CS 200 Sections 02 & 04 Spring 2013

Week #8: Top 3 Lessons Learned

The personal lessons that I learned from this lecture were from chapter three (3.2 and 3.3) were (1) the complete evaluation for type Boolean and the short-circuit (lazy evaluation) from chapter three section two, (2) what a binary decision structure and decision structure with multiple options (switch/case), and (3) the rules for using switch/case statements in programming.

The complete evaluation for type Boolean for "or" use: |; for "and" use: &; if Boolean x is true, Boolean y is false, and Boolean z is true and are using the symbols | or &, the evaluation is checked on all variables. As for the short-circuit evaluation, which is also known as lazy evaluation, the statement in this case, the evaluation stops at x being true, then if statement is executed; however, if this was an "and" statement with y being the first variable in the compound statement, then the statement would be skipped.

As for the decision structures mentioned here, they are binary (if/else) and multiple options (switch/case). The binary decision structure is an if/else statement in which a point when a decision outcome determines which of the two segments of code should be executed. The multiple options decision structure is a switch/case statement in which a point when a decision structure with outcome with multiple choices, and then it determines which segment of code should be executed.

At last, there are four main rules for using a switch/case statement in programming. The four rules of the switch/case statement are (1) controlling expression: which must be a char, byte, short, or int data type, (2) case label: must be a constant of the same type as the controlling expression, (3) break statement: this must be included at the end of each case compound statement, otherwise once a case is matched, all subsequent commands will be executed until the end of the switch compound statement, and (4) default statement: this is used to catch invalid data input (which is good programming form).

D. McManus

1) There is a short-cut logic expression with boolean. Instead of writing this if statement: if(truthValue == true), we can write if(truthValue). Both have the same meaning, is the boolean value of truthValue true.

2) Short-circuit lazy evaluation is used by the compiler. Example 1: value1 || value2, if the value1 is true then the whole expression is true. Example 2: value1 && value2, if the value1 is false then the whole expression is false.

3) Switch/case is a decision structure. It does not have to be written in order. The controlling expression has to be int, short, byte or char data type. Therefore, the case label must have the same data type as the controlling expression.

D. Starostka

Three things that I learned on 02/26/2013 are that a controlling expression for the switch statement must be of a char, byte, short, or int data type. You can use enum to use values that are not of char, byte, short, or int data type in a controlling expression. You can do this because enum stores those values as integers. Second, the convention for naming values in an enumeration is that all the letters must be uppercase. Third in a switch statement, it is a good idea to include a default case because it is considered good programming form.

J. Gomez

1. The controlling expression for a switch must be of type char, byte, short, or int and the case label must be the same type as the controlling expression.
2. Each statement in a switch case must include a break. If it doesn't include a break, then all the statements after the matched case will also execute until the end of the switch or until a break is encountered.. This is the only time in this class that we'll be allowed to use a break.
3. The enumerated type acts like a class and it needs to be declared outside of main. Declaring a variable for an enumeration is very similar to a string because you just use the *ENUM variableName*, but initialization involves dot notation with the enum definition.
enum Grades{A, B, C, D, F}
Grades student1 = Grades.C;

Here's what I found out about &&, ||, &, and | after quite a bit of digging. I found a web site called stackoverflow.com and they state that & and | were "C" bitwise operators and also used for logical operators. It got confusing in the code and was hard to distinguish what it was being used for, so Dennis Ritchie(Bell Labs/Lucent Technologies) decided to update the C language so the logical operators would be && and ||. The & and | can be used for both bitwise operators and logical operators, but will perform complete evaluation where the && and || will perform short circuit evaluation. Here are the 2 web pages explaining the bitwise and logical operators and how they came to be:

http://stackoverflow.com/questions/11597978/are-the-bitwise-operators-or-logical-operators

http://cm.bell-labs.com/who/dmr/chist.html

I don't know if I'll ever use the bitwise operators, but thanks for taking me fishing because it was a nice feeling to catch something.

E. Zacharias

1.) When a case is selected as part of the switch statement, the statements under that case will activate until the break is reached. You can have several cases have the same statement by not including the break until the end of the last case.

2.) The default statement is always at the end of the switch; the statements here are usually used when the user does not select any of the cases in the switch. A default it not needed, but it is good programming etiquettes to have one.

3.)The enum declarations does not need a semi colon at the end, even if you do add one the compiler will just ignore it so no changes will be made to the program. Using enum allows the switch/case to be more readable, since a switch can only be a char, byte, short, and int type.

