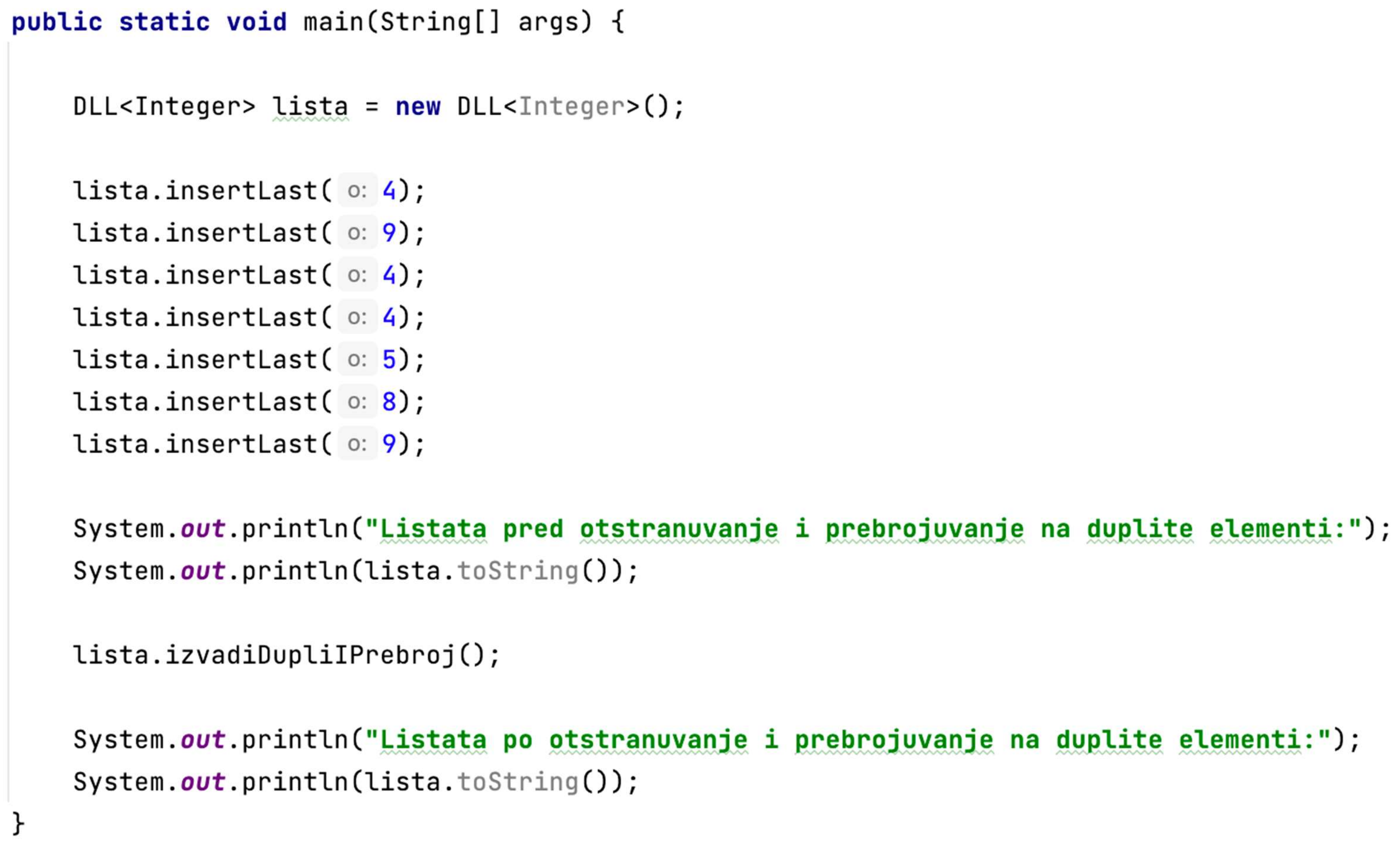
```java
public class DLLNode<E> {
    protected E element;
    protected int brPojavuvanja;
    protected DLLNode<E> pred, succ;

    public DLLNode(E elem, DLLNode<E> pred, DLLNode<E> succ) {
        this.element = elem;
        this.pred = pred;
        this.succ = succ;
        this.brPojavuvanja=1;
    }

    @Override
    public String toString() {
        return element.toString() + "(Br. Pojavuvanja: " + this.brPojavuvanja + ")";
    }
}
```

Problem 1 - solution



```java
public void izvadiDupliIPrebroj() {
    if(first!=null) {
        DLLNode<E> tmp = first;
        DLLNode<E> tmp2 = tmp.succ;
        while(tmp.succ!=null) {
            while(tmp2!=null) {
                if(tmp.element.equals(tmp2.element)) {
                    tmp.brPojavuvanja++;
                    if(tmp2.succ!=null) {
                        tmp2 = tmp2.succ;
                        this.delete(tmp2.pred);
                    } else {
                        this.delete(tmp2);
                        tmp2 = tmp2.succ;
                    }
                } else {
                    tmp2 = tmp2.succ;
                }
            }
            tmp = tmp.succ;
            tmp2 = tmp.succ;
        }
    }
}
```

