# CS 200 Sections 02 & 04 Spring 2013

## Week #7: Top 3 Lessons Learned

1) **Branching statements choose one action path from a list of two possible actions (t or f).** Instead of the regular flow control that we have previously have been experimenting with, we have another tool that adds more options to the linear process. We can create more complex branching by joining or nesting (if / else statements).

2) **if-else statement: it first checks the expression in parentheses after the keyword "if". The expression must be either true or false. If it is true, the one statement directly following is executed. If the expression is false, the one statement after the else is executed.**

3) **If you want to include more than one statement in each branch, enclose the statements in braces { }. Even though the "if" will still run if you have one statement and no braces, It is still a good practice to include braces.**

EX:

```
if(.....)
{
....
}
else
{
.....
....
.....
}
```

*** Also DO NOT include a ";" after the "if" statement as you have then written the NULL statement and Java will not execute what you think is the body of the IF.**

-Alex K.

The personal lessons that I learned from this lecture were from chapter three (3.1) were (1) what a decision structure is and how it is used in Java , (2) the different types of decision structures used when making a decision , and (3) the different types of syntax that follow the flowchart structures within this section of chapter three.

What is a decision structure in Java programming? A decision structure is a decision outcome that determines which segment of code should be executed within any program.

As for the different types of different structures being used when making a decision are used when making a decision when writing a program in Java. The different types of decisions used in this section of chapter three are if statement, if-else statement, the nested if statement, and the if-else-if-else statement. Each of these types of decision structures in programming has their own flowchart structures and syntax.

The different types of syntax included in each of these types are:

(1) only one statement, following the if-statement will be executed if the condition is true (if-statement),

(2) if the expression is true, the statement is executed, otherwise the second statement is executed (the else labeled statement),

(3) if the expression given is true, then, if the second expression is true, (nested-if statement), and the last piece of syntax used in the decision structure for this section of chapter three is

(4) if there are more than 2 branches to a decision it can be set up with (if-else-if-else statement), where the last else is if all previous statements are false.

D. McManus

1.) I learned how the if-else how two seperate paths of commands can be executed. One path if the boolean experession within the if statement evaluates to be true, and for else a 2nd path of command is executed before flow continues in a linear function.

J. Morales

1. **Always put starting and ending curly braces when making an if statement in java or it will be hard to notice the missing braces with a complicated code.**
2. **Mark the ending brace with a comment to let yourself know it is the end of the if statement.**
3.Only one statement following the if statement will be executed if the condition is true.

S. Malik

1. **The diamond shape in the flow chart structure is the decision process. There should be two points coming off of it and they should be labeled as yes (true) and no (false).** This is what the if statement does.

2. **If a logic statement is true, it executes only one statement. If you want to do more than one true statement you would treat them as a unit by creating a compound statement using the curly braces { } and add comments. Remember that you do not add the semicolon after the if statement.**

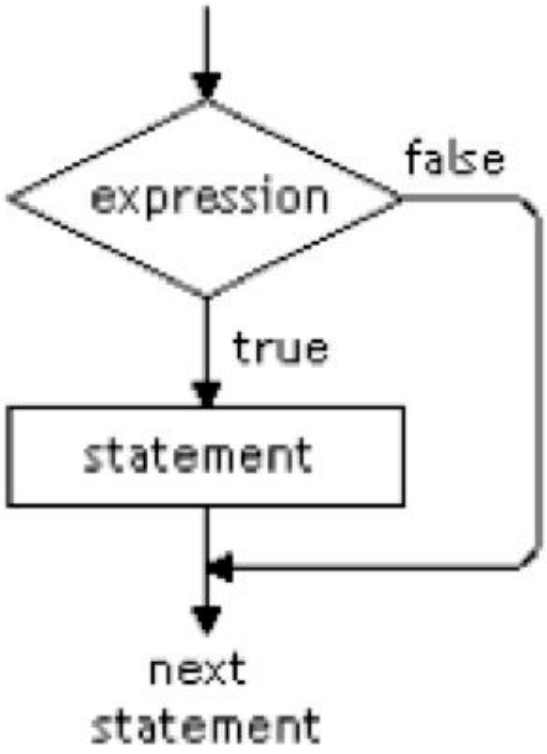
3. **A unary operator works on one operand, a binary operator works on two operands, but the ternary operator uses 3 operands. The ternary operator takes the first expression which is a statement that is going to be true or false, the ? mark is the operator. If the expression is true then it executes what is on the left side of the colon. If it is false then it executes what is on the right side of the colon.**
**Example: Expr_1 ? Expr_2: Expr_3**

