File Edit View Help

File Edit View Help

Branching Statements
Part 02

File Edit View Help

Nested Statements

- Some statements can be nested within the body of another statement
- Nested if-statements are if-statements within the body of another if-statement
  - Same is true with if-else statements
  - Else statements still must have a corresponding if-statement
- Very useful when testing a combination of conditions

Syntax

```
if(<<Boolean expression>>)
{
        if(<<Boolean expression>>)
        {
                …
        }
}
```

Example

```
if(a == b)
{
        System.out.println(“a is equal to b”);
        if(b == c)
        {
                System.out.println(“b is equal to c”);
        }
        else
        {
                System.out.println(“b is not equal to c”);
        }
}
```

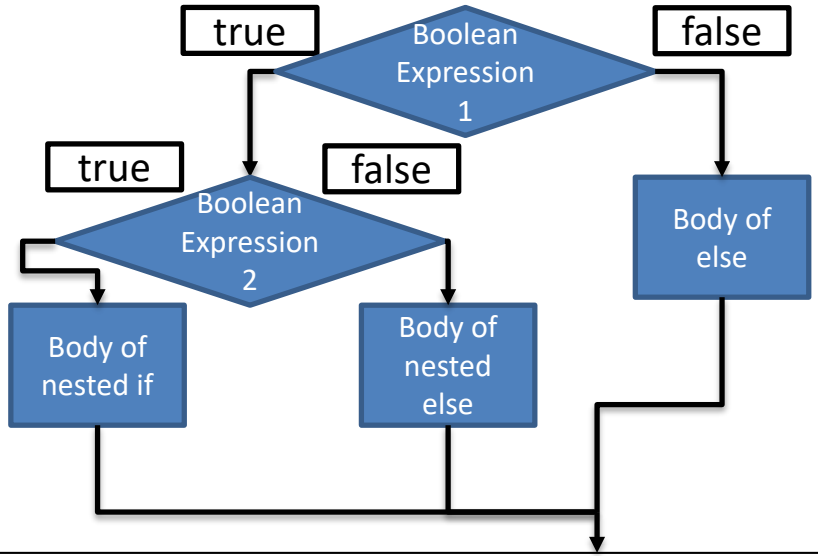
File Edit View Help

Nested Statements

Syntax

```
if(<<Boolean expression 1>>)
{
      if(<<Boolean expression 2>>)
      {
            //Body of nested if
      }
      else
      {
            //Body of nested else
      }
}
else
{
      //Body of else
}
```

General Nested If-Statement Flow Chart



File Edit View Help

Nested Statements

- You may omit the curly braces whenever an if-statement or else-statement has exactly ONE statement in its body
- An if-else is considered as one statement in Java
- This may cause logic errors if not careful
- It's a good idea to put the curly braces to clearly define the body of the statements
- The 2 examples do not have the same logic

```
if(a == b)
{
      if(b == c)
            e = f;
}
else
      e = g;
```

```
if(a == b)
     if(b == c)
          e = f;
else
     e = g;
```

File Edit View Help

Nested Statements

- You may omit the curly braces whenever an if-statement or else-statement has exactly ONE statement in its body
- An if-else is considered as one statement in Java
- This may cause logic errors if not careful
- It's a good idea to put the curly braces to clearly define the body of the statements
- The 2 examples do not have the same logic

```
if(a == b)
{
      if(b == c)
            e = f;
}
else
      e = g;
```

```
if(a == b)
     if(b == c)
          e = f;
     else
          e = g;
//Whoops!
```

File Edit View Help

Else-if Statement

- The else-if statement is a shorthand for an if-statement in the body of the else of another if-statement
- Else-if statements require first an if-statement
  - Just like an else-statement
- Can have multiple else-if statements in succession
- For the statements in an else-if to run the previous if's or else-if's must be false
  - Else-if's should only be used when the conditions are dependent

Syntax

```
if(<<Boolean expression>>)
{
        //Body of if-statement
}
else if(<<Boolean expression>>)
{
        //Body of else-if statement
}
else
{
        //Body of else statement
}
```

Example

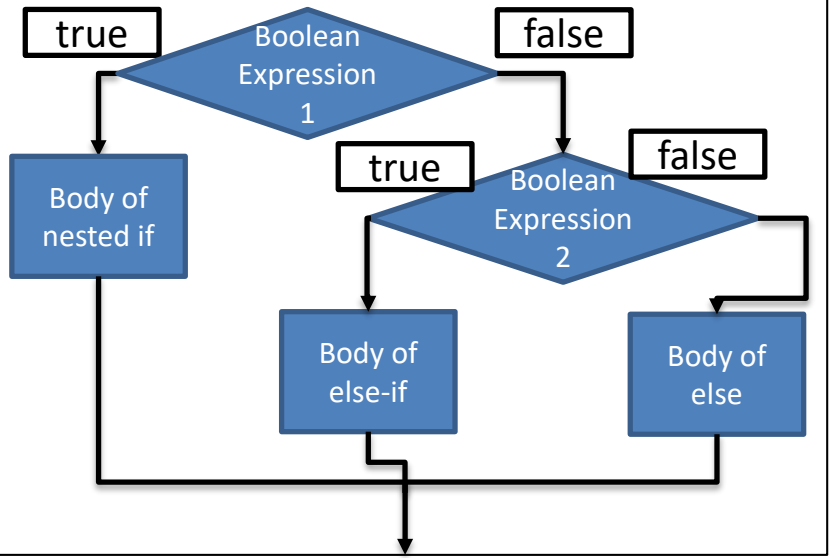
```
if(a == b)
{
        System.out.println("a is equal to b");
}
else if(a == c)
{
        System.out.println("a is equal to c, and not b");
}
else
{
        System.out.println("a is not equal to b or c");
}
```

