Lab: If (Conditional Logic)

Similar to your calculate lab, your page (named if.jsp) shall accept user input, perform validation, do a calculation and show the calculated results. You may expand on the same calculation you did for last lab or you may create a new unrelated calculation. It would be nice if your calculation has something to do you’re your web site topic, but it is not mandatory. Your page shall also have the following:

Named Constant
- Somewhere in your code, a “named constant” shall be used. Look at lecture notes and sample JSP code for help understanding what this is. Remember that the convention for naming constants is ALL_CAPS (not camelCase like you would use for variables). Choose a name that is self documenting (probably a bit longer, indicates what the constant represents).

Input and validation
- if.jsp shall tell the user what to enter and explain what calculations will be performed (say in words what the calculations will be, so we can grade your homework).
- Your page shall take at least one real number input, and at least one integer input. Using Double.parseDouble() and Integer.parseInt(), convert the user’s numeric input to the proper data types so that you can do some arithmetic. (Sample JSP code can help you with this.)
- If the user enters any non-numeric values (where numeric values are expected), the page shall display a user-friendly, descriptive error message (use try/catch exception handling for this).
- All input boxes will have their own (field level) error messages. Color these error messages red by using CSS, e.g., `<span class="error"><%=fldLevelErrorMsg%></span>`. If the error message is “” (empty string), then no error will appear. Hint: in order to have field level error messages, each numeric input value must have its own try/catch block (where you try to convert the value to a number).
- **Scope is an important concept** in computer languages. The “scope of a variable” is the block in which the variable is declared. So this code will NOT work:

```java
try {
    int intQty = Integer.parseInt(strQty);
} catch (Exception e) {
    strQtymsg = "Please enter an integer";
}
//…
dblAmount = intQty * dblPrice; // intQty is out of scope – red compiler error in NetBeans
```

But this code WILL work:

```java
int intQty = 0;
try {
    intQty = Integer.parseInt(strQty);
} catch (Exception e) {
    strQtymsg = "Please enter an integer";
}
//…
dblAmount = intQty * dblPrice; // OK. intQty is IN scope (in the block in which it was declared)
```
**if.jsp** shall take at least one string input that is expected to be from a short list of at least 3 acceptable answers. The user will enter this value into a text box (not select list). What the user enters here shall have an impact on the calculated result. To perform this validation (and have the impact on the calculated result, use the multi-branch if structure: if, else if, ..., else if, else. Hint: the final “else” block will be where you set an error message (that the user has not entered one of the acceptable values).

- Before trying to compare the input string, make sure that this string has been trimmed (no leading or trailing spaces) and that it has been converted to upper case.
  
  ```java
  strCustomerType = strCustomerType.trim();
  strCustomerType = strCustomerType.toUpperCase();
  ```

- Here is how you compare one string to another:
  ```java
  if (strA.equalsIgnoreCase(strB)) {    //…
  ```

- When the user clicks submit, the input values shall persist (continue to show in their input boxes). The word “null” shall never display on the form.

**Calculation and Output**

- If all input passes validation, then your page shall perform some useful calculation that is based on all the inputs (including the string input) – as well as the numeric constants. You may be wondering how the value of a string input could affect a numeric calculation. Here is an example that might help to explain:

  ```java
  double dblDiscountRate = 0;
  if (strCustomerType.equalsIgnoreCase("EMP")) {
    dblDiscountRate = .12;
  } else if (strCustomerType.equalsIgnoreCase("OWN")) {
    dblDiscountRate = 0.18;
  } else if (strCustomerType.equalsIgnoreCase("CUS")) {
    dblDiscountRate = 0.05;
  } else {
    msg = "Invalid Customer Type";
  }
  ```

- As part of your calculation, use at least one method from the java “Math” class. To see the list of possible Math methods, Google “javadoc Math” and find this page https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html Under “Method Summary”, see the list of Math methods you can use. Methods such as min, max, round, and random are quite nice and easy to use.

- If all input passes validation, then your page shall display a message on the page. This message shall incorporate (label/display) all the input values as well as the calculated output, formatted nicely, e.g.,

  ```java
  double dblTotalAmt = dblPrice * intUnits;
  msg = "The total is " + "$" + String.format("",10.2f", dblTotalAmt);
  ```

  The writeup for this lab (in the 1056 web page) links to a JSP page that provides some sample code related to formatting numeric output.

**Self Documenting Variable Names**

- All variable names should be self documenting (a bit long) using camelCase naming convention.

- Use a type designation as a prefix for your variable names, e.g., strPrice vs. dblPrice (makes your code easier to read and less error prone).
Style and Linking
- The style of `if.jsp` must match the layout of your other pages.
- Create a blog (short paragraph) on your labs page to describe what you are learning in this lab and link from the labs page to this page, `if.jsp`.

Code Reuse and Syntax Errors
- This page (like all other pages) shall reference your external style sheet as well as your common HTML code using JSP include statements. There shall be no syntax errors when we “View Source” from Firefox.

Regression Testing
- Your web app shall be regression tested, which means “all your old pages still work properly” (including a consistent look and feel). This should be easy if you are employing code reuse. With JSP/java, you do not publish your java source code, so it is your responsibility to keep track of your source code and not lose it. At the end of the semester, all of your code will be checked for good programming style and all pages must be properly functioning from the source code (e.g. locally on your pc, not just on the server).

Other
- Be sure to read the entry entitled “Requirements for All Labs and Project” at the top of the 1056 web page. You also need to meet the requirements specified in there (but I think I have included most of them in this writeup).

Submission
- As for every lab, be sure to write a blog in your labs page (talking about what you learned, what was hard, what was easy). Include a link from your blog to the JSP page you wrote this week.
- Publish and test your web application (which should now have 4 pages: index.jsp, labs.jsp, calculate.jsp, and if.jsp) then submit a zip file of your project to Blackboard by the due date.