实验报告三 栈和队列

班级： 2018053 姓名： 洪勇 学号： 20181389 专业： 计算机科学与技术

1. 实验目的：
2. 掌握栈的基本操作的实现方法。
3. 利用栈先进后出的特点，解决一些实际问题。
4. 掌握链式队列及循环队列的基本操作算法。
5. 应用队列先进先出的特点，解决一些实际问题。
6. 实验内容：
7. 使用一个栈，将一个十进制转换成二进制。

粘贴源程序：

Seqlist类：

package 数据结构JAVA.实验三.实验内容二;

public class Seqlist<T> extends Object {

protected Object[] element;

protected int lenth;

public Seqlist(int lenth) {

this.element=new Object[lenth];

this.lenth=0;

}

public Seqlist() {

this(1024);

}

public Seqlist(T[] values) {

this(values.length);

for(int i=0;i<values.length;i++)

this.element[i]=values[i];

this.lenth=values.length;

}

public boolean isEmpty() {

return this.lenth==0;

}

public int size() {

return this.lenth;

}

public T get(int i) {

if(i<this.lenth&&i>=0)

return (T)this.element[i];

return null;

}

public void set(int i,T x) {

if(x==null)

throw new NullPointerException("x==null");

if(i>=0&&i<this.lenth)

this.element[i]=x;

else throw new java.lang.IndexOutOfBoundsException(i+"");

}

public String toString() {

String str=this.getClass().getName()+"(";

for(int i=0;i<this.lenth;i++) {

str+=this.element[i].toString();

if(i!=this.lenth)

str+=",";

}

return str+")";

}

public void expose() {

Object temp[]=new Object[this.lenth];

for(int i=0;i<this.lenth;i++) {

temp[i]=this.element[i];

}

this.element=new Object[this.lenth\*2];

for(int i=0;i<this.lenth;i++) {

this.element[i]=temp[i];

}

}

public void insert(T x) {

if(this.element.length<=this.lenth)

this.expose();

this.element[this.lenth++]=x;

}

public int top() {

return this.lenth-1;

}

public T peekElement() {

if(this.lenth!=0)

return (T)this.element[this.lenth-1];

else

return null;

}

}

SeqStack类:

package 数据结构JAVA.实验三.实验内容二;

interface Stack<T>{

public abstract boolean isEmpty();

public abstract void push(T x);

public abstract T pop();

public abstract T peek();

}

public class SeqStack<T> implements Stack<T>{

private Seqlist<T> list;

public SeqStack(int length) {

list=new Seqlist<T>(length);

}

public SeqStack() {

this(1024);

}

public boolean isEmpty() {

return list.isEmpty();

}

public void push(T x) {

this.list.insert(x);

}

public T pop() {

T values=this.list.peekElement();

this.list.lenth--;

return values;

}

public T peek() {

return this.list.peekElement();

}

public int top() {

return this.list.lenth;

}

}

Demo类：

package 数据结构JAVA.实验三.实验内容一;

import 数据结构JAVA.实验三.实验内容二.\*;

import java.util.\*;

public class Demo {

public static void BinaryTransfrom(int target) {

SeqStack<Integer> TwoBinary=new SeqStack<Integer>();

while(target!=0) {

TwoBinary.push(target%2);

target/=2;

}

int len=TwoBinary.top();

for(int i=0;i<len;i++) {

System.*out*.print(TwoBinary.pop());

}

}

public static void main(String[] args) {

int number=0;

Scanner reader=new Scanner(System.*in*);

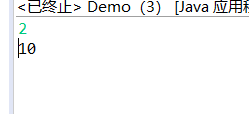
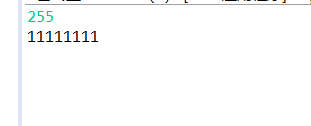
number=reader.nextInt();

*BinaryTransfrom*(number);

}

}

粘贴测试数据及运行结果：



2、回文是指正读反读均相同的字符序列，如“acdca”、“dceecd”均是回文，但“book”不是回文。利用1中的基本算法，试写一个算法判定给定的字符串是否为回文。（提示：将一半字符入栈，依次弹出与另一半逐个比较）

