МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего
образования
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий
Кафедра информатики и систем управления

ОТЧЕТ
по лабораторной работе
по дисциплине:
Алгоритмы и структуры данных

РУКОВОДИТЕЛЬ:
Капранов С.Н.
_____
(подпись)
(фамилия, и.,о.)

СТУДЕНТ:
Еричев Д.А.
_____
(подпись)
(фамилия, и.,о.)
_____
(шифр группы)
Работа защищена «___» _____
С оценкой _____

Нижний Новгород 2022

Задача 10
Написать процедуру, которая меняла бы в односвязном списке крайние элементы.

1) Программный код.

Класс List.java:

Собственная реализация списка.

Методы списка:

1) addFront(T data) Метод добавления элемента в начало списка(первым элементом)
2)set(int index, T data) метод добавления элемента по указанному индексу
3) T getDataIsElem(int index) метод возвращающий элемент по указанному индексу
4) addBack(T data) Метод добавления элемента в конец списка
5) int getLength() Метод возвращающий длину списка
6) void print() Метод который выводит информацию из элементов списка
7) void deleteFront() Метод удаления первого элемента
8) void deleteNodes() { Метод удаления всех элементов списка
9) void replaceFrontAndBack() Метод который меняет крайние элементы

```
package Laba1;

public class List<T> implements Iterable<T>{
   private Node<T> head;    //Создаем первый узел(первый элемент списка)
   private int length;     //Длина списка
   List(){               //Конструктор класса List
      length = 0;
      head = null;
   }
   public Iterator<T> iterator(){
      return new Iterator<T>() {
         Node<T> currentNode = head;
         public boolean hasNext() {
            return currentNode!=null;
         }
         public T next() {
            T data = currentNode.data;
            currentNode = currentNode.nextNode;
            return data;
         }
      };
   }
   private class Node<T> {  //Класс отдельного элемента списка
      Node(){};           //Конструктор без параметров
      T data;             // Переменная хранит информацию элемента
      Node<T> nextNode;    //Создаем новый элемент, чтобы изпользовать
его как ссылку на следующий
```

```
        Node(T data, Node<T> nextNode) {  //Конструктор с параметрами, принимающий информацию и следующий элемент

            this.data = data;
            this.nextNode = nextNode;
        }
    }
    public void addFront(T data) {      //Метод добавления элемента в начало списка(первым элементом)

        head = new Node<T>(data,head);  //В переменную первого элемента, создаем новый элемент, передаем data,
                                        //и в качестве ссылки на следующий элемент передаем еще не измененный
                                        //узел head

        length++;                       //изменяем длину списка
    }
    public void set(int index, T data) {
        if(index>length || index<=0) {
            System.out.println("Элемент не добавлен на место номер: " + index);
        }else {
            Node<T> currentElement = head;
            int countIndex = 0;
            while(countIndex+1 != index) {              //делаем +1 чтобы вставить элемент на index+1 место
                currentElement = currentElement.nextNode;      //передвигаем указатель
                countIndex++;                           //увеличиваем счетчик индексов
            }
            Node<T> newNode = new Node<>();             //создаем новый узел
            newNode.data = data;
            newNode.nextNode = currentElement.nextNode;    //в новый узел передаем ссылку на след. элемент после
                                                    //элемента который получили ранее
            currentElement.nextNode = newNode;             //в след. элемент после полученного записываем новый узел
        }
    }
    public T getDataIsElem(int index) {   //метод возвращающий элемент по указанному индексу
        Node<T> currentElement = head; //создаем узел указывающий на первый эолемент списка
        T data = null;                  //переменная для запоминания информаации из списка
        if(index > length || index < 0) { //если ввели неверный индекс
            System.out.println("Вы ввели неверный индекс.");
        }else {
            int countIndex = 0;     //переменная для контроля текущего индекса
```

```
            while(countIndex != index) {    //пока не указанный индекс
                currentElement = currentElement.nextNode;  //берем следующий
элемент списка
                countIndex++;                           //увеличиваем текущий индекс
            }
            data = currentElement.data;  //запоминаем информацию из элемента
        }
        return data;            // возвращаем информацию из элемента
    }
    public void replaceFrontAndBack() {   //Метод который меняет крайние
элементы местами
        if(head == null || head.nextNode == null) {       //Если следующий
элемент после 1-го пустой
            return;                     //ничего не делаем
        }
        Node<T> currentElement = head;       //Создаем узел для прохода по
списку с его начала
        T currentData = null;             //Переменная для запоминания
информации элемента
        while(currentElement.nextNode != null) {    //Пока не получили пустой
элемент списка
            currentElement = currentElement.nextNode;  //В узел для прохода
записываем ссылку на следующий элемент
                                        //для перемещения дальше

            if(currentElement.nextNode == null){    //Если следующий элемент
пустой, значит мы получили последний элемент списка
                currentData = currentElement.data;  //Запоминаем информацию из
последнего элемента
            }
        }
        currentElement.data = head.data;  //Записываем информацию в
последний элемент из первого
        head.data = currentData;       //Записываем информацию в первый
элемент из полученного последнего элемента
    }
    public void addBack(T data) {      //Метод добавления элемента в конец
списка
        if(head == null){             //Если 1-й элемент пустой
            head = new Node<T>(data,null);  //Создаем в нем новый узел с
информацией
        }
        else{
            Node<T> newNode = head;               //Создаем узел для прохода по
списку
            while(newNode.nextNode != null) {       //Пока не получили последний
элемент, после которого пусто
                newNode = newNode.nextNode;         //Меняем элемент для
прохода на след. элемент
            }
```

```
            newNode.nextNode = new Node<T>(data,null);  //Т.к. получили
последний элемент, создаем в нем новый узел
        }
        length++;               //Увеличиваем длину списка
    }
    public int getLength() {    //Метод возвращающий длину списка
        return length;
    }
    public void print() {       //Метод который выводит информацию из
элементов списка
        Node<T> currentElement = head;     //Создаем узел для прохода по
списку
        System.out.println("Элементы списка: ");
        if(head == null){   //Если 1-й элемент пустой
            System.out.println("Список пуст!");
            return;
        }else {
            System.out.println(head.data + " ");  //Выводим информацию из него
        }
        while (currentElement.nextNode != null) {     //Пока мы не дошли до
конца списка(т.е. до элемента после которого идет пустой элемент)
            currentElement = currentElement.nextNode;    //Меняем элемент для
прохода на след. элемент
            System.out.println(currentElement.data + " ");   //Выводим
информацию из текущего элементе
        }
    }
    public void deleteFront() {     //Метод удаления первого элемента
        head = head.nextNode;       //Первый элемент ссылается на следующий
        length--;                   //Меняем длину списка
    }
    public void deleteNodes() {     //Метод удаления всех элементов списка
        while(length > 0) {         //Пока список не пуст
            this.deleteFront();         //Вызываем метод удаления первго
элемента списка
        }
    }
}
```

