实 验 报 告

课程名称	数据结构（Java）
实验项目	实验二 栈和队列的基本操作与应用

学　　院	计算机学院
系　　别	软件工程
班级/学号	软工 2101/2021011561
学 生 姓 名	陈志浩
实 验 日 期	2022/10/30
成　　绩	
指 导 教 师	蒋玉茹

实验二 栈和队列的基本操作与应用

一、实验目的

1．自定义栈和队列数据结构

2．在实际问题中应用栈和队列数据结构

二、实验内容及要求

（1） 实现栈的顺序或链式存储结构的基本操作，包括初始化栈、清空栈、入栈、出栈、取栈顶元素等。

（2） 实现队列的顺序或链式存储结构的基本操作，包括初始化队列、清空队列、入队、出队、取队头元素等。

三、系统设计

1.概述

本实验一共设计了 3 个类，2 个接口列表展示：

- 类 1 Student 类
- 类 2 SeqStack 类
- 类 3 SeqQueue 类
- 接口 1 SSstack
- 接口 2 Queue

2.类 1 Student 类的描述

属性及含义：

Student 类有两个属性分别为 code、score；分别表示学生的学号和分数

方法及功能：

Student 类有以下的方法：

int getCode() // 返回学生的学号

void setCode(int code) // 设置学生的学号

int getScore() // 返回学生的分数

void setScore(int score) // 设置学生的分数

String toString() // 重写 toString 方法方便打印

3.类 2 SeqStack 类的描述

属性及含义：

SeqStack 类有两个属性分别为 value、top,分别用于存储数据和指向栈顶

方法及功能：

boolean isEmpty() // 用于判断栈是否为空

void push(T item) // 将元素压入栈中

T pop() // 弹出栈顶元素

T peek() // 取出栈顶元素，但不弹出

boolean clean() // 清空栈

四、系统环境

1.操作系统及版本

Windows 10 家庭中文版

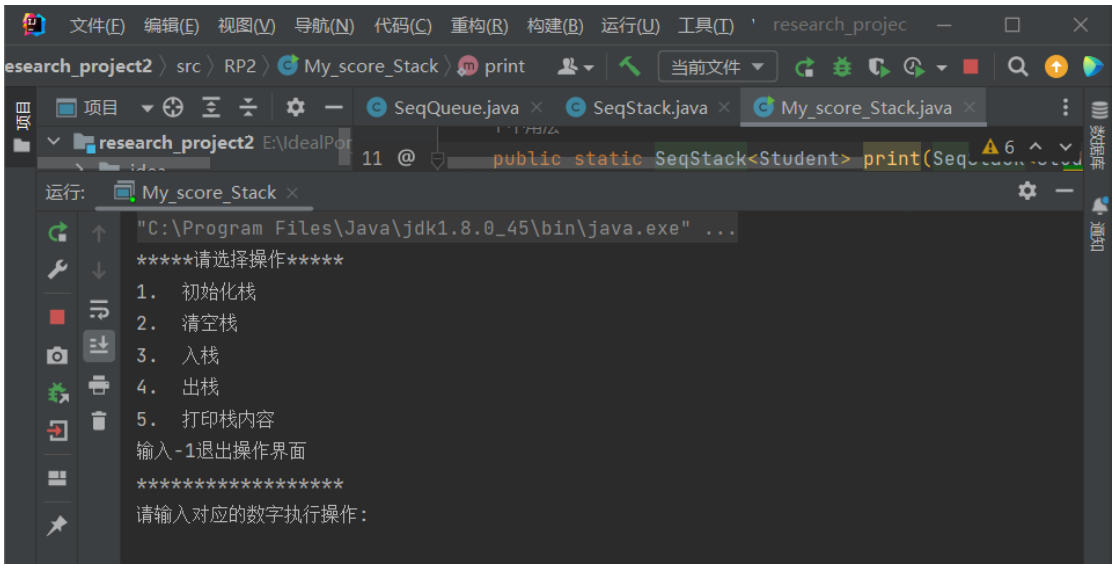
2.JDK 及版本

java version “1.8.0_15”