粘贴源程序：

Seqlist类：与上题相同

SeqStack类：与上题相同

Demo类：

package 数据结构JAVA.实验三.实验内容一;

import java.util.Scanner;

public class Demo {

public static boolean judgeString(String target) {

int len=target.length();

char[] temp=target.toCharArray();

SeqStack<Character> List=new SeqStack<Character>(len/2);

for(int i=0;i<len/2;i++) {

List.push(temp[i]);

}

for(int i=(len+1)/2;i<len;i++) {

if(temp[i]!=List.pop())

return false;

}

return true;

}

public static void main(String[] args) {

// TODO 自动生成的方法存根

String s1;

Scanner reader=new Scanner(System.*in*);

s1=reader.next();

if(*judgeString*(s1))

System.*out*.println("Yes");

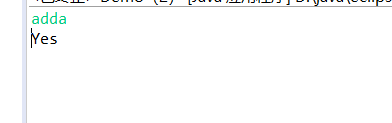
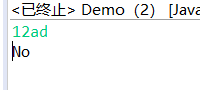
else

System.*out*.println("No");

}

}

粘贴测试数据及运行结果：



1. 使用3个队列分别保留手机上最近10个“未接来电（MissedCall）”、“已接来电(ReceivedCall)”、“已拨电话(CallBack)”。

输入：全部通话记录，每行一条记录，总长度小于10行。每条记录包含两个数字，第一个数代表记录类型，第二个数代表手机号码。按照电话发生的时间顺序，由前向后排列，最近一个电话在最后，以此类推。

处理：使用3个队列分别保留手机上最近10个“未接来电”、“已接来电”、“已拨电话”。

输出：分3个队列分别输出未接来电、已接来电、已拨电话。

粘贴源程序：

Seqlist类：同上题

SeqStack类：同上题

Call类：

package 数据结构JAVA.实验三.实验内容三;

import java.util.\*;

import 数据结构JAVA.实验三.实验内容二.\*;

public class Call {

protected SeqStack<String> CallNumber;

protected int len;

Scanner reader;

public Call() {

this.CallNumber=new SeqStack<String>(10);

this.len=0;

}

public void DisplayAllCallNumberAndClear() {

System.out.println("此次操作将展示所有电话记录并清除，请确认是否继续：");

reader=new Scanner(System.in);

boolean flag=false;

System.out.println("1：是 0：否");

flag=reader.hasNext();

if(flag) {

int len=this.CallNumber.top();

for(int i=0;i<len;i++) {

System.out.println((i+1)+". "+this.CallNumber.pop().toString());

}

}else {

System.out.println("操作已撤销。");

//reader.close();

return ;

}

return ;

}

public void DisplayPartOfNumberAndClear(int index) {

System.out.println("此操作将展示最近的"+index+"条记录并清除，请确认是否继续：");

reader=new Scanner(System.in);

boolean flag=false;

System.out.println("1：是 2：否");

flag=reader.hasNext();

if(flag) {

for(int i=0;i<index;i++) {

System.out.println((i+1)+". "+this.CallNumber.pop().toString());

}

}else {

System.out.println("操作已撤销。");

//reader.close();

return ;

}

}

}

AddCallNumber接口：

package 数据结构JAVA.实验三.实验内容三;

public interface AddCallNumber {

public abstract void addNumber();

public abstract void addNumber(MissedCall a);

}

MissedCall类：

package 数据结构JAVA.实验三.实验内容三;

import java.util.\*;

public class MissedCall extends Call implements AddCallNumber{

Scanner reader;

public void addNumber() {

reader=new Scanner(System.*in*);

System.*out*.println("您有未接来电！");

String number=reader.next();

this.CallNumber.push(number);

//reader.close();

}

public void addNumber(MissedCall a) {

}

}

ReceivedCall类：

package 数据结构JAVA.实验三.实验内容三;

import java.util.\*;

public class ReceivedCall extends Call implements AddCallNumber{

Scanner reader;

public void addNumber(MissedCall a) {

reader=new Scanner(System.*in*);