Класс Main.java:

```
public class Main {
    public static void main(String[] args ) throws Exception {
        List<Integer> lst = new List<>();
//        addListElementsInFile(lst,"/home/denis/JavaPrograms/src/Laba1/file.txt");
        inputElements(lst);
        lst.print();
        lst.replaceFrontAndBack();
        lst.print();
    }
    static void addListElementsInFile(List<Integer> lst, String path)  throws
```

```
Exception {
        try{
            FileReader fr = new FileReader(path);
            Scanner scanner = new Scanner(fr);
            while(scanner.hasNext()) {
                lst.addBack(scanner.nextInt());
            }
            fr.close();
        } catch (InputMismatchException e){
            System.out.println("Из файла был считан элемент  не
соответствующий типу.");
            System.out.println("Данный элемент был удален.");
        }catch (FileNotFoundException e) {
            System.out.println("Ошибка открытия файла!");

        }
    }
    static int input(int value) {
        Scanner scanner = new Scanner(System.in);
        if(scanner.hasNextInt()){
            value = scanner.nextInt();
            return value;
        }else{
            System.out.println("Вы ввели значение не соответствующее типу,
вводите заново: ");
            return input(value);
        }
    }
    static void inputElements(List<Integer> lst) {
        int value = 0;
        int countValue = 0;
        System.out.print("Введите количество элементов целочисленного
списка: ");
        countValue = input(countValue);
        System.out.println("Вводите элементы списка: ");
        for(int i=0;i<countValue;i++) {
            value = input(value);
            lst.addBack(value);
        }

    }
}
```

2)Описание функций:

1) static void addListElementsInFile(List<Integer> lst, String path) throws Exception {
    try{
        FileReader fr = new FileReader(path);
        Scanner scanner = new Scanner(fr);
        while(scanner.hasNext()) {
            lst.addBack(scanner.nextInt());
        }
        fr.close();
    } catch (InputMismatchException e){
        System.*out*.println("Из файла был считан элемент не соответствующий типу.");
        System.*out*.println("Данный элемент был удален.");
    }catch (FileNotFoundException e) {
        System.*out*.println("Ошибка открытия файла!");

    }
}

addListElementsInFile- функция, для считывания элементов списка и файла,которая получается в качестве параметров односвязный список и название файла.

2) static int input(int value) {
    Scanner scanner = new Scanner(System.*in*);
    if(scanner.hasNextInt()){
        value = scanner.nextInt();
        return value;
    }else{
        System.*out*.println("Вы ввели значение не соответствующее типу, вводите заново: ");
        return *input*(value);
    }
}

input — функция проверка ввода целочисленного значения.

3) static void inputElements(List<Integer> lst) {
    int value = 0;
    int countValue = 0;
    System.*out*.print("Введите количество элементов целочисленного списка: ");
    countValue = *input*(countValue);
    System.*out*.println("Вводите элементы списка: ");
    for(int i=0;i<countValue;i++) {
        value = *input*(value);
        lst.addBack(value);
    }

}

inputElements- функция ввода элементов односвязного списка

3)Результаты работы программы.

Ввод элементов с клавиатуры:

```
Введите количество элементов целочисленного списка: 10
Вводите элементы списка:
1
3
4
sh
Вы ввели значение не соответствующее типу, вводите заново:
8
10
12
i
Вы ввели значение не соответствующее типу, вводите заново:
320
34
29
40
```

Повторный вывод начального списка и измененного:

```
Элементы списка:
1
3
4
8
10
12
320
34
29
40
Элементы списка:
40
3
4
8
10
12
320
34
29
1

Process finished with exit code 0
```

Чтение элементов из файла:

file.txt

```
1
3
4
8
10
12
320
34
29
40
```

```
Элементы списка считываются из файла.
Элементы списка:
1
3
4
8
10
12
320
34
29
40
Элементы списка:
40
3
4
8
10
12
320
34
29
1

Process finished with exit code 0
```