```
::::::::::::::
Lab5Status.txt
::::::::::::::
Problem 0: compiles, runs correctly on all provided input::::::::::::::
Lab5Conclusions.txt
::::::::::::::
Not started, but prelab finished::::::::::::::
Lab5P0Driver.java
::::::::::::::
import java.io.*;

/*
 * Purpose: Data Structure and Algorithms Lab 5 Problem 0
 * Status: Complete and thoroughly tested
 * Last update: 10/05/23
 * Submitted:  10/05/23
 * Comment: test suite and sample run attached
 * Comment: I declare that this is entirely my own work
 * @author: William Carr
 * @version: 2023.10.05
 */

public class Lab5P0Driver {

    static BufferedReader stdin = new BufferedReader(new InputStreamReader(System.
in));

    public static void main(String[] args) throws NumberFormatException, IOExcepti
on {

        StackSLS<String> myStack = new StackSLS<String>();

        System.out.println("StackSLS ver.\nSelect from the following menu:\n"
                           +"\t0. Exit program\n"
                           +"\t1. Push item onto the stack\n"
                           +"\t2. Pop item from the stack\n"
                           +"\t3. Display the top item of the stack\n"
                           +"\t4. Display items in the stack\n"
                           +"\t5. Clear the stack\n");

        int selection;
        boolean continuing = true;
        do {

            System.out.print("Make your menu selection now: ");
            selection = Integer.parseInt(stdin.readLine().trim());
            System.out.println(selection);

            switch(selection) {
            case 1:
                pushItem(myStack);
                break;
            case 2:
                popItem(myStack);
                break;
            case 3:
                topItem(myStack);
                break;
            case 4:
                printStack(myStack);
                break;
            case 5:
                clearStack(myStack);
                break;
            default: // continuing unless told to stop
                continuing = false;
                System.out.println("Exiting program... Goodbye!");
                break;
            }

        } while(continuing);

    }

    public static void pushItem(StackInterface<String> stack) throws IOException{
        System.out.print("You are now inserting an item"+
                         " onto the top of the stack.\n\tEnter item: ");
        String itemName = stdin.readLine().trim();
        System.out.println(itemName);

        stack.push(itemName);
        System.out.printf("Item %s inserted onto"+
                          " the top of the stack.%n%n", itemName);
    }

    public static void popItem(StackInterface<String> stack) {
        if(stack.isEmpty())
            System.out.println("Stack is empty\n\n");
        else {
            System.out.printf("Item %s popped from"+
                              " the stack.%n%n", stack.pop());
        }
    }

    public static void topItem(StackInterface<String> stack) {
        if(stack.isEmpty())
            System.out.println("Stack is empty\n\n");
        else {
            System.out.printf("Item %s retrieved from"+
                              " the top of the stack.%n%n",
                              stack.peek());
        }
    }

    /**
     * Prints stack after checking for null/empty
     * @param stack
     */
    public static void printStack(StackInterface<String> stack) {
        if(stack == null || stack.isEmpty())
            System.out.print("Stack is empty.\n\n");
        else
            System.out.printf("Stack has the following items: %s%n%n",
                              stack.toString());
    }

    public static void clearStack(StackInterface<String> stack) {
        stack.popAll();
        System.out.println();
    }

}::::::::::::::
StackRA.java
::::::::::::::
```

```
public class StackRA<T> implements StackInterface<T>{

        private T[] items;
        private int top;

        public StackRA() {
                items = (T[])new Object[3]; //hand picked value of 3
                top = -1;
        }

        public boolean isEmpty() {
                return top == -1;
        }

        public void popAll() {
                items = (T[])new Object[3]; //see StackRA()
                top = -1;
        }

        /**
         * Assumes that the array is not empty & is full
         */
        private void resize() { // testing must reach 4 items in stack
                int lenItems = items.length;
                T[] temp = (T[])new Object[lenItems*2];
                System.arraycopy(items, 0, temp, 0, lenItems);
                this.items = temp;
        }

        public void push(T newItem) throws StackException{
                if(top == items.length-1)
                        resize();
                items[++top] = newItem;
        }

        public T pop() throws StackException{
                if(top == -1)
                        throw new StackException("pop(): stack is empty");
                T result = items[top];
                items[top--] = null;
                return result;
        }

        public T peek() throws StackException{
                if(top == -1)
                        throw new StackException("peek(): stack is empty");
                return items[top];
        }

        public String toString() {
                StringBuilder sb = new StringBuilder();
                for(int i = top; i >= 0; i--) {
                        sb.append(items[i] + " ");
                }
                return sb.toString();
        }
}:::::::::::::::
StackSLS.java
:::::::::::::::
public class StackSLS<T> implements StackInterface<T> {

        Node<T> top;

        public StackSLS() {
                top = null;
        }

        public boolean isEmpty() {
                return top == null;
        }

        public void popAll() {
                top = null;
        }

        public void push(T newItem) throws StackException{
                this.top = new Node<T>(newItem, top);
        }

        public T pop() throws StackException{
                if(top == null)
                        throw new StackException("pop(): stack is empty");
                T item = top.getItem();
                this.top = top.getNext();
                return item;
        }

        public T peek() throws StackException{
                if(top == null)
                        throw new StackException("pop(): stack is empty");
                return top.getItem();
        }

        public String toString() {
                StringBuilder sb = new StringBuilder();
                Node<T> current = top;
                while(current != null) {
                        sb.append(current.getItem()+" ");
                        current = current.getNext();
                }
                return sb.toString();
        }

}:::::::::::::::
Node.java
:::::::::::::::
public class Node<T> {
    private T item;
    private Node<T> next;

    public Node(T newItem) {
        item = newItem;
        next = null;
    } // end constructor

    public Node(T newItem, Node<T> nextNode) {
        item = newItem;
        next = nextNode;
    } // end constructor

    public void setItem(T newItem) {
        item = newItem;
    } // end setItem
```

```
    public T getItem() {
        return item;
    } // end getItem

    public void setNext(Node<T> nextNode) {
        next = nextNode;
    } // end setNext

    public Node<T> getNext() {
        return next;
    } // end getNext

    public String toString()
    {
        return item+" ";
    } //end toString

} // end class Node::::::::::::::
StackInterface.java
::::::::::::::
public interface StackInterface<T> {
  public boolean isEmpty();
  // Determines whether the stack is empty.
