# 栈和队列

## 栈

栈的特点

先进后出：如同一个桶：东西放进桶里(只有一个出口)，先放进去的在最里面，后放进去的在最上面

堆栈在代码中的实现

1、栈的底层实现是一个数组

2、维护着栈顶数据变量

3、从栈中拿出的数据，相应着这个堆栈就会消失，如桶里有苹果，拿出来了桶里肯定就会少了这个苹果了

MyStack类

public class MyStack {

private long[] arr; //long类型的数组

private int top; //栈顶的索引

//默认构造方法

public MyStack() {

arr=new long[10];

top=-1;

}

//带参数构造方法：参数为数组初始化大小

public MyStack(int maxsite) {

arr=new long[maxsite];

top=-1;

}

//添加数据

public void push(int value) {

arr[++top]=value;

}

//移除数据：用于循环来获取数据，相应着这个堆栈就会少数据

public long pop() {

return arr[top--];

}

//查看数据:栈顶上的数据

public long peek() {

return arr[top];

}

//判断是否为空:只要看栈顶变量top是否为-1

public boolean isEmpty() {

return top == -1;

}

//判断是否满了:栈顶变量top是否等于arr.length-1(数组：a[10]那就是10个，索引从0开始算到9)

public boolean isFull() {

return top==arr.length - 1;

}

}

测试代码

public static void main(String[] args) {

MyStack ms= new MyStack(3);

ms.push(1);

ms.push(2);

ms.push(3);

//判断堆栈是否为空的方法

System.out.println(ms.isEmpty()); //打印false

//判断堆栈是否已满的方法

System.out.println(ms.isFull()); //打印true

//查看栈顶数据

System.out.println(ms.peek()); //打印3

//把数据拿出来：使用MyStack方法中的移除数据方法来获取数据

while(!ms.isEmpty()) {

System.out.println(ms.pop()); //先进后出原则：打印：3，2，1

}

//再次判断堆栈是否为空的 这时候就打印 true 上面已经全部拿出来了

System.out.println(ms.isEmpty());//true

//再次判断堆栈是否已满的方法

System.out.println(ms.isFull());//false

}

## 队列

队列特点

先进先出：如生活中的买票，先排队先买票；生活中的水管：先进来的水，水管出口中先出

堆栈在代码中的实现

1、列队的底层实现是一个数组

2、维护着有3个：有效数据的大小、队头、队尾

3、从队头依次拿出(移除)数据，相应的队列的也会相应清除，

MyQueue类

/\*\*

\* 列队类

\* @author Administrator

\*

\*/

public class MyQueue {

//底层使用数组

private long[] arr;

//有效数据的大小

private int elements;

//队头

private int front;

//队尾

private int end;

/\*\*

\* 默认构造方法

\*/

public MyQueue() {

arr = new long[10];

elements = 0;

front = 0;

end = -1;

}

/\*\*

\* 带参数的构造方法，参数为数组的大小

\*/

public MyQueue(int maxindex) {

arr = new long[maxindex];

elements = 0;

front = 0;

end = -1;

}

/\*\*

\* 添加数据：从队尾开始添加，每次排队都要队尾中进行排队

\*/

public void insert(long value) {

arr[++end]=value;

elements++;

}

/\*\*

\* 删除数据：从队头删除 ：排队买完票：就要从队列离开

\*/

public long reomve() {

elements--;

return arr[front++];

}

/\*\*

\* 查看数据：查看队头数据

\*/

public long peek() {

return arr[front];

}

/\*\*

\* 判断是否为空

\*/

public boolean isEmpty() {

return elements == 0;

}

/\*\*

\* 判断是否已满

\*/

public boolean isFull() {

return elements == arr.length;

}

}

测试代码

public static void main(String[] args) {

MyQueue mq = new MyQueue(3);

mq.insert(1);

mq.insert(2);

mq.insert(3);

System.out.println(mq.isEmpty());

System.out.println(mq.isFull());

//查看 队头数据

System.out.println(mq.peek());

//循环取出队列的数据

while(!mq.isEmpty()) {

System.out.println(mq.reomve());

}

//再次判断堆栈是否为空的

System.out.println(mq.isEmpty());//true

//再次判断堆栈是否已满的

System.out.println(mq.isFull());//false

mq.insert(3);

//注意这里再次插入的时候由于队头和队列都没有处理，只能使用一次队列，下面再进行插入会报错，不能循环使用，可以优化

}