```java
package oop2_dat2_u;

/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class Main
{

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        // Aufgabe 1
        System.out.println("##### AUFGABE 1 #####");
        // Stack initialisieren
        StackArrayList myStack = new StackArrayList<String>();

        /* Stack Test mit "manuellem" pop
        myStack.push("Hallo Du");
        myStack.push("Das ist ein Test");
        myStack.push("LIFO, ich bin das 3. Element und werde als erstes ausgeben...");

        System.out.println("POP #1: " + myStack.pop());
        System.out.println("POP #2: " + myStack.pop());
        System.out.println("POP #3: " + myStack.pop());
        if ( ! myStack.isEmpty()) {
        System.out.println("POP #4: " + myStack.pop());
        }
         */

        // Stack neue Elemente einfügen und mittels while ausgeben
        myStack.push("Das");
        myStack.push("ist");
        myStack.push("ein");
        myStack.push("Test");
        myStack.push(";-)");
        // Stack ausgeben
        int i = 0;
        while ( ! myStack.isEmpty()) {
            i ++;
            System.out.println("POP #" + i + ": " + myStack.pop());
        }


        // Aufgabe 2
        System.out.println("\n\n\n##### AUFGABE 2 #####");
        // Queue / Ringbuffer Test
        Ringbuffer myRing = new Ringbuffer(10);
        myRing.enqueue("Queue Test");
        myRing.enqueue("Bla bla bla");
        myRing.enqueue("Test ende...");

        // Queue ausgeben
        while (myRing.getNumberElements() > 0) {
            System.out.println("Queue: " + myRing.dequeue());
        }


        // Aufgabe 3
        System.out.println("\n\n\n##### AUFGABE 3 #####");
        // Doppeltverkette Liste
        MyLinkedList<String> stringList;
        stringList = new MyLinkedList<String>();

        stringList.insert("Die Ersten werden die Letzten sein ;-)");
        stringList.insert("zwei zwei");
        stringList.insert("333");
        stringList.insert("Die Letzten werden die Ersten sein :D");

        System.out.println("> In der Liste sind " + stringList.size() + " Elemente.");

        System.out.println("\n> stringList ausgeben:");
        stringList.print();

        System.out.println("\n> stringList rückwärts ausgeben:");
        stringList.printReverse();

        System.out.println("\n> Gibt es das Element '333'? " + stringList.isFound("333"));
        System.out.println("> Den String '333' entfernen...");
        stringList.remove("333");
        System.out.println("> Gibt es das Element '333'? " + stringList.isFound("333"));
        System.out.println("> stringList ausgeben:");
        stringList.print();
    }
}
```

```java
package oop2_dat2_u;

/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class MyLinkedList<Object>
{

    private MyListNode<Object> head; // Kopf bzw. Anfang der Liste
    private MyListNode<Object> tail; // Schwanz bzw. Ende der Liste
    private int numOfElements; // Anzahl Elemente in dieser Liste

    public MyLinkedList()
    {
        head = null;
        tail = null;
        numOfElements = 0;
    }

    /**
     * Prüft, ob Liste leer ist.
     */
    public boolean isEmpty()
    {
        return (head == null);
    }

    /**
     * Fügt Objekt obj am Anfang in die Liste ein.
     */
    public void insert(Object obj)
    {
        // aktuelles Element zwischenspeichern
        MyListNode<Object> actualNode = head;

        // neues Objekt einfügen
        head = new MyListNode<Object>(head, obj);

        // Liste wieder zusammenhängen, head / tail updaten
        if (actualNode != null) {
            actualNode.setPrev(head);
        } else {
            tail = head;
        }
        numOfElements ++;
    }

    /**
     * Prüft, ob ein gleiches Objekt wie obj bereits in der Liste enthalten ist.
     */
    public boolean isFound(Object obj)
    {
        MyListNode<Object> actualNode = head;
        while ((actualNode != null) &&  ! obj.equals(actualNode.getData())) {
            actualNode = actualNode.getNext();
        }
        if (actualNode == null) {
            return false;
        } else {
            return true;
        }
    }

    /**
     * Entfernt aus der Liste das erste Objekt gleich obj.
