Branching Statements
Part 01
Flow of control is the order in which a program performs actions.

A **branching statement** chooses between two or more possible actions.

A **loop statement** repeats an action until a stopping condition occurs.

Flow Charts diagram the flow of a program
- Boxes are Statements
- Diamonds are Decisions
  - True branch
  - False branch
- Arrows indicate the flow of statements and decisions
- Pseudocode is mostly used

**Flow Chart Example**

1. Prompt user for input
2. input = get user input
3. input is valid?
   - true: Do the Thing
   - false: Don’t the Thing
• If-statement
• If the Boolean expression is “true” then the body of the if-statement is executed, and otherwise is ignored
• Putting curly braces “{}” to denote the body of the if-statement is strongly encouraged
• Do not put a semicolon “;” after the parenthesis
  – It will ignore the Boolean expression
• Spoken
  – “if this is true then do this”

Syntax

if(<<Boolean expression>>) {
   //Body of the if-statement
}
//Outside Body of the if-statement

Examples

if(a == b) {
   System.out.println("a is equal to b");
}
Syntax

if(<<Boolean expression>>) {
    //Body of the if-statement
}
//Outside Body of the if-statement
• Else-statement
• Requires a proceeding if-statement
  – If-statements do not require an else-statement
• If the Boolean expression is “false” then
  the body of the else-statement is executed, and otherwise is ignored
• Putting curly braces “{}” to denote the body of
  the else-statement is strongly encouraged
• Spoken:
  – “if this is true then do this, otherwise (else) do that”

**Syntax**

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

**Examples**

```java
if(a == b) {
    System.out.println("a is equal to b");
} else {
    System.out.println("a is not equal to b");
}
```
**Syntax**

```plaintext
if(<<Boolean expression>>)  
{
    //Body of the if-statement
}  
else  
{
    //Body of the else-statement
}  
//Outside of if and else
```
• True or False Value
• Common Boolean Operators
  – “==” : Equal to
  – “!=“ : Not Equal
  – “<“ : strictly less than
  – “>” : strictly greater than
  – “<=“: less than or equal to
  – “>=“: greater than or equal to

Syntax

<<value>> <<Boolean operator>> <<value>>;

Examples

boolean a = 12 > 3;
if(a)//Or a == true
{
    System.out.println(“Here”);
}
else
{
    System.out.println(“Not here”);
}
• Combines multiple Boolean expressions
• Common Compound Boolean Expression Operators
  – “&&” : AND – both must be true to yield true
  – “||” : OR – only one must be true to yield true

Syntax

```
<<Boolean expression>> <<operator>> <<Boolean expression>>;
```

Examples

```java
boolean a = 2 != 0 && 12 > 3;
if(a)//Or a == true
{
    System.out.println("Here");
}
else
{
    System.out.println("Not here");
}
```
## Truth Table

| A   | B    | A && B | A || B |
|-----|------|--------|--------|
| TRUE| TRUE | TRUE   | TRUE   |
| TRUE| FALSE| FALSE  | TRUE   |
| FALSE| TRUE | FALSE  | TRUE   |
| FALSE| FALSE| FALSE  | FALSE  |
- The NOT operator “!” is used to negate the value of a Boolean expression

<table>
<thead>
<tr>
<th>Negated Expression</th>
<th>Equivalent Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>!(A &lt; B)</td>
<td>(A &gt;= B)</td>
</tr>
<tr>
<td>!(A &lt;= B)</td>
<td>(A &gt; B)</td>
</tr>
<tr>
<td>!(A &gt; B)</td>
<td>(A &lt;= B)</td>
</tr>
<tr>
<td>!(A &gt;= B)</td>
<td>(A &lt; B)</td>
</tr>
<tr>
<td>!(A == B)</td>
<td>(A != B)</td>
</tr>
<tr>
<td>!(A != B)</td>
<td>(A == B)</td>
</tr>
<tr>
<td>!(A &amp;&amp; B)</td>
<td>(A</td>
</tr>
<tr>
<td>!(A</td>
<td></td>
</tr>
</tbody>
</table>

- The NOT operator “!” is used to negate the value of a Boolean expression
Example
<table>
<thead>
<tr>
<th>Using “==”</th>
<th>Great for comparing memory addresses of Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>• The operator “==” is great for determining if two values are equal in some cases, but not all</td>
<td></td>
</tr>
<tr>
<td>• Great to use when comparing integer values</td>
<td></td>
</tr>
<tr>
<td>• Not great to use when comparing floating-point values</td>
<td></td>
</tr>
<tr>
<td>• Not great for comparing contents of an object</td>
<td></td>
</tr>
<tr>
<td>• Great for comparing memory addresses of Objects</td>
<td></td>
</tr>
<tr>
<td>• Check if objects are NULL</td>
<td></td>
</tr>
<tr>
<td>• Check if two identifiers reference the same place in memory</td>
<td></td>
</tr>
<tr>
<td>• Not great for comparing contents of an object</td>
<td></td>
</tr>
<tr>
<td>• Use the “.equals()” method instead</td>
<td></td>
</tr>
</tbody>
</table>

- Round of errors
- Use a combination of >= and <= with a tolerance
Alternate Example