3.其他 JAR 包等 无

五、程序运行过程

【步骤 1】若想使用栈进行存储，需要运行 My_score_Stack.java 文件



【步骤 2】初始化栈

初始化栈需输入 1

```
请输入对应的数字执行操作:
1
**********栈初始化完成! **********
```

【步骤三】入栈

输入数字 3 后，需要输入学号和成绩进行信息录入

```
请输入对应的数字执行操作:
3
请输入学号和成绩以空格间隔
1 3
**********入栈完成! **********
```

【步骤四】出栈

输入数字 4 后，将弹出栈顶的元素

```
请输入对应的数字执行操作:
4
学号:1分数:3
**********出栈完成! **********
```

【步骤五】打印栈的内容

输入数字 5，打印栈中的元素

```
请输入对应的数字执行操作:
5
学号:5分数:10
学号:4分数:10
学号:3分数:10
学号:2分数:10
学号:1分数:10
```

【步骤六】清空栈

输入数字 2，清空栈

```
请输入对应的数字执行操作:
2
**********栈清空完成! **********
```

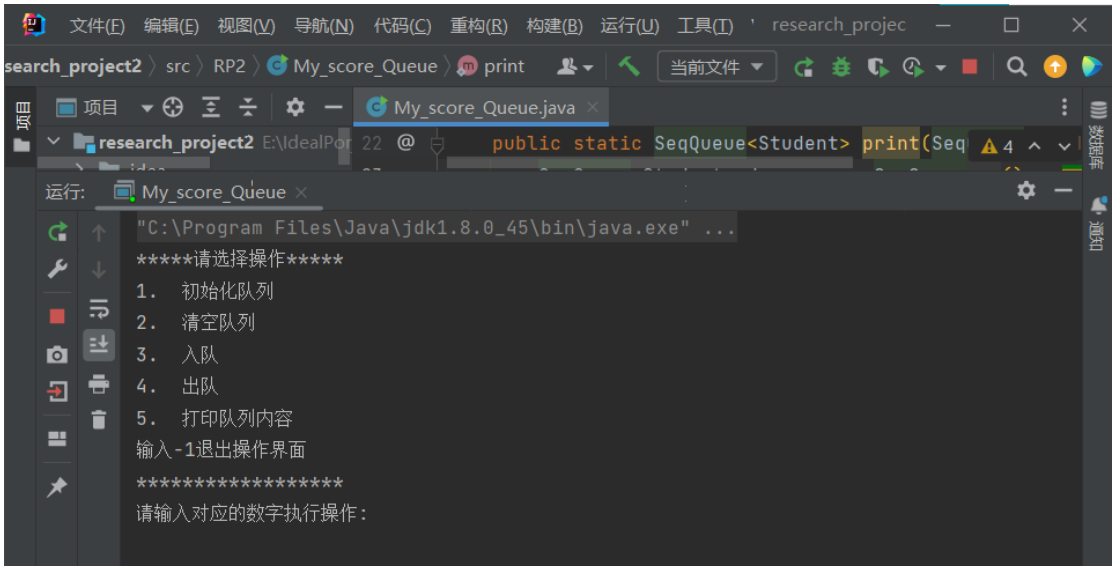
【步骤七】推出操作界面

输入-1 即可退出操作界面

```
请输入对应的数字执行操作:
-1
退出

进程已结束,退出代码0
```

【步骤 1】若想使用队列进行存储，需要运行 My_score_Queue.java 文件



【步骤 2】初始化队列

初始化队列需输入 1

```
请输入对应的数字执行操作:
1
*********队列初始化完成! **********
```

【步骤三】入队列

输入数字 3 后，需要输入学号和成绩进行信息录入

```
请输入对应的数字执行操作:
3
请输入学号和成绩以空格间隔
3 10
**********入队完成! **********
```

【步骤四】出队列

输入数字 4 后，队头的元素讲出队

```
请输入对应的数字执行操作：
4
学号:1分数:10
**********出队完成！**********
```

【步骤五】打印队列中的内容

输入数字 5，打印队列中的元素

```
请输入对应的数字执行操作：
5
学号:2分数:10
学号:3分数:10
学号:1分数:10
学号:4分数:10
学号:5分数:10
学号:6分数:10
```