  // Precondition: None.
  // Postcondition: Returns true if the stack is empty;
  // otherwise returns false.

  public void popAll();
  // Removes all the items from the stack.
  // Precondition: None.
  // PostCondition: Stack is empty.

  public void push(T newItem) throws StackException;
  // Adds an item to the top of a stack.
  // Precondition: newItem is the item to be added.
  // Postcondition: If insertion is successful, newItem
  // is on the top of the stack.
  // Exception: Some implementations may throw
  // StackException when newItem cannot be placed on
  // the stack.

  public T pop() throws StackException;
  // Removes the top of a stack.
  // Precondition: None.
  // Postcondition: If the stack is not empty, the item
  // that was added most recently is removed from the
  // stack.
  // Exception: Throws StackException if the stack is
  // empty.

  public T peek() throws StackException;
  // Retrieves the top of a stack.
  // Precondition: None.
  // Postcondition: If the stack is not empty, the item
  // that was added most recently is returned. The
  // stack is unchanged.
  // Exception: Throws StackException if the stack is
  // empty.


  public String toString();
  // Return the String representation of the items in the
  // collection (top to bottom; single space delimited)
  // - no additional narratives

}  // end StackInterface
::::::::::::::
StackException.java
::::::::::::::
public class StackException
          extends java.lang.RuntimeException {
  public StackException(String s) {
    super(s);
  }  // end constructor
}  // end StackException::::::::::::::
Lab5P0Sampleruns.txt
::::::::::::::
StackRA ver.
Select from the following menu:
        0. Exit program
        1. Push item onto the stack
        2. Pop item from the stack
        3. Display the top item of the stack
        4. Display items in the stack
        5. Clear the stack

Make your menu selection now: 2
Stack is empty


Make your menu selection now: 3
Stack is empty


Make your menu selection now: 4
Stack is empty.

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: fox
Item fox inserted onto the top of the stack.

Make your menu selection now: 3
Item fox retrieved from the top of the stack.

Make your menu selection now: 4
Stack has the following items: fox

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: brown
Item brown inserted onto the top of the stack.

Make your menu selection now: 4
Stack has the following items: brown fox

Make your menu selection now: 2
Item brown popped from the stack.

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: brown
Item brown inserted onto the top of the stack.
```

```
Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: quick
Item quick inserted onto the top of the stack.

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: The
Item The inserted onto the top of the stack.

Make your menu selection now: 4
Stack has the following items: The quick brown fox

Make your menu selection now: 3
Item The retrieved from the top of the stack.

Make your menu selection now: 2
Item The popped from the stack.

Make your menu selection now: 4
Stack has the following items: quick brown fox

Make your menu selection now: 5

Make your menu selection now: 4
Stack is empty.

Make your menu selection now: 0
Exiting program... Goodbye!
StackSLS ver.
Select from the following menu:
        0. Exit program
        1. Push item onto the stack
        2. Pop item from the stack
        3. Display the top item of the stack
        4. Display items in the stack
        5. Clear the stack

Make your menu selection now: 2
Stack is empty


Make your menu selection now: 3
Stack is empty


Make your menu selection now: 4
Stack is empty.

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: fox
Item fox inserted onto the top of the stack.

Make your menu selection now: 3
Item fox retrieved from the top of the stack.

Make your menu selection now: 4
Stack has the following items: fox

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: brown
Item brown inserted onto the top of the stack.

Make your menu selection now: 4
Stack has the following items: brown fox

Make your menu selection now: 2
Item brown popped from the stack.

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: brown
Item brown inserted onto the top of the stack.

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: quick
Item quick inserted onto the top of the stack.

Make your menu selection now: 1
You are now inserting an item onto the top of the stack.
        Enter item: The
Item The inserted onto the top of the stack.

Make your menu selection now: 4
Stack has the following items: The quick brown fox

Make your menu selection now: 3
Item The retrieved from the top of the stack.

Make your menu selection now: 2
Item The popped from the stack.

Make your menu selection now: 4
Stack has the following items: quick brown fox

Make your menu selection now: 5

Make your menu selection now: 4
Stack is empty.

Make your menu selection now: 0
Exiting program... Goodbye!
```