Министерство образования и науки Российской Федерации

ФГБОУ ВО Ульяновский государственный технический университет

Кафедра «Вычислительная техника»

Лабораторная работа №11

«Списки на Java»­

Выполнил студент

группы ИВТАПбд-11

Мухаметзянов Т. А.

Ульяновск, 2022

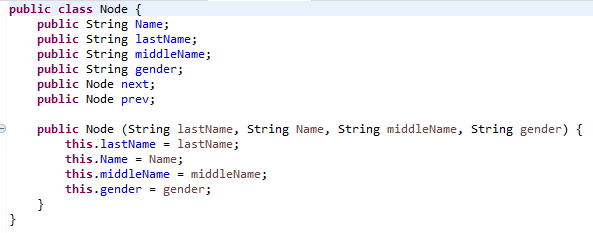
**Задача:**

Необходимо написать программу, которая позволяет работать с двухсвязным списком фамилий, имен, отчеств и пола. Отображать список в прямо или обратном направлении по выбору пользователя. Поиск осуществлять по фамилиии, при удалении удалять только первый элемент, реализовать поиск по имени с выводом всех найденных результатов.

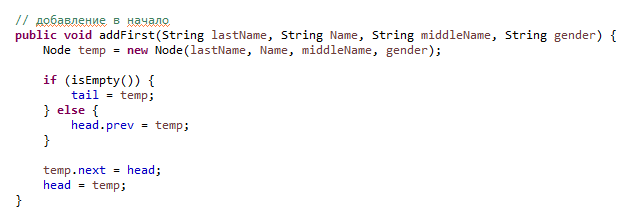
**Особенности реализации:**

Создано три класса:

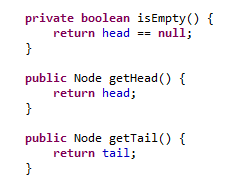
Node – хранение элементов списка.



Сами методы в классе List



Также реализованы методы проверки на пустоту (Null).



addFirst метод который добавляет элемент в начало списка.

**public** **void** addFirst(String lastName, String Name, String middleName, String gender) {

Node temp = **new** Node(lastName, Name, middleName, gender);

**if** (isEmpty()) {

tail = temp;

} **else** {

head.prev = temp;

}

temp.next = head;

head = temp;

}

addLast.- метод, который добавляет элемент в конец списка.

**public** **void** addLast(String lastName, String Name, String middleName, String gender) {

Node temp = **new** Node(lastName, Name, middleName, gender);

**if** (isEmpty()) {

head = temp;

} **else** {

tail.next = temp;

}

temp.prev = tail;

tail = temp;

}

addByIndex метод, который добавляет элемент по заданному индексу.

**public** **void** addByIndex(**int** index, String lastName, String Name, String middleName, String gender) {

Node cur = head;

**int** c = 0;

**while** (cur != **null** && c != index) {

cur = cur.next;

c++;

}

Node temp = **new** Node(lastName, Name, middleName, gender);

cur.prev.next = temp;

temp.prev = cur.prev;

cur.prev = temp;

temp.next = cur;

}

removeFirst – удаляет первый элемент списка.

**public** **void** removeFirst() {

**if** (head.next == **null**) {

tail = **null**;

} **else** {

head.next.prev = **null**;

}

head = head.next;

}

removeLast – удаление последнего элемента списка.

**public** **void** removeLast() {

**if** (head.next == **null**) {

head = **null**;

} **else** {

tail.prev.next = **null**;

}

tail = tail.prev;

}

removeAt – удаление по ключу (индексу).

