Objective:

This lab introduces the stack abstract data type. You will implement the Stack interface using an ArrayList. You will use the Stack class to check the validity of an expression with nested parentheses.

Background:

This lab is based on material found in Section 20.10 (Stack ADT) of our online textbook.

A stack stores a collection of objects. A stack differs from, say, an ArrayList by obeying the last-in-first-out (LIFO) discipline meaning that when you remove an item from a stack you remove the last item put into the stack. It may help to think of a stack of trays or a stack of plates in Hickey dining hall. Typically you retrieve the tray or the plate on the top of the stack. Another example of the idea of a stack is the “Back” button in a Web browser. Whenever you follow a link you are pushing (inserting) the URL onto a stack. When you click the “Back” button you are popping (removing) the last URL visited from the stack and returning to that web page.

In the case of a stack the insert operation is called push and the remove operation is called pop. There is also the peek operation which returns the item at the top of the stack without removing it.

Instructions:

1. Begin by creating a new Java project in Eclipse. Add the file Stack.java to this project. This file defines the Stack<E> interface. Examine this interface. You may want to generate and then print the Javadoc specification of the interface.

Create a class named ArrayListStack<E> that implements the Stack<E> interface. Implement the Stack<E> interface using an ArrayList<E>. The private data of the class should include an ArrayList to store the items. All insertions and deletions need to take place at one end of the ArrayList, usually called the top of the stack. There are two possibilities. You can do all insertions and deletions at the beginning of the list (position 0) or at the end of the list (position size()). Explain why is it better to do this at the end rather than at the beginning.

Implement the four Stack methods so that all operations take place at the end of the ArrayList. Don't forget to check for and throw the exception EmptyStackException when required.

Document the ArrayListStack<E> class, print a copy of this file, and hand it in with your lab write-up.
2. Create a main class named StackMain. Within the main method create an ArrayListStack<Character> and then push (insert) the characters ‘A’, ‘E’, ‘I’, ‘O’, and ‘U’ onto the stack in that order.

Answer the following questions:
- After pushing the characters on to the stack, what character is now at the top of the stack?
- What statement(s) can you use to print that character (without removing it from the stack)?
- What statement(s) can you use to remove that character from the stack?
- After you remove the character at the top of the stack what character is now at the top of the stack?

Modify the main method to print and then remove each of the characters from the top of the stack (do this after the 5 characters have been pushed onto the stack). Use a while loop to accomplish this. The pseudo-code is:

```
loop while the stack is not empty
    print the top element
    remove the top element
```

Run the modified program to be sure it works. Record what is displayed.

Modify the body of the while loop to use a single statement to both print and remove the top element. Further modify the program to display the output on a single line.

Run the modified program and record what is displayed.

3. Modify the main method in StackMain to read a line of text from the console and then display the line in reverse order. Use an ArrayListStack to reverse the order.

Use the nextLine() Scanner method to read a line of text from the console. Once you have the line of text as a String, use a loop to push each character from the line onto a stack. (Use the charAt(i) String method to get the individual characters from the line.) After you have pushed all the characters onto the stack use the code from the previous step to display the characters stored on the stack on one line.

Run the program on three different input strings. For each one print your results and hand them in with your lab write-up.

Run the program again with the string “Able was I ere I saw Elba”. Print your results and hand them in with your lab write-up.

What property does this last string have? What name is given to this property? Can you give another word/phrase (with at least 6 letters) with this property?

Document the file StackMain.java. Print a copy of the file and hand it in with your lab write-up.

4. One application of stack operations is a program to check for balanced parentheses and braces. By balanced we mean that every open parenthesis is matched with a corresponding close parenthesis and that parentheses are properly nested. This problem becomes slightly more interesting by considering both parentheses and braces. All other characters are to be ignored. So, for example, the inputs

```
(x(y)(z)) and a( { (b) } c)
```

are balanced, whereas the inputs

```
w) (x) and p( (q) r )
```

are not balanced.
Modify the main method to read an expression from the console and determine if the expression is balanced or not. Define a boolean variable `balanced` which is initially true. If at any point you find the balanced property violated set `balanced` to false. Here is the pseudo-code for this task:

initialize `balanced` to true
create an empty Stack
read an expression from the console as a line of text
for each character in the expression
    if the character is a '(' or a '{'
        then push the character onto the stack
    if the character is a ')' or a '}'
        then check if the stack is empty
            if it is empty
                then the expression is not balanced
            if it is not empty
                then remove the character from the top of the stack
                    if the character removed from the top of the stack is not the matching (opening) character
                        then the expression is not balanced
    if the stack is not empty
        then the expression is not balanced
Print a message indicating if the expression is balanced or not.

Run the program on each of the previous example expressions. Also run the program on each of the following exercises:
   a) \( (w*(x+y)/z-(p/(z-q))) \)
   b) \( (w*(x+y)/z-(p/{r-q})) \)
   c) \( (a+b*{c/(d-e)})+(d/e) \)
Print a copy of the console window with the input and the output in each case and hand these in with your lab write-up.

Document the main class. Print a copy of the main class and hand it in with your lab write-up.

**Extra Credit:**

Modify the program you wrote for Step 4 to express it as a function/method and to make it more efficient. The pseudo-code given requires the entire expression be processed even if it is found not to be balanced early in the expression. Modify the code to terminate the method as soon as the balanced property is found to be violated. The main method is responsible for reading the expression and then displaying the appropriate message.

Run the modified program on the same set of exercises that you used in Step 4 above. Print a copy of the console window with the input and the output in each case and hand these in with your lab write-up. Print a copy of the main class and hand it in with your lab write-up.
Hand in:

The write-up you hand in for this lab should include:
- the reason given in Step 1.
- the printed copy of the ArrayListStack class requested in Step 1.
- the answers to the questions asked in Steps 2 and 3.
- the record of results requested in Step 2.
- the printed copy of output (console window) requested in Steps 3 and 4.
- the printed copy of the main method requested in Steps 3 and 4.
- the printed copy of the results and the code requested in the extra credit step.

Help Policy:

Help Policy in Effect for This Assignment:  Group Project with Limited Collaboration

In particular, you may discuss the assignment and concepts related to the assignment with the following persons, in addition to an instructor in this course: any member of your group; any St. Bonaventure Computer Science instructor; and any student enrolled in CS 132.

You may use the following materials produced by other students: materials produced by members of your group.

You may use the following materials produced by other students: NONE.