【步骤六】清空队列

输入数字 2，清空队列

```
请输入对应的数字执行操作：
2
*********队列清空完成！**********
```

【步骤七】推出操作界面

输入-1 即可退出操作界面

```
请输入对应的数字执行操作：
-1
退出

进程已结束,退出代码0
```

六、编写程序中遇到的问题总结

1.将出栈用于打印栈中的元素

在使用打印方法时，我使用出栈的方法打印栈中的内容，导致打印完成后栈中的内容全没有了。在意识到这个问题后我在 My_score_Stack.java 中添加了 print 方法，使用一个临时的栈存储弹出的元素，打印完成后再存入原栈中。

七、参考文献

1.https://www.runoob.com/java/java-scanner-class.html

八、源代码

类：My_score_Queue

```
package RP2;

import SeqQueue.SeqQueue;
import Stack.SeqStack;

import java.util.Scanner;

public class My_score_Queue {
    public static Student build(){
        return getStudent();
    }

    static Student getStudent() {
        System.out.println("请输入学号和成绩以空格间隔");
        Scanner sc = new Scanner(System.in);
        String word = sc.nextLine();
```

```
        String[] word_list = word.split(" ");
        int number = Integer.parseInt(word_list[0]);
        int score = Integer.parseInt(word_list[1]);
        return new Student(number, score);
    }
    public static SeqQueue<Student> print(SeqQueue<Student> stu_que){
        SeqQueue<Student> demo = new SeqQueue<Student>();
        while(!stu_que.isEmpty()){
            Student temp = stu_que.poll();
            demo.add(temp);
            System.out.println(temp);
        }
        while(!demo.isEmpty()){
            Student temp1 = demo.poll();
            stu_que.add(temp1);
        }
        return stu_que;
    }

    public static void main(String [] args){
        SeqQueue<Student> stu_que = new SeqQueue<Student>();
        int a = -2;
        while(a!=-1){
            System.out.println("*****请选择操作*****\n" +
                    "1.\t 初始化队列\n" +
                    "2.\t 清空队列\n" +
                    "3.\t 入队\n" +
                    "4.\t 出队\n" +
                    "5.\t 打印队列内容\n" + "输入-1 退出操作界面");
            System.out.println("********************");
            System.out.println("请输入对应的数字执行操作:");
            Scanner num = new Scanner(System.in);
            a = num.nextInt();
            int flag = 0;
            switch (a){
                case 1:
                    System.out.println("**********队列初始化完成！**********\n");
                    break;
```

```
                    case 2:
                        if(stu_que.clear()){
                            System.out.println("**********队列清空完成！**********\n");
                        }
                        break;
                    case 3:
                        stu_que.add(build());
                        System.out.println("**********入队完成！**********\n");
                        break;
                    case 4:
                        System.out.println(stu_que.poll());
                        System.out.println("**********出队完成！**********\n");
                        break;
                    case 5:
                        print(stu_que);
                        break;
                    case -1:
                        a = -1;
                        System.out.println("退出");
                        break;
                }
            }
        }
}
```

类：My_score_Stack

```
package RP2;

import SeqQueue.SeqQueue;
import Stack.SeqStack;

import java.util.Scanner;

import static RP2.My_score_Queue.getStudent;

public class My_score_Stack {
```

```
public static SeqStack<Student> print(SeqStack<Student> stu_stack_ar){
    SeqStack<Student> demo = new SeqStack<Student>();
    while(!stu_stack_ar.isEmpty()){
        Student temp = stu_stack_ar.pop();
        demo.push(temp);
        System.out.println(temp);
    }
    while(!demo.isEmpty()){
        Student temp1 = demo.pop();
        stu_stack_ar.push(temp1);
    }
    return stu_stack_ar;
}
public static Student build(){
    return getStudent();
}
public static void main(String [] args){
    SeqQueue<Student> stu_que = new SeqQueue<Student>();
    SeqStack<Student> stu_stack_ar = new SeqStack<Student>();
    int a = -2;
    while(a!=-1){
        System.out.println("*****请选择操作*****\n" +
                           "1.\t 初始化栈\n" +
                           "2.\t 清空栈\n" +
                           "3.\t 入栈\n" +
                           "4.\t 出栈\n" +
                           "5.\t 打印栈内容\n" + "输入-1 退出操作界面");
        System.out.println("******************");
        System.out.println("请输入对应的数字执行操作:");
        Scanner num = new Scanner(System.in);
        a = num.nextInt();
        int flag = 0;
        switch (a){
            case 1:
                System.out.println("**********栈初始化完成! **********\n");
                break;
            case 2:
                if(stu_stack_ar.clean()){
```

```
                        System.out.println("**********  栈  清  空  完  成  ！
**********\n");
                    }
                    break;
                case 3:
                    stu_stack_ar.push(build());
                    System.out.println("**********入栈完成！**********\n");
                    break;
                case 4:
                    System.out.println(stu_stack_ar.pop());
                    System.out.println("**********出栈完成！**********\n");
                    break;
                case 5:
                    print(stu_stack_ar);
                    break;
                case -1:
                    a = -1;
                    System.out.println("退出");
                    break;
            }
        }
    }
}
```