     */
    public void remove(Object obj)
    {
        MyListNode<Object> actualNode = head;
        MyListNode<Object> prevNode = null;
        while ((actualNode != null) &&  ! obj.equals(actualNode.getData())) {
            prevNode = actualNode;
            actualNode = prevNode.getNext();
        }
        // Liste wieder ordentlich zusammenhängen:
        if (actualNode != null) {
            if (actualNode == head) {
                head = actualNode.getNext();
                head.setPrev(null);
            } else {
                prevNode.setNext(actualNode.getNext());
                actualNode.getNext().setPrev(prevNode);
            }
            numOfElements --;
        }
    }

    /**
     * Gibt alle Objekte der Reihe nach zeilenweise auf die Konsole aus.
     */
    public void print()
    {
        MyListNode<Object> actualNode = head;
        while (actualNode != null) {
            System.out.println(actualNode.getData());
            actualNode = actualNode.getNext();
        }
    }

    /**
```

```
102         * Gibt alle Objekte rückwärts der Reihe nach zeilenweise auf die Konsole aus.
103         */
104        public void printReverse()
105        {
106            MyListNode<Object> actualNode = tail;
107            while (actualNode != null) {
108                System.out.println(actualNode.getData());
109                actualNode = actualNode.getPrev();
110            }
111        }
112
113        public int size()
114        {
115            return numOfElements;
116        }
117    }
```

```java
package oop2_dat2_u;

/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class MyListNode<Object>
{

    private Object data;
    private MyListNode<Object> next;
    private MyListNode<Object> prev;

    public MyListNode(MyListNode<Object> newNext, Object obj)
    {
        data = obj;
        next = newNext;
        prev = null;

    }

    public void setData(Object obj)
    {
        data = obj;
    }

    public Object getData()
    {
        return data;
    }

    public void setNext(MyListNode<Object> nextElem)
    {
        next = nextElem;
    }

    public MyListNode<Object> getNext()
    {
        return next;
    }

    public void setPrev(MyListNode<Object> prevElem)
    {
        prev = prevElem;
    }

    public MyListNode<Object> getPrev()
    {
        return prev;
    }
}
```

```java
package oop2_dat2_u;

/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class Ringbuffer
{

    private int size;
    private int nbrElt = 0;
    private int in = 0;
    private int out = 0;
    private Object[] queue;

    public Ringbuffer(int s)
    {
        size = s;
        queue = new Object[size];
    }

    public void enqueue(Object o)
    {
        // Fügt x am Ende in die Warteschlange ein,
        // falls die Warteschlange nicht voll ist;
        nbrElt ++;
        if (in == size) {
            in = 0;
        }
        queue[in] = o;
        in ++;
    }

    public Object dequeue()
    {
        // Entfernt das erste Element aus der Warteschlange,
        // falls die Warteschlange nicht leer ist;
        nbrElt --;
        if (out == size) {
            out = 0;
        }
        Object o = queue[out];
        out ++;
        return o;
    }

    public boolean isEmpty()
    {
        //liefert true genau dann, wenn die Warteschlange kein Element enthält
        return (nbrElt == 0);
    }

    public boolean isFull()
    {
        //liefert true genau dann, wenn die Warteschlange voll ist
        return (nbrElt == size);
    }

    /**
     * @return the nbrElt
     */
    public int getNumberElements()
    {
        return nbrElt;
    }
}
```

```java
package oop2_dat2_u;

import java.util.ArrayList;

/**
 * Stack (LIFO: Last-In  First-Out) mittels ArrayList
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class StackArrayList<Object>
{

    private ArrayList<Object> stack = new ArrayList<Object>();

    public void push(Object o)
    {
        stack.add(o);
    }

    public Object pop()
    {
        // damit es keinen Fehler gibt zur Laufzeit, falls der Stack leer ist
        if ( ! isEmpty()) {
            return stack.remove(stack.size() - 1);
        } else {
            return null;
        }
    }

    public boolean isEmpty()
    {
        return stack.isEmpty();
    }

    public boolean isFull()
    {
        return false;
    }
}
```