Problem 1 - solution - main



```
public static void main(String[] args) {

    DLL<Integer> lista = new DLL<Integer>();

    lista.insertLast( o: 4);
    lista.insertLast( o: 9);
    lista.insertLast( o: 4);
    lista.insertLast( o: 4);
    lista.insertLast( o: 5);
    lista.insertLast( o: 8);
    lista.insertLast( o: 9);

    System.out.println("Listata pred otstranuvanje i prebrojuvanje na duplite elementi:");
    System.out.println(lista.toString());

    lista.izvadiDupliIPrebroj();

    System.out.println("Listata po otstranuvanje i prebrojuvanje na duplite elementi:");
    System.out.println(lista.toString());
}
```

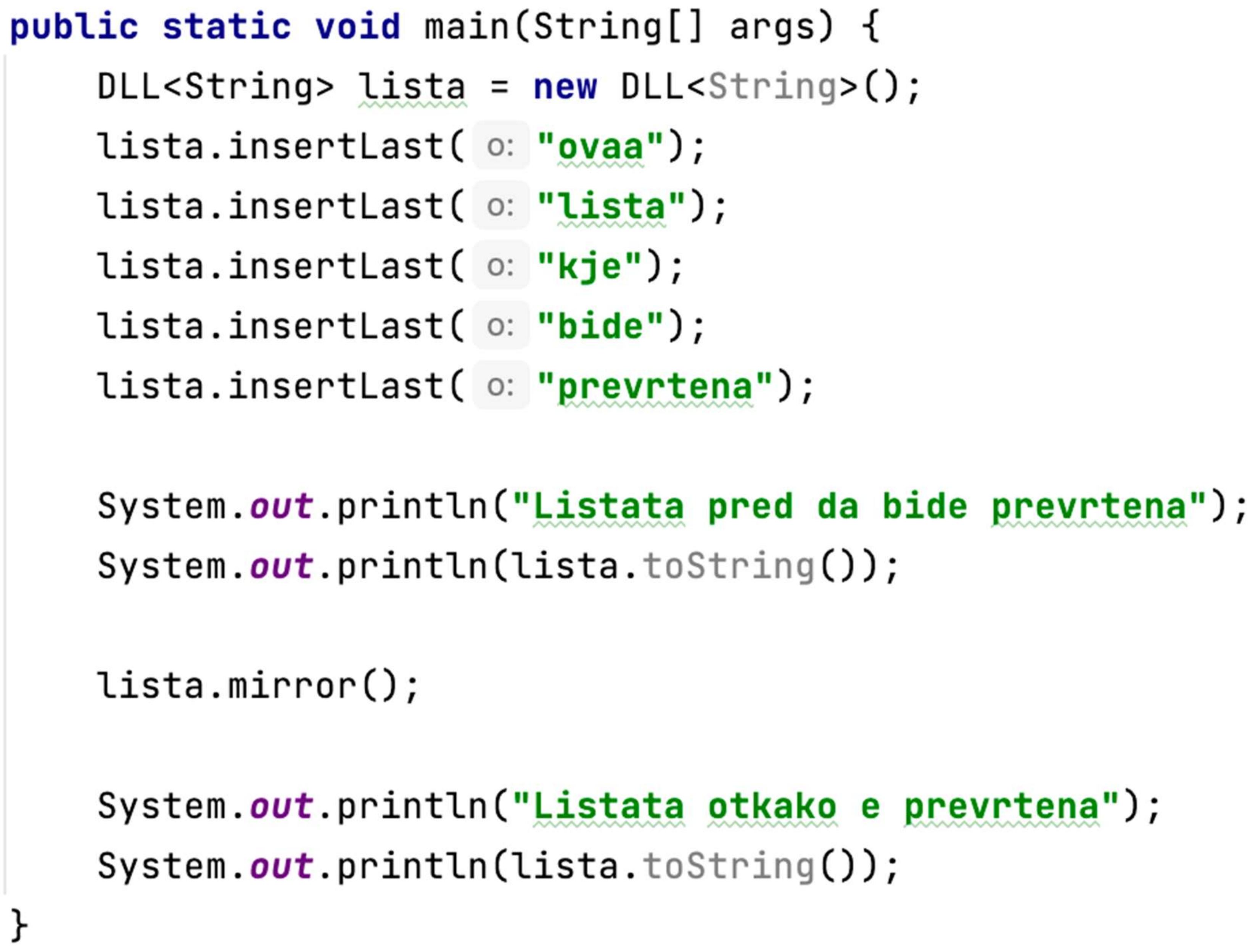
Problem 2.

- Write a function (method) that will reverse a given double linked list.

Problem 2 - solution

```java
public void mirror() {

    DLLNode<E> tmp = null;
    DLLNode<E> current = first;
    last = first;
    while(current!=null) {
        tmp = current.pred;
        current.pred = current.succ;
        current.succ = tmp;
        current = current.pred;
    }

    if(tmp!=null && tmp.pred!=null) {
        first=tmp.pred;
    }
}
```

Problem 2 - solution - main



```
public static void main(String[] args) {
    DLL<String> lista = new DLL<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");
    System.out.println(lista.toString());

    lista.mirror();

    System.out.println("Listata otkako e prevrtena");
    System.out.println(lista.toString());
}
```