E. Herring

(1) ' || ' signifies, 'or' also known as an OR gate. '&&' means 'and' also known as an AND gate.

Example.. if(s.charAt(0) = 'y' || s.charAt(0) = 'Y') one of these must be true.

Another example... if (pennyCount > 0 && pennyCount < 100) both of these must be true.

(2) Break statement: at the end of each case compound statement. Must be included. Otherwise all the following cases will execute.

(3) Default statement, can be used to catch invalid data input. Its a separate case. The program asks to input a number 0-9. If the input is a negative number or anything greater than or equal to 10. The default case will be executed.

S. McGovern

Here is my Lessons Learned

Week 9 :

{ // open Lessons Learned

First of all, The boolean type stores values : true of false.
True or False determines the path of commands to be executed using the “if” statement.
Break statement has to be included at the end of each case statement to break out of the switch statement.
Default statement is often use to catch invalid data input.
As user data entry, true, TRUE, True , false, FALSE, False or any other combination will work, no matter with lowercase and Uppercase.

PS: Even through a chapter is not going to be in a final exam, its important for us to work on it, for our career, our background in order to do not have regrets later.

} // close Lessons Learned
J. Konan

1. In an expression of the form subexpression1 || subexpression2, only 1 of the subexpressions needs to be true for the overall expression to be true. If Java finds subexpression1 to be true it doesn't evaluate subexpression2 because the overall expression is already considered true.
In an expression of the form subexpression1 && subexpression2, only 1 of the subexpressions needs to be false for the overall expression to be false. If Java finds subexpression1 to be false it doesn't evaluate subexpression2 because the overall expression is already considered false.

2. The switch statement uses a controlling statement which is evaluated and then compared to a series of cases. When a case matches the controlling statement's value, the action for that case is run.

3. Use enumeration when you want to restrict the possible values of a variable. You list the values when you define the enumeration.

J. Hoffman

| does not do short-circuit evaluation in boolean expressions. || will stop evaluating if the first operand is true, but | won't.
In addition, | can be used to perform the bitwise-OR operation on byte/short/int/long values. ||cannot.
source: http://stackoverflow.com/questions/7101992/why-do-we-usually-use-not-what-is-the-difference

1. User data entry : nextBoolean();
true, TRUE, True and false, FALSE, False: or any other combination will work.
2. Short-circuit / lazy evaluation
* boolean x=true;
* boolean y=false;
* boolean z=true;
* if(x||y||z)
  //evaluation stops at x being true and the if is executed
* if (y&&z)
  //evaluation stops at y being false and the if is skipped
3. You can nest if-else -if part and the else
part can have another if-else statement as its action. Using braces to form compound statements can clarify your intent. Without braces, remember that each else is paired with the nearest preceding and unpaired if.

A special form of nested if-else statements forms a multibranch structure.
In this form, you typically write the next if part immediately after an else.
Thus, you write if, else if, else if,…, else. The last else does not contain an “if” portion and should be the last bucket of possible data outcomes.

D. Mirdadi

1. short-circuit evaluation - If the first argument is true/false, the entire expression is true/false.
if (x || y || z) //stops at x!
if (y && z) //stops at y!

2. In case of switch statement, break statement causes exit from switch, if not used, it will continue check all the next cases until the last case.

3. Enumerated type (enum) is basically used for code readability.

Ex:

```
public enum Currency {PENNY, NICKLE, DIME, QUARTER};
Currency coin = Currency.PENNY;
coin = 1; //compilation error
```

It is a type of class (therefore, declared outside of main)

```
Currency usCoin = Currency.DIME;
   switch (usCoin) {
         case PENNY:
               System.out.println("Penny coin");
               break;
         case NICKLE:
               System.out.println("Nickle coin");
               break;
         case DIME:
               System.out.println("Dime coin");
               break;
         case QUARTER:
               System.out.println("Quarter coin");
   }
```

M. Mardosz

```
x=true;
y = false;
z=true;
```
if (x || y || z) //evaluation stops at x being true and the if statement is executed. (whether y and z are false is not checked)
if (y && z) //evaluation stops at y being false and the if statement is skipped.
In order to check all variables in the if statement a single '&' or '|' is used
if (x | y | z) all the variables will be evaluated
if (x & y) both of them are checked even if the x is false
when a user enters a boolean true or false, any combination of the word is accepted.

S. Malik

1.An enumeration gives us a way to restrict values of a variable.
2.Each action in switch statement ends with a break statement.
3.Boolean specifies only two values: True or False
4.Unary operators are first in precedence.

M. Zaki