队列

1.基本介绍

2.数组模拟队列

1.特点

2.代码实现:

3.数组模拟环形队列

队列

*应用场景：有序排队，银行排队

1.基本介绍

1. 队列是一个有序列表，可以用数组或者链表
2. 遵循先入先出原则

2.数组模拟队列

1.特点

队列本身是有序列表，若用数组来储存队列的数据，则队列数组需要声明一个对象，有front和rear来分别记录队列头和队列尾，以及maxSize来记录队列的最大容量

front随着数据输出而改变 front指向队列最前元素前面（不含第一个元素）

rear随着数据输入而改变 rear是队列最后元素（含）

空队列是front=rear

队列满是rear=maxSize-1

2.代码实现:

```
package 线性结构.队列Queue;

/**
 * @author jndeng
 * @create 2019-11-05 20:40
 */
```

```
public class ArrayQueue {
    private final int maxSize;
    private int front;
    private int rear;
    private int[] array;

    public ArrayQueue(int maxSize) {
        this.maxSize = maxSize;
        this.front = -1;
        this.rear = -1;
        array = new int[maxSize];
    }

    public void addQueue(int num) {
        if (front == maxSize - 1) {
            System.out.println("队列已满，不能添加");
            return;
        }
        rear++;
        array[rear] = num;
    }

    public int getQueue(int num) {
        if (front == rear) {
            throw new RuntimeException("空队列");
        }
        front++;
        return array[front];
    }
    /**
     * 显示队列全部数据（无论是否已经出队列）
     */
    public void showQueueAllData() {
        if (front == rear) {
            System.out.println("空队列，没有数据");
            return;
        }
        for (int i=0;i<array.length;i++) {
            System.out.println(i+" ");
        }
    }
}
```

3.数组模拟环形队列

1.特点：

数据被取出后，能够重复使用存储空间

2.改进思路：

1.改变front定义：front指向第一个元素（含） 初始值=0

2.改变rear定义：rear指向最后一个元素的后一个位置（不含最后 一个元素）初始值=0

这里 整个数据预留了一个空间，不存储数据，为了能够进行环形转换

实际的maxSize比用户输入的maxSize大1

当 （rear+1）%maxSize=front

当rear==front 队列空

3.有效数据个数 ：(rear+maxSize-front)%maxSize

3.代码实现

```
package 线性结构.队列Queue;

import java.util.Scanner;

/**
 * @author jndeng
 * @create 2019-11-05 21:34
 */
public class CircleArrayQueue {
    private final int maxSize;
    private int front;
    private int rear;
    private int[] array;

    public CircleArrayQueue(int maxSize) {
        this.maxSize = maxSize;
        this.front = 0;
        this.rear = 0;
        array = new int[maxSize];
    }

    public boolean isFull() {
        return (rear+1) % maxSize == front;
    }
    public boolean isEmpty() {
        return  front==rear;
    }
```

```
public void addQueue(int num) {
    if (isFull()) {
        System.out.println("队列已满，不能添加");
        return;
    }
    array[rear]=num;
    rear=(rear+1)%maxSize;
}

// 获取队列的数据, 出队列
public int getQueue() {
    // 判断队列是否空
    if (isEmpty()) {
        // 通过抛出异常
        throw new RuntimeException("队列空，不能取数据");
    }
    int value = array[front];
    front = (front + 1) % maxSize;
    return value;

}



// 显示队列的所有数据
public void showQueue() {
    // 遍历
    if (isEmpty()) {
        System.out.println("队列空的，没有数据~~");
        return;
    }
    // 思路：从front开始遍历，遍历多少个元素
    for (int i = front; i < front + size() ; i++) {
        System.out.printf("arr[%d]=%d\n", i % maxSize, array[i % maxSize]);
    }
}

// 求出当前队列有效数据的个数
public int size() {
    // rear = 2
    // front = 1
    // maxSize = 3
    return (rear + maxSize - front) % maxSize;
}

// 显示队列的头数据， 注意不是取出数据
public int headQueue() {
```

```
        // 判断
        if (isEmpty()) {
            throw new RuntimeException("队列空的，没有数据~~");
        }
        return array[front];
    }

    public static void main(String[] args) {

        //测试一把
        System.out.println("测试数组模拟环形队列的案例~~~");

        // 创建一个环形队列
        CircleArrayQueue queue = new CircleArrayQueue(3); //说明设置4, 其队列的有效数据最大是3
        char key = ' '; // 接收用户输入
        Scanner scanner = new Scanner(System.in);//
        boolean loop = true;
        // 输出一个菜单
        while (loop) {
            System.out.println("s(show): 显示队列");
            System.out.println("e(exit): 退出程序");
            System.out.println("a(add): 添加数据到队列");
            System.out.println("g(get): 从队列取出数据");
            System.out.println("h(head): 查看队列头的数据");
            key = scanner.next().charAt(0);// 接收一个字符
            switch (key) {
                case 's':
                    queue.showQueue();
                    break;
                case 'a':
                    System.out.println("输出一个数");
                    int value = scanner.nextInt();
                    queue.addQueue(value);
                    break;
                case 'g': // 取出数据
                    try {
                        int res = queue.getQueue();
                        System.out.printf("取出的数据是%d\n", res);
                    } catch (Exception e) {
                        // TODO: handle exception
                        System.out.println(e.getMessage());
                    }
                    break;
                case 'h': // 查看队列头的数据
                    try {
                        int res = queue.headQueue();
```

```
                System.out.printf("队列头的数据是%d\n", res);
            } catch (Exception e) {
                // TODO: handle exception
                System.out.println(e.getMessage());
            }
            break;
        case 'e': // 退出
            scanner.close();
            loop = false;
            break;
        default:
            break;
      }
    }
    System.out.println("程序退出~~");
  }
}
```