类：Student

```
package RP2;

public class Student {
    private int code;
    private int score;

    public Student(int code, int score) {
        this.code = code;
        this.score = score;
    }
```

```
    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public int getScore() {
        return score;
    }

    public void setScore(int score) {
        this.score = score;
    }

    @Override
    public String toString() {
        return "学号:" + code + "分数:" + score;
    }
}
```

类：SeqQueue

```
package SeqQueue;


public class SeqQueue<T> implements Queue<T>{
    private Node<T> head;
    private Node<T> tail;
    public SeqQueue(){
        this.head = null;
        this.tail = null;
    }

    @Override
    public boolean isEmpty() {
```

```
        return this.head == null;
    }

    @Override
    public T add(T item) {
        if(item==null){
            return null;
        }
        Node<T> temp = new Node<>(item, null);
        if(this.head==null){
            this.head = temp; // 当队列为空时
            this.tail = temp;
        }else{
            this.tail.next = temp; // 当队列不为空时
            this.tail = this.tail.next;
        }
        return this.tail.data;
    }

    @Override
    public T poll() {
        if(isEmpty()){
            return null;
        }
        T temp = this.head.data;
        this.head = this.head.next;
        if(this.head==null){ // 当原队列中只有一个元素时 poll 后 此时 head 指向 null
此时需要将 tail 也指向 null
            this.tail = null;
        }
        return temp;
    }

    @Override
    public T peek() {
        if(isEmpty()){
            return null;
        }
```

```
        return this.head.data;
    }

    @Override
    public boolean clear() {
        this.head = null;
        this.tail = null;
        return true;
    }
}
```

类：SeqStack

```
package Stack;

public class SeqStack<T> implements SSstack<T>{
    private Object value[];
    private int top;

    public SeqStack(int size){
        this.value = new Object[size];
        this.top = 0;
    }
    public SeqStack(){
        this(32);
    }

    @Override
    public boolean isEmpty() {
        return this.top==0;
    }

    @Override
    public void push(T item) {
        if(item == null){
            return ;
        }
```

```
        if(this.top == this.value.length){
            Object [] temp = this.value;
            this.value = new Object[temp.length*2];
            for(int i = 0; i<temp.length; i++){
                this.value[i] = temp[i];
            }
        }
        this.value[top] = item;
        this.top++;
    }

    @Override
    public T pop() {
        T a;
        if(!isEmpty()) {
            a = (T)this.value[this.top-1];
            this.top--;
        }else{
            a = null;
        }
        return a;
    }

    @Override
    public T peek() {
        if(!isEmpty()) {
            return (T)this.value[this.top-1];
        }else{
            return null;
        }
    }

    @Override
    public boolean clean() {
        this.top = 0;
        return true;
    }
```

```
}
```

接口：Queue

```
package SeqQueue;

public interface Queue<T> {
    boolean isEmpty();
    T add(T item); // 向队列中添加元素
    T poll(); // 出队 返回队头的元素
    T peek(); // 取出队头的元素但不删除

    boolean clear(); // 清空队列
}
```

接口：SSstack

```
package Stack;

public interface SSstack<T>{
    boolean isEmpty(); // 判断栈是否为空
    void push(T item); // 压栈
    T pop(); // 将栈顶的元素弹出 并删除
    T peek(); // 取栈顶的元素并返回，但不删除
    boolean clean(); // 清空栈
}
```