Lecture 4: Java Control Statement

By Ahmed E. Khaled
Recap to previous lecture!

- What is a Variable?
- What are Java data-types?
example

• boolean result = true;
• char capitalC = 'C';
• byte b = 100;
• short s = 10000;
• int i = 100000;
• double d1 = 123.4;
• float f1  = 123.4f;
Lecture 04, Java control statement

• Java Selection statements.
• Relational and logical operations.
• How to get input from user?
Java Block

• A block is a group of zero or more statements between balanced brace.

```java
{
    Some code statements
}
```
Global vs. local w.r.t Block

```java
int variable1 = 8; //global variable declaration
{
    //block scope
    int variable2 = 10; //local variable declaration
    System.out.println(variable1); //OK
    System.out.println(variable2); //OK
}
//end of block scope

System.out.println(variable1); //OK
System.out.println(variable2); //compilation error (out scope)
```
Nested blocks

```java
int variable1 = 8;
{
    int variable2 = 7;
    {
        int variable3 = 6;
        System.out.println(variable1);
        System.out.println(variable2);
        System.out.println(variable3);
    }
    System.out.println(variable1);
    System.out.println(variable2);
    System.out.println(variable3);  //Compilation Error
}
System.out.println(variable1);
System.out.println(variable2);  //Compilation Error
System.out.println(variable3);  //Compilation Error
```
Java Selection

The program can decide which statements to execute based on a condition, this is what is called selection statements.

If you enter a negative value for radius, your program can displays an invalid result instead of computing the area.
if (Boolean expression) {
    do something
}
else {
    do something else
}
Boolean Expressions

Selection statements use conditions that are *Boolean expressions*.

Boolean expression is an expression that evaluates to a Boolean value: *true or false*.

How do you compare two values, and check whether a radius is greater than 0, equal to 0, or less than 0?

Java provides six relational operators (also known as comparison operators) to compare two values.
## Relational operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Java Operator</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than</td>
<td>&lt;</td>
<td>If (radius &lt; 5)</td>
</tr>
<tr>
<td>Less than or equal</td>
<td>&lt;=</td>
<td>If (radius &lt;= 5)</td>
</tr>
<tr>
<td>Greater than</td>
<td>&gt;</td>
<td>If (radius &gt; 5)</td>
</tr>
<tr>
<td>Greater than or equal</td>
<td>&gt;=</td>
<td>If (radius &gt;= 5)</td>
</tr>
<tr>
<td>Equal to</td>
<td>==</td>
<td>If (radius == 5)</td>
</tr>
<tr>
<td>Not equal to</td>
<td>!=</td>
<td>If (radius != 5)</td>
</tr>
</tbody>
</table>
int radius = 10;
float area;

if (radius < 0) {
    System.out.println("Incorrect input");
}
else {
    area = radius * radius * 3.14159;
    System.out.println("Area is " + area);
}
If statement

An if statement is a construct that enables a program to specify alternative paths of execution.

Java has several types of selection statements:
• one-way if statements
• two-way if-else statements
• nested if statements
• multi-way if-else statements
• switch statements
One-way if statement

Executes an action if and only if the condition is \textit{true}.

```
......
......
......
if (boolean-expression) {
    statement(s);
}
......
......
```
Two –way if statements

A one-way if statement performs an action if the specified condition is true. If the condition is false, nothing is done.

But what if you want to take alternative actions when the condition is false, you can use a two-way if-else statement.

```java
if (boolean-expression) {
    statement(s)-for-the-true-case;
}
else {
    statement(s)-for-the-false-case;
}
```
Nested if and Multi-Way if-else

The statement in an if or if-else statement can be any legal Java statement, including another if or if-else statement.

