Welcome to the Branches, Eclipse & Styling lab!

Learning Outcomes

By the end of this lab:
- Be able to define chapter 4 terms
- Trace, explain and write:
  - expressions using if, if-else and switch statements. (not nested)
  - expressions with logical boolean operators (||, &&, !)
  - expressions with relational/comparison operators (==, <, >, <=, etc.)
  - expression using == and equals for comparing references and String values
- Create and use a truth table for a boolean expression
- Use a professional development environment to create, compile and run a program. (Eclipse)
- Format and comment a program according to course standards.

Exercise A: Terms

boolean data type, "true" vs true, branching, if vs if-else, use of braces { } with if, = vs ==, == vs equals(), multiple ifs vs if-else, logical operators (&&, ||, !), switch statement, case, break, default.

Operator Precedence Table:

<table>
<thead>
<tr>
<th>Level</th>
<th>Operator</th>
<th>Description</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>higher</td>
<td>( &lt;expression&gt; )</td>
<td>grouping with parentheses array index, method call, member access (dot operator)</td>
<td>left to right.</td>
</tr>
<tr>
<td></td>
<td>[ ] ( ) .</td>
<td>post-increment, post-decrement pre-increment, unary plus/minus, logical negation</td>
<td>left to right.</td>
</tr>
<tr>
<td></td>
<td>++ --</td>
<td></td>
<td>right to left.</td>
</tr>
<tr>
<td></td>
<td>++ -- + - !</td>
<td></td>
<td>right to left.</td>
</tr>
<tr>
<td></td>
<td>(type) new</td>
<td>casting and creating object multiplication, division, modulus addition, subtraction, concatenation relational and Java's instanceof operator</td>
<td>right to left.</td>
</tr>
<tr>
<td></td>
<td>* / %</td>
<td></td>
<td>left to right.</td>
</tr>
<tr>
<td></td>
<td>+ - +</td>
<td></td>
<td>left to right.</td>
</tr>
<tr>
<td></td>
<td>&lt; &lt;= &gt; &gt;=</td>
<td></td>
<td>left to right.</td>
</tr>
<tr>
<td></td>
<td>instanceof</td>
<td></td>
<td>left to right.</td>
</tr>
<tr>
<td></td>
<td>== != &amp; &amp;</td>
<td>equality conditional AND (short-circuits) conditional OR (short-circuits) ternary conditional assignment</td>
<td>left to right.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>right to left.</td>
</tr>
<tr>
<td>lower</td>
<td>= += -= * = / = % =</td>
<td>assignment</td>
<td>right to left.</td>
</tr>
</tbody>
</table>

Exercise B: Trace and Explain

A. boolean completed = false;
   System.out.println( b);
B. boolean completed = true;
   if (completed) {
       System.out.println("completed");
   } else {
       System.out.println("not completed");
   }

C. boolean completed = false;
   if (completed || !completed) {
       System.out.println("hmmm");
   } else {
       System.out.println("HMMM");
   }

D. boolean finished = false;
   finished = !finished;
   System.out.println( finished);

E. boolean happy = false;
   if (happy) System.out.println("happy");
   else System.out.println("not happy");

F. if (true) System.out.println("does this print?");

   boolean done = false;
   if (done || true) {
       System.out.println( done);
   }

G. boolean flag = false;
   if (flag = true) {
       System.out.println("flag is true");
   } else {
       System.out.println("flag is false");
   }

H. boolean flag = false;
   if (flag == true) {
       System.out.println("flag is true");
   } else {
       System.out.println("flag is false");
   }

I. boolean flag = false;
   if (flag) {
       System.out.println("flag is true");
   } else {
System.out.println("flag is false");
}

J. boolean a = true;
    boolean b = false;
    if ( a || b) System.out.println("yo");
    if ( a && b) System.out.println("another yo");

Are there any values of a and b that would cause both "yo" and "another yo" to print out?

K. boolean flag;
    //flag gets some value
    flag = flag == false;

L. Scanner input = new Scanner( System.in);
    String str = "end";
    System.out.print("Type end:"); //when prompted type: end
    String str2 = input.next();
    if ( str == str2) {
        System.out.println("str == str2");
    } else {
        System.out.println("str != str2");
    }

    if ( str.equals( str2)) {
        System.out.println("str == str2");
    } else {
        System.out.println("str != str2");
    }

Can you summarize the difference between == and equals() for strings?

Exercise C: Get Started with Eclipse

Follow the tutorial here to get started with Eclipse.