Else-if Statements

Syntax

```
if(<<Boolean expression 1>>)
{
     //Body of if
}
else if(<<Boolean expression 2>>)
{
     //Body of else-if
}
else
{
     //Body of else
}
```

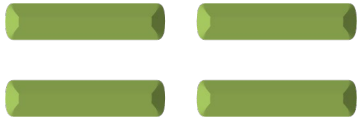
General Nested Else-If-Statement Flow Chart



File Edit View Help

Else-if Logic

```
if (Boolean_Expression_1)
{
        Statement_1
}
else if (Boolean_Expression_2)
{
        Statement_2
}
else if (Boolean_Expression_3)
{
    Statement_3
}
else
{
    Default_Statement
}
```

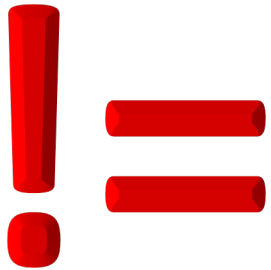


```
if (Boolean_Expression_1)
{
          Statement_1
}
else
{
   if (Boolean_Expression_2)
   {
             Statement_2
   }
   else
   {
        if(Boolean_Expression_3)
        {
            Statement_3
        }
        else
        {
            Default_Statement
        }
   }
}
```

File Edit View Help

Else-if Logic

```
if (Boolean_Expression_1)
{
        Statement_1
}
else if (Boolean_Expression_2)
{
        Statement_2
}
else if (Boolean_Expression_3)
{
   Statement_3
}
else
{
   Default_Statement
}
```



```
if (Boolean_Expression_1)
{
        Statement_1
}
if (Boolean_Expression_2)
{
        Statement_2
}
if (Boolean_Expression_3)
{
   Statement_3
}
else
{
   Default_Statement
}
```

File Edit View Help

File Edit View Help

Let's Make a Decision!

File Edit View Help

File Edit View Help

Contestant 1

File Edit View Help

Contestant 1

Contestant 2

File Edit View Help

Contestant 1

Contestant 2

Box #1

Box #2

Box #3

File Edit View Help

Contestant 1

Contestant 2

File Edit View Help

Contestant 1

Contestant 2

Box #1

Box #2

Box #3

File Edit View Help

Contestant 1

Contestant 2

Box #1

Box #3

File Edit View Help

Contestant 1

Contestant 2

Box #3

File Edit View Help

Contestant 1

Contestant 2

File Edit View Help

THANK YOU FOR PLAYING!

File Edit View Help

Conditional Operator

- The “?” and the “:” together are called the conditional opertator
  - AKA ternary operator
  - AKA “One-line if”
- The value after the “?” corresponds to the body of the if-statement
- The value after the “:” corresponds to the body of the else-statement
- Can be useful for writing print statements

Syntax

```
<<var>> = (<<Boolean expression>>)?<<value 1>>:<<value 2>>;
```

Example

```
maxValue = (value1 >= value2) ? Value1 : value2;
```

File Edit View Help

Exit Method

- The exit method immediately stops the program
- Can be used when there arises an error that will prevent a program from working
- The value “0” is generally used when the program is exiting under normal conditions

Syntax

```
System.exit(<<Integer Value>>);
```

Example

```
if(!validInput)
{
      System.exit(0);
}
```

File Edit View Help

Expanded Precedence Rules

Highest Precedence

Unary Operators	+, -, ++, --, !
Binary Arithmetic	* , /, %
Binary Arithmetic	+, -
Boolean Operators	<, >, <=, >=
Boolean Operators	==, !=
Boolean Operator	&
Boolean Operator	|
Boolean Operator	&&
Boolean Operator	||

Lowest Precedence

File Edit View Help

Short Circuit Evaluation

- Sometimes only a part of a Boolean expression needs to be evaluated to determine the entire value
  - If the first operand of an && is false, then the entire expression is false
  - If the first operand of an || is true, then the entire expression is true
- Sometimes called Lazy Evaluation
- Very efficient and sometimes necessary

Example 1

```
boolean a = false;
boolean b = true;
if(a && b)//b is never checked
{

}
```

Example 2

```
boolean a = true;
boolean b = false;
if(a || b)//b is never checked
{

}
```