T. Blanchard

1. Differences between if and if-else:

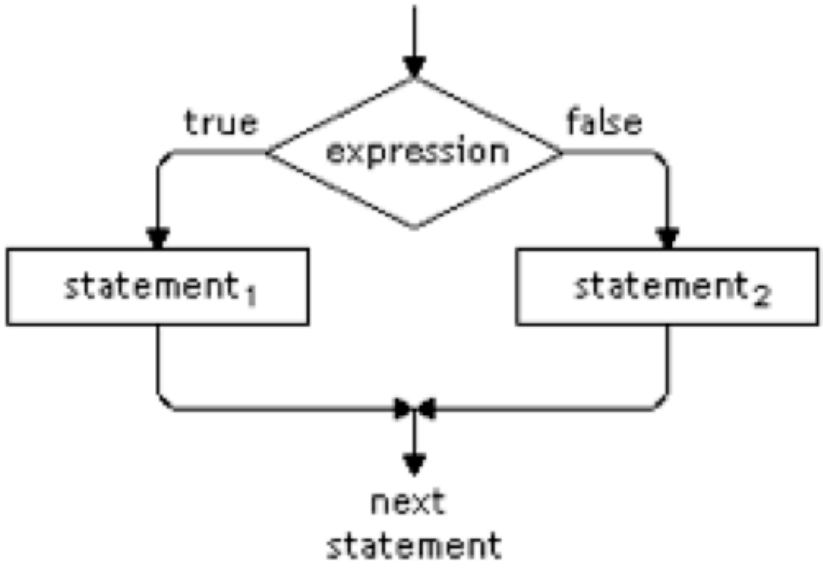
**if -** *If expression satisfied then run statement, if not, skip the statement*

```
if (expression) {
  statements;
}
```



**if-else -** *if expression satisfied, then run statement1 or if not satisfied, then run statement2*

```
if (expression) {
  statements1;
} else {
  statements2;
}
```




2. But if we will use only ifs, the program will show one of ifs and go directly to else. (!) Example: telephone directory (when we use if, if, if, if, else instead of if, if else, if else, if else, else; If a user will choose a number, program will print 'the department voicemail message' and 'wrong number command') Therefore, we have to use:

```
if (expression) {
  statements1;
if else(expression) {
   statements2;
} else {
  statements3;
}
```

instead of:

```
if (expression) {
  statements1;
if (expression) {
  statements2;  // The program will always print else. That's why we have to use if else
} else { // and we will get something from it (if expression == true) and else.
  statements3;  // we can use System.exit(0) to stop at some point (at some if)
} // but it is not a good idea at all
```

3. **And something very important to remember, if and if-else statements are not loops!**

M. Mardosz

1. **The == and = have different meanings in programming. The == is used to check if the expressions on both sides are equal. Example: *x == 2* checks if x equals the value two. The = is used assign a value to variable. Example: *x = 5* makes the x variable equal to five.**

2. If...else if...else is a decision structure which executes statements under certain circumstances. When writing the code, **there is no need to write a condition for the last else statement.**
Example: Any score below 60 will show the grade as "Fail".

```
if(score >= 90)
  grade = "A";
else if(score >= 60)
  grade = "Pass";
else
  grade = "Fail";
```

3. The conditional operator or ternary operator can be used for faster coding. This requires the use of the **?** and **:** symbols. Example: result = (n1 > n2) **?** option1 **:** option2;
This can be written in a normal if else decision structure as follows.

```
If (n1 > n2)
  result = option1;
else
  result = option2;
```

D. Starostka

Three things that I learned in week 7 are that an "if" statement only executes one statement, and that an "else" statement also only executes one statement. If we want them to execute multiple statements, we need to enclose those statements in curly braces because statements inside curly braces are considered to be one large statement.
Second, the = is the assignment operator and == is the equality operator. **In addition, you cannot use the equality operator to determine if two strings have the same values.**

Third, at the end of a multibranch if-else statement you should use an "else" instead of an "else if".
J. Gomez

1.) **When making an if statement I do NOT put a ; afterwards as that creates the "null statement" meaning if true, do nothing.**
2.) an if-else can only run one statement for its if and one for its else, but by using { } we can compound multiple statements into the if and/or else if need be. **To keep from getting brackets confused with one another, it's a good practice to make the pair, then label the closing bracket with a comment.**
3.) **You can not have an argument with more than two statements, a Boolean will not allow that. what you CAN do is use the connectos AND (&&) and OR (||) in order to make the argument you want. An example of the improper coding is: ( 0 < speedLimit < 100 ); it should be coded as ((0 < speedLimit) || (speedLimit < 100)).**
E. Herring

1. Only one statement, following the if statement will be executed if the condition is true.
2. **The symbol pair && means and in Java. We can use && to form a larger boolean expression out of two smaller boolean expression, both smaller statements must be true for the compound expression to be true. The symbol pair || means or in Java. As long as at least one of the smaller statements connected with the "or" is true, the compound expression is true.**
3. **When testing strings for equality, use either of the methods equals or equalsIgnoreCase.**

D. Mirdadi