Exercise D: Style and Comment

Using Eclipse, format MessedUp.java following the CS 200 Style Guide.

In particular:
Commenting
- Each file has a file header comment
- Each class has a class header comment
Each method has a method header comment
- Variable declarations briefly commented
- Comments within methods (highlight major steps, explain long calculations, clarify unusual code)

**Style**
- 4 space indenting for each level of nesting
- Vertical whitespace (organize code into meaningful parts)
- Horizontal whitespace (more readable statements)
- Column limit: 100 characters (wrap with extra indent)
- 1 statement per line
- Consistent formatting and use of braces
- Descriptive names
- Names follow conventions (CONSTANT, ClassName, variableName)
- Use of constants instead of magic numbers (literals such as 3.1415, 7, etc)

**Hints:**
- **Formatting:** In Eclipse, select code, go to Source menu, select Format, and then select Correct Indentation
- **Renaming Variables:** In Eclipse, right click on a variable name, select Refactor, select Rename. Type the new name and the name will be changed everywhere.

**Exercise E:**

```java
int age;
// read in age
if ( age < 5 ) {
    System.out.println("preschool");
} else if ( 5 < age ) {
    System.out.println("not preschool");
}
```

Consider: will all possibilities of age result in at least one path through the if statement being executed? How could this be written simpler to make sure all possibilities of age are handled?

**Exercise F: Truth Table**

```java
if ( !a || a && b ) System.out.println( true);
```

If a and b are boolean variables we want to know for which values of a and b this will be true. Can you figure this out with a truth table?

**Step 1:** Identify the boolean variables
- a, b

**Step 2:** identify the order the operators are applied. From the precedence table we see that ! is highest then && and then ||.

- a, !a, b, a && b, !a || a && b
Step 3: put each boolean sub expression as the top of a column in a table. List boolean variables first.

| a | b | !a | a && b | !a || a && b |
|---|---|----|--------|-------------|
|   |   |    |        |             |

Step 4: list all possible combinations of true and false for all the variables

| a  | b  | !a | a && b | !a || a && b |
|----|----|----|--------|-------------|
| true | true |   |        |             |
| true | false |   |        |             |
| false | true |   |        |             |
| false | false |   |        |             |

Step 5: fill in boolean values for the rest of expressions in the order they would be evaluated

| a  | b  | !a | a && b | !(a) || (a && b) |
|----|----|----|--------|------------------|
| true | true | false | true | true             |
| true | false | false | false | false            |
| false | true | true | false | true             |
| false | false | true | false | true             |

Step 6: summarize by looking at the results for the final expression:

| a  | b  | !a | a && b | !(a) || (a && b) | !a || b |
|----|----|----|--------|------------------|--------|
| true | true | false | true | true             |        |
| true | false | false | false | false            |        |
| false | true | true | false | true             |        |
| false | false | true | false | true             |        |

Is it possible to get an equivalent result with a simpler expression? Which expression would we want to use? (It depends... which is more easily understandable by someone reading your code?)

```java
if (       ) System.out.println( true);
```
Exercise G:

Assuming flag is a boolean variable what is a 1 sentence summary of the following expression?

flag = flag == false

Use the truth table to verify your answer.

<table>
<thead>
<tr>
<th>flag</th>
<th>flag == false</th>
<th>flag = (flag == false)</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If flag starts as true what is its resulting value?

If flag starts as false what is its resulting value?

How would you summarize what this expression does?

Exercise H:

boolean flag = false;
boolean result = flag = true || flag;

Can you summarize what this does?

Finish filling out the truth table.

| flag  | true || flag | flag = (true || flag) | result |
|-------|---------|--------------|----------------------|--------|
| true  |         |              |                      |        |
| false |         |              |                      |        |

What can you say about result based on the different values of flag?

Exercise I: When does old age begin?

Write a program to prompt the user for their age. Based on their age print out which generation they are, when they think youth ends, when old age begins and what age is considered the prime of life.

---

1Parenthesis added to emphasize order of evaluation following precedence rules.
2Parenthesis added to emphasize order of evaluation following precedence rules.
What is your age: 45
You are Generation X.
Youth ends at 31, old age begins at 65.
The prime of life is age 47.

Read this article for the data: "This is the age you become old"

**Additional Learning Materials**

When you have mastered everything in this lab, then you are welcome to learn from additional learning resources available on the web and beyond this course:
https://cs200-www.cs.wisc.edu/wp/learn-to-program-resources/

*Note: Due to programs and zyBooks being individual work, it is Not appropriate to work on them during the Team Lab.*

Lab designed by Jim Williams