**public** **void** removeAt(String queryLn, **boolean** onlyFirst) {

Node cur = head;

**while** (!cur.lastName.equals(queryLn)){

cur = cur.next;

**if**(cur == **null**) {

**return**;

}

}

**if**(cur == head) {

removeFirst();

} **else** {

cur.prev.next = cur.next;

}

**if**(cur == tail) {

removeLast();

} **else** {

cur.next.prev = cur.prev;

}

**if** (onlyFirst) {

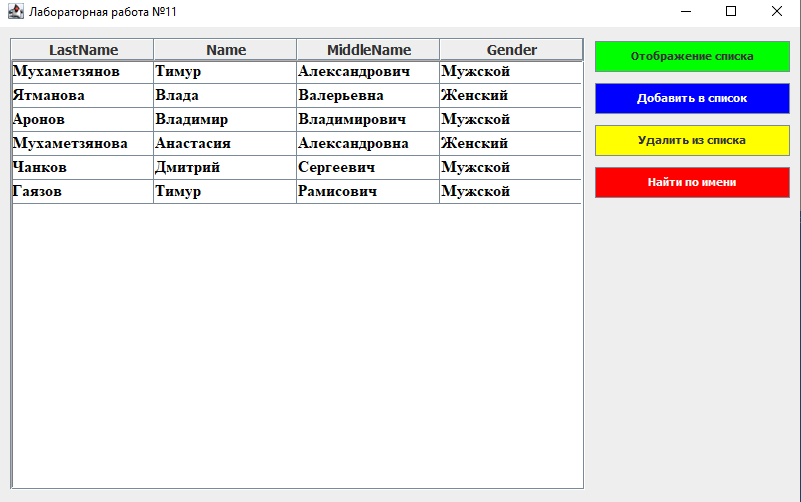
**return**;

}

}

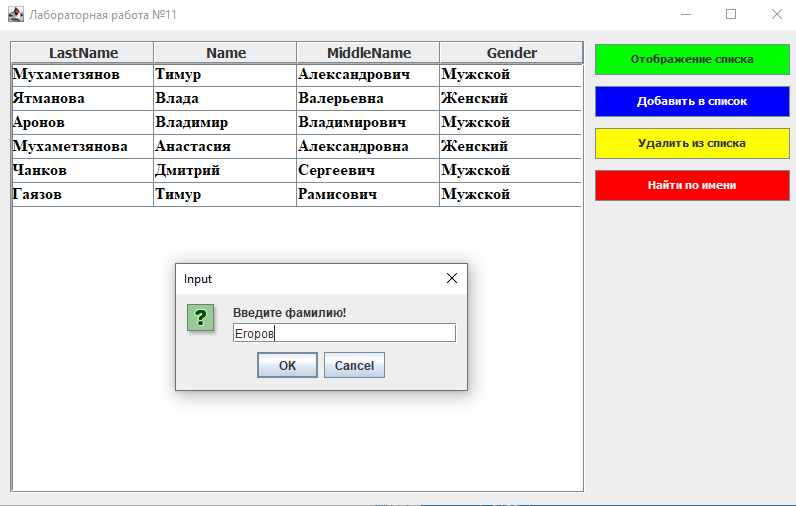
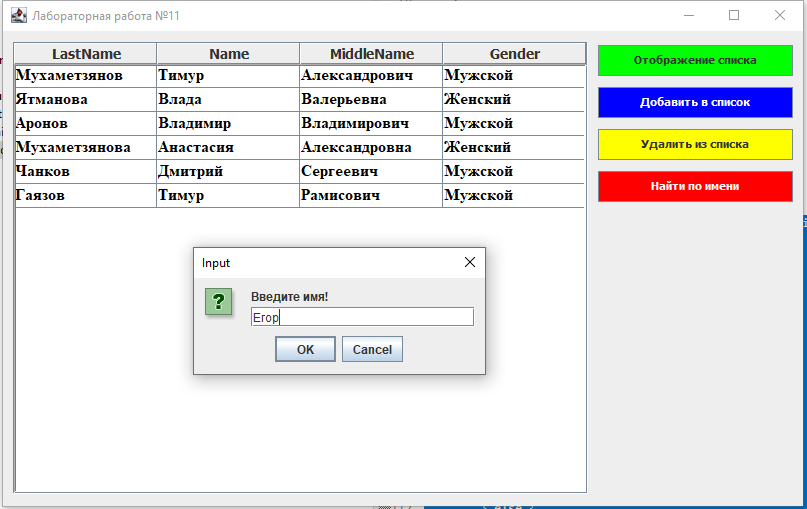
Пользователь может взаимодействовать со списком через специальный пользовательский интерфейс, при нажатии на кнопки которого происходит парсинг и работа со списком.

