重 庆 大 学

学 生 实 验 报 告

实验课程名称 JAVA EE 程序设计

开课实验室 DS1501

学 院 软件学院 年级 2021 专业班 软件工程 X 班

学 生 姓 名 XXX 学 号 2021XXXX

开 课 时 间 2022 至 2023 学年第 2 学期

成 绩	
教师签名	

大数据与软件学院制

《JAVA EE 程序设计》实验报告

开课实验室：DS1501 **2023 年 3 月 19 日**

学院	大数据与软件学院	年级、专业、班	2021 级软件工程 X 班	姓名	XXX	成绩	
课程名称	**JAVA EE 程序设计**	实验项目名称	实验5-6：代码调试及数据结构实现	指导教师	XX		
教师评语							

一、 实验目的

实验五：实现理论课上讲述的内容 PPT 内容的验证。

实验六：掌握利用JAVA完成数据结构课程中的内容。

选择：链表、栈、队列或二叉树等典型的数据结构其中的一个，实现其代码封装。实现该数据结构的基本方法：提取元素、插入元素、删除一个元素、找匹配某元素的个数等。具体内容参考《数据结构》课程内容介绍（在实验报告中增加该部分内容）。

二、实验内容

实验五内容：

调试，验证，课件ppt内容，课后布置的作业（实验报告内容中可不写）。

完成教材关于抽象类与接口方面的范例内容的代码验证、后面作业的代码设计（实验报告中可不写）。

实验六内容：

选择：链表、栈、队列或二叉树等典型的数据结构其中的一个，实现其代码封装。实现该数据结构的基本方法：提取元素、插入元素、删除一个元素、找匹配某元素的个数等。具体内容参考《数据结构》课程内容介绍（在实验报告中增加该部分内容）。

三、使用仪器、材料

IntelliJ IDEA 2022.1.3, Java SE 19

四、实验过程原始记录(数据、图表、计算等)：

实验六：

链表的实现：

源代码：

Node.Java（结点类）

```java
package experiment6;

public class Node {
    Node next=null;
    int element;
    public Node(int e){
        element=e;
    }
```

LList.Java（链表类）

```java
package experiment6;

public class LList {
    Node head=null;

    // 返回链表的长度
    public int length() {
        int thelen = 0;
        Node temp = head;
        while (temp != null) {
            thelen++;
            temp = temp.next;
        }
        return thelen;
    }

    // 遍历链表并将元素打印到控制台
    public void display() {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.element + " ");
            temp = temp.next;
        }
        System.out.print('\n');
    }
```

```java
    // 在链表末尾插入一个新元素
    public void insert(int v) {
        Node new_node = new Node(v);
        if (head == null) {
            head = new_node;
            return;
        }
        Node temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = new_node;
    }

    // 删除链表中的第l个元素
    public void delete(int l) {
        Node nn = head;
        for (int i = 0; i <= l - 3; i++) {
            nn = nn.next;
        }
        Node curr = nn.next;
        nn.next = curr.next;
    }

    // 查找链表中值为v的元素并返回其下标
    public int lookfor(int v) {
        int index = 0;
        Node temp = head;
        while (temp != null) {
            if (temp.element == v) {
                return index;
            }
            index++;
            temp = temp.next;
        }
        return -1;
    }

    // 统计链表中值为v的元素的个数
    public int count(int v) {
        Node temp = head;
        int cnt = 0;
        while (temp != null) {
            if (temp.element == v) {
                cnt++;
            }
            temp = temp.next;
        }
        return cnt;
    }
}
```

LListTest.Java（测试程序）

```java
package experiment6;

public class LListTest {
    public static void main(String[] args) {
        LList list = new LList();

        // 插入元素
        list.insert( v: 2);
        list.insert( v: 0);
        list.insert( v: 2);
        list.insert( v: 1);
        list.insert( v: 5);
        list.insert( v: 1);
        list.insert( v: 7);
        list.insert( v: 8);

        // 打印链表
        System.out.println("链表为: ");
        list.display();

        // 删除元素
        list.delete( i: 2);

        // 打印删除元素后的链表
        System.out.println("删除掉第2个元素后链表为: ");
        list.display();

        // 查找特定元素位置
        int index = list.lookfor( v: 7);
        System.out.println("链表中为7的元素下标为: " + index);

        // 统计元素个数
        int count = list.count( v: 2);
        System.out.println("链表中元素为2的个数为: " + count);

        // 打印链表长度
        System.out.println("链表长度为: " + list.length());
    }
}
```

运行结果：

```
F:\CQU\Java\JDK\bin\java.exe "
链表为:
2 0 2 1 5 1 7 8
删除掉第2个元素后链表为:
2 2 1 5 1 7 8
链表中为7的元素下标为: 5
链表中元素为2的个数为: 2
链表长度为: 7

进程已结束,退出代码0
```