The inner if statement is said to be nested inside the outer if, This inner if statement can contain another if statement (no limit to the depth of the nesting).
if (i > k) {
    if (j > k) {
        System.out.println("i and j are greater than k");
    } else {
        System.out.println("i is less than or equal to k");
    }
}
if (var1 > var2) {
    if (var2 > var3) {
        System.out.println("var1 > var2 > var3");
    }
}
else {
    if (var4 > var5) {
        System.out.println("var1 <= var2 and var4 > var5");
    }
    else {
        System.out.println("var1 <= var2 and var4 <= var5");
    }
}
Try to guess the output

Suppose x = 3 and y = 4, guess the output:
if (x > 2) {
    if (y > 2) {
        int z = x + y;
        System.out.println(" summation is " + z);
    }
}
} Else{
    System.out.println("x is " + x);
}
Suppose $x = 2$ and $y = 2$, guess the output:

```java
if (x > 2) {
    if (y > 2) {
        z = x + y;
        System.out.println("z is " + z);
    }
}

Else{
    System.out.println("x is " + x);
}
```
Try to guess the output

Suppose \( x = 3 \) and \( y = 2 \), guess the output:

```java
if (x > 2) {
    if (y > 2) {
        z = x + y;
        System.out.println("z is "+z);
    }
}

Else{
    System.out.println("x is "+x);
}
```
Try to guess the output

Score is 80, 70, 60?

if (score >= 60.0)
    System.out.println("D");
else if (score >= 70.0)
    System.out.println("C");
else if (score >= 80.0)
    System.out.println("B");
else if (score >= 90.0)
    System.out.println("A");
else
    System.out.println("F");
Logical operations

- Sometimes, the program requires a combination of several conditions to trigger conditional statement.

- You can use logical operators to combine these conditions to form a compound Boolean expression.

- Logical operators, also known as Boolean operators, operate on Boolean values to create a new Boolean value.
<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>!</td>
<td>not</td>
<td>logical negation</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>and</td>
<td>logical conjunction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>^</td>
<td>exclusive or</td>
<td>logical exclusion</td>
</tr>
</tbody>
</table>
The *and* (`&&`) of two Boolean operands is true if and only if *both* operands are true.

The *or* (`||`) of two Boolean operands is true if at least one of the operands is true.

<table>
<thead>
<tr>
<th><code>p_1</code></th>
<th><code>p_2</code></th>
<th><code>p_1 &amp;&amp; p_2</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

| `p_1` | `p_2` | `p_1 || p_2` |
|-------|-------|--------------|
| false | false | false        |
| false | true  | true         |
| true  | false | true         |
| true  | true  | true         |
The not (!) of a Boolean operands is true if the operand is false and vice versa.

The *exclusive or* (^) operator is true if and only if the two operands *have different Boolean values*.

<table>
<thead>
<tr>
<th>p</th>
<th>!p</th>
<th>p1 ^ p2</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false true</td>
<td>true false</td>
<td>false true</td>
</tr>
<tr>
<td>true false</td>
<td>true true</td>
<td>true false</td>
</tr>
</tbody>
</table>
Try to guess the output

Boolean flag = false;
If(!flag){
    System.out.println(“this is the if block !”)
}else{
    System.out.println(“this is the else block !”)
}
public static void main(String args[]) {
    boolean a = true;
    boolean b = false;

    If( a && b){
        System.out.println("a && b is true");
    }

    If( a || b){
        System.out.println("a || b is true");
    }
}
Try to guess the output

```java
public static void main(String args[]) {
    boolean a = true;
    boolean b = false;

    if (a && (!b)){
        System.out.println("a && (!b) is true");
    }

    if (! (a && b ) ){
        System.out.println(" ! (a && b) is true");
    }
}
```
Try to guess the output

```java
public static void main(String args[]) {
    boolean a = true;
    boolean b = false;

    if (a ^ b) {
        System.out.println("a ^ b is true");
    }
    if (a ^ (!b)) {
        System.out.println("a ^ (!b) is true");
    }
}
```
Try to guess the output

```java
Int num1 = 3;
Int num2 = 5;
Int num3 = 7;
If (num1 == 3 || num3=9 ){
    System.out.println(“here we are”);
}
If (num1 == 9 && ( num2 == 2 || num3 == 7 ) ){
    System.out.println(“here we are”);
}
```