System.*out*.print("来电：");

String number=reader.next();

System.*out*.println("是否接听？");

int flag=0;

System.*out*.println("1. 是 0. 否");

flag=reader.nextInt();

if(flag==1) {

this.CallNumber.push(number);

//reader.close();

}else {

a.CallNumber.push(number);

//reader.close();

}

}

public void addNumber() {

}

}

CallBack类：

package 数据结构JAVA.实验三.实验内容三;

import java.util.\*;

public class CallBack extends Call implements AddCallNumber{

Scanner reader;

public void addNumber() {

reader=new Scanner(System.*in*);

System.*out*.println("回拨号码：");

String number=reader.next();

this.CallNumber.push(number);

//reader.close();

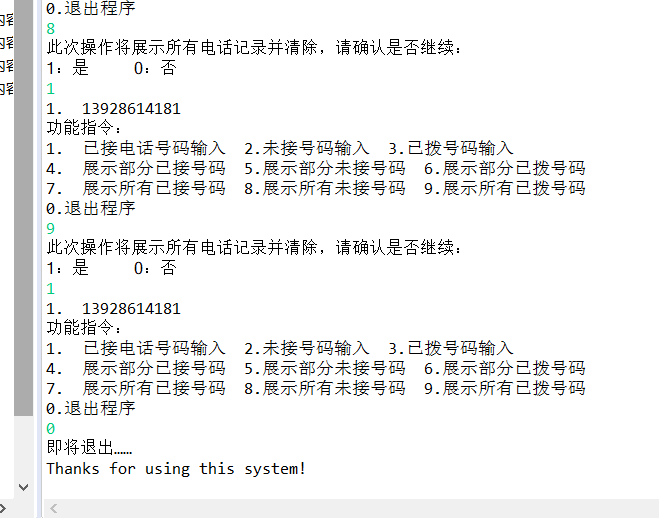
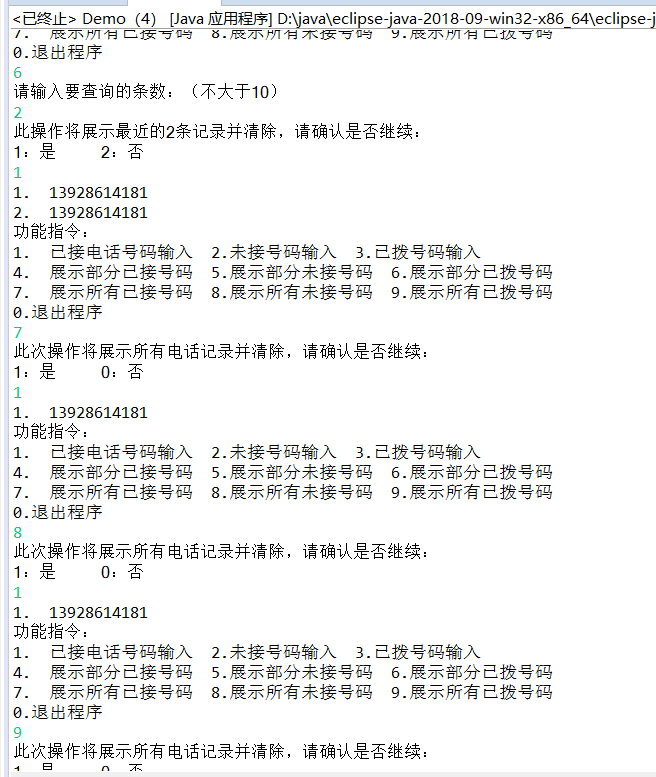
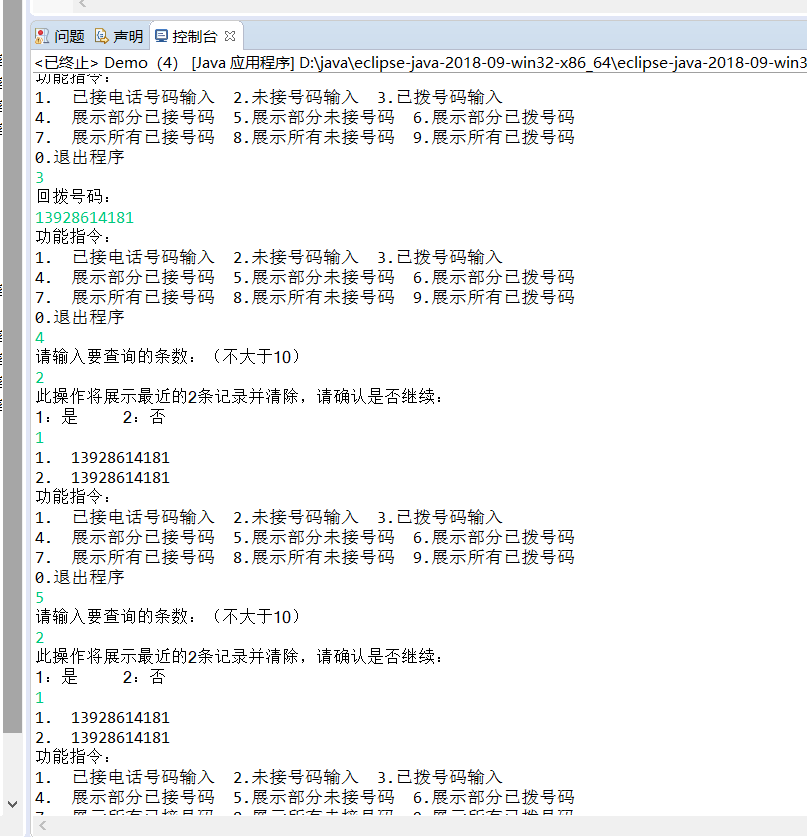
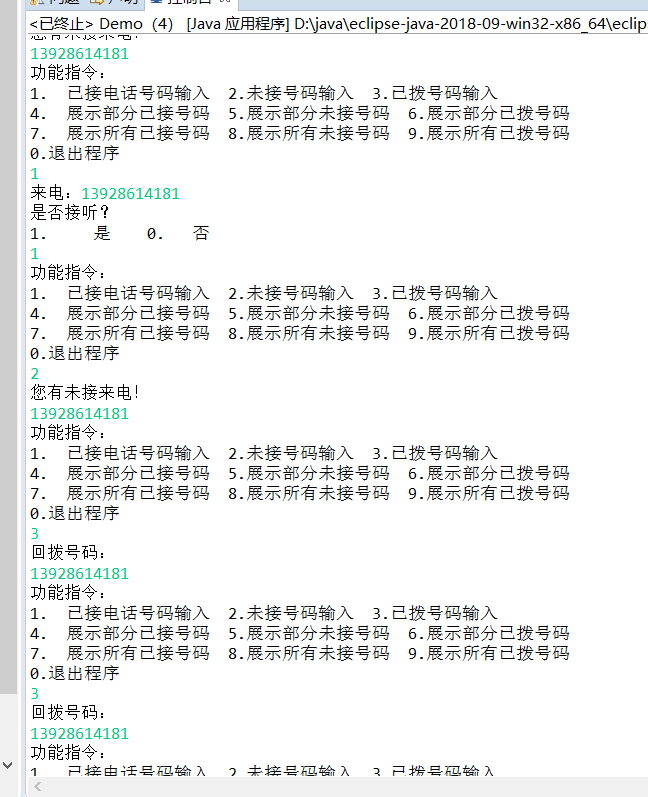
}

public void addNumber(MissedCall a) {

}

}

粘贴测试数据及运行结果：



1. 心得体会：（含上机中所遇问题的解决办法，所使用到的编程技巧、创新点及编程的心得）
2. 使用泛型类型，使数据类型用起来很方便。
3. 用了接口抽象，父类继承，实现多态性。
4. 堆栈在一些特定条件非常好用。
5. Scanner类的关闭要慎重，不要乱关，会导致线程关闭无法使用。
6. 对于电话号码这种数据，用string比用long要方便。