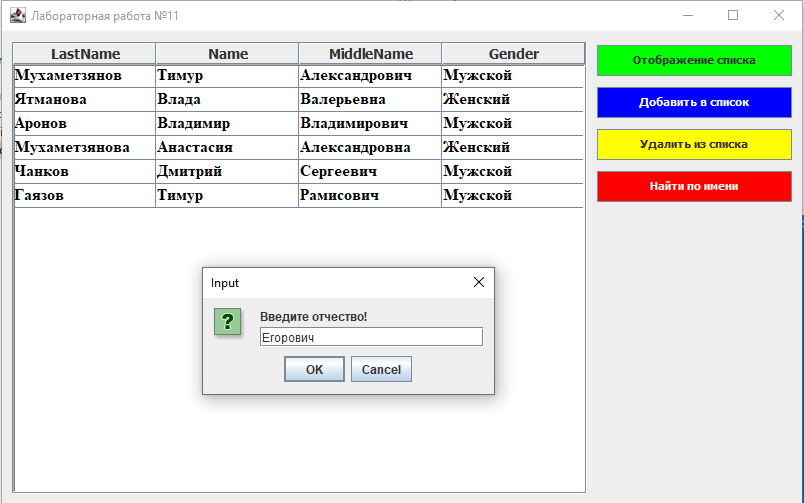
**Тестирование**

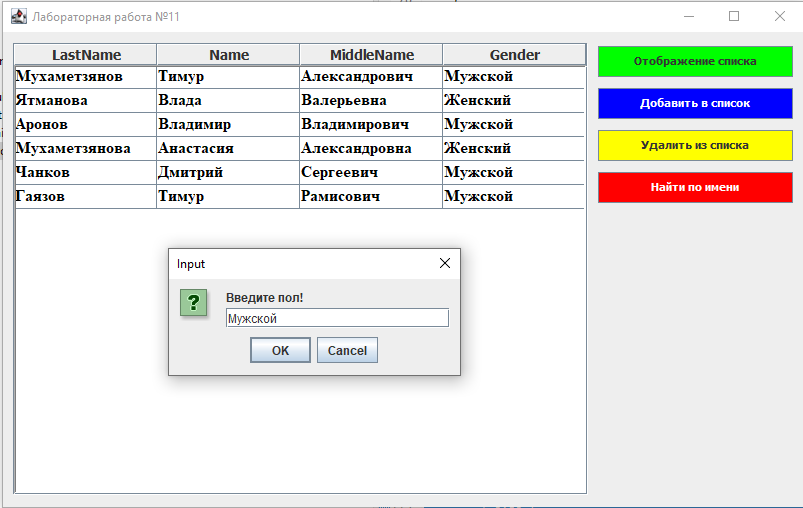


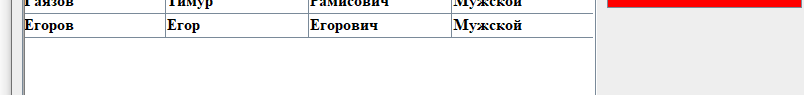


**Рис – 1. Смена направления**

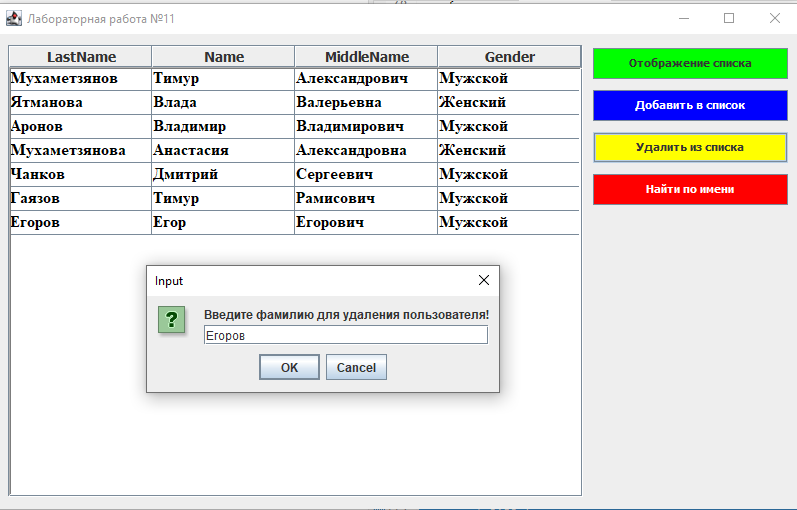
 



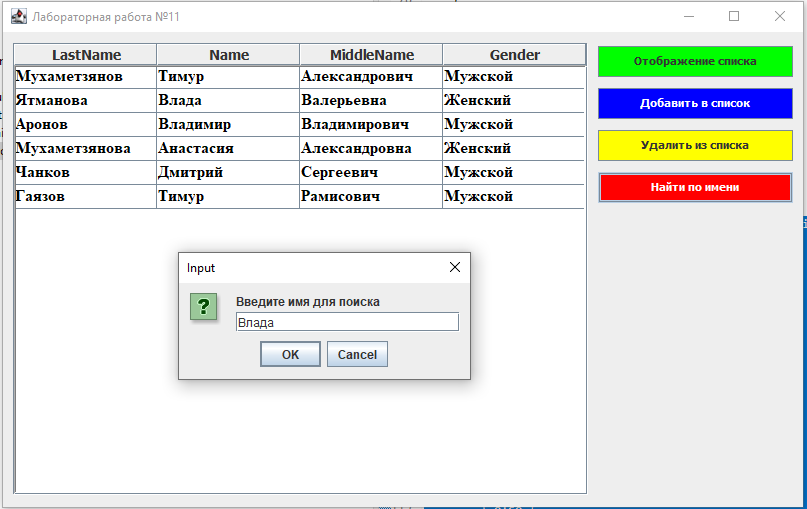


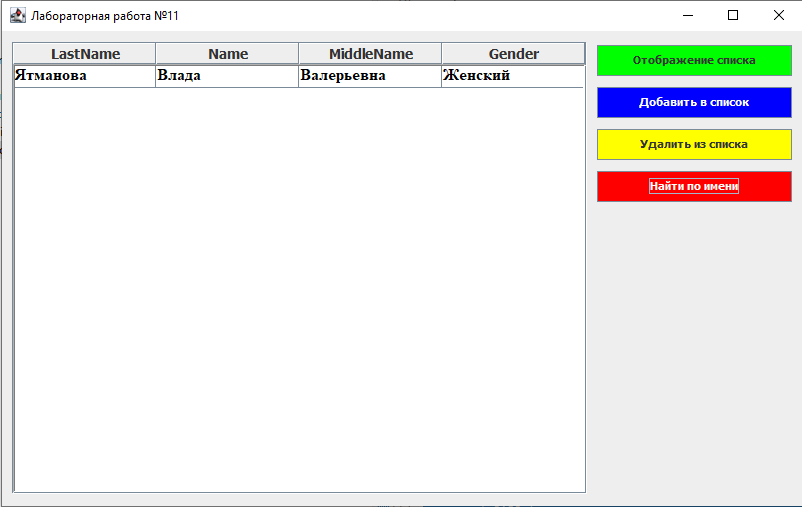


**Рис 2 – 6. Добавление в список.**



**Рис – 7. Удаление из списка.**





**Рис 8 – 9. Поиск по имени.**

**Выводы о проделанной работе:**

Я научился работать с односвязными и двухсвязными списками на Java, а также создавать для работы с ними пользовательский интерфейс.