# Conditional Logic and Block Statements

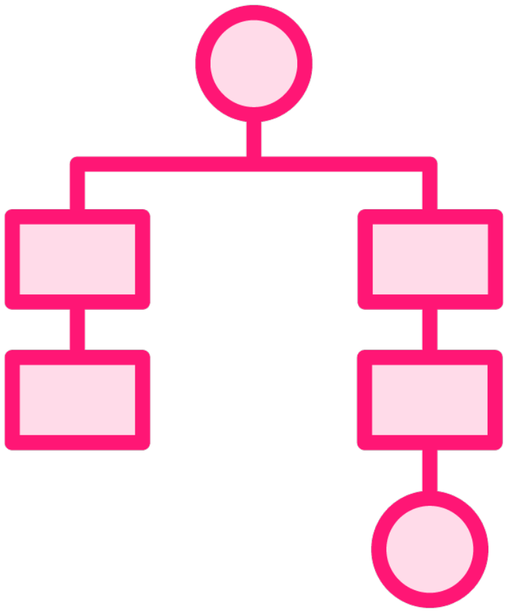
**Jim Wilson**

Mobile Solutions Developer & Architect

@hedgehogjim | jwhh.com

# Overview

- Relational operators
- Conditional assignment
- If-else
- Chaining if-else
- Logical operators
- Block statements
- Switch



**Conditional logic**

- Perform a test
- Perform action based on test result

# Relational Operators

| | Operator | Integer, Floating Point Example | Character Example | Boolean Example |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

```java
int value1 = 7;
int value2 = 5;
int maxValue = value1 > value2 ? value1 : value2 ;
System.out.println(maxValue);
```

## Conditional Assignment

Return a value based on the result of a condition

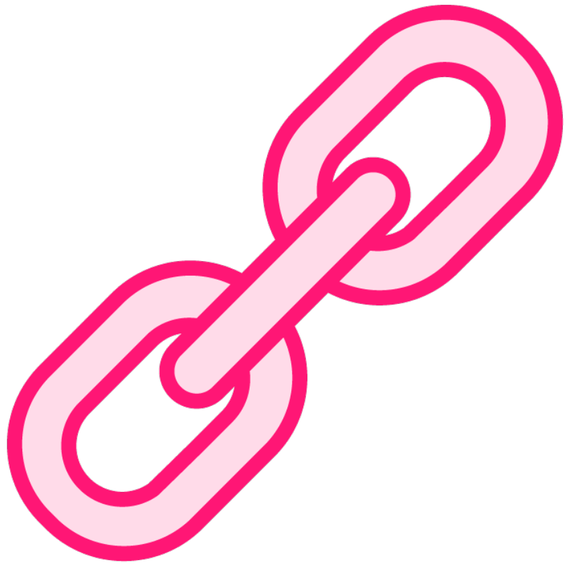
condition

```
int value1 = 10;
int value2 = 4;
if (value1 > value2)
     System.out.println(“value 1 is bigger”);
else
     System.out.println(“value 1 is not bigger”);
```

## If-else

**An if statement conditionally executes a statement**
**Else clause executes a statement when condition is false**
**- Else clause is optional**

```
if ( condition )
  true-statement ;
else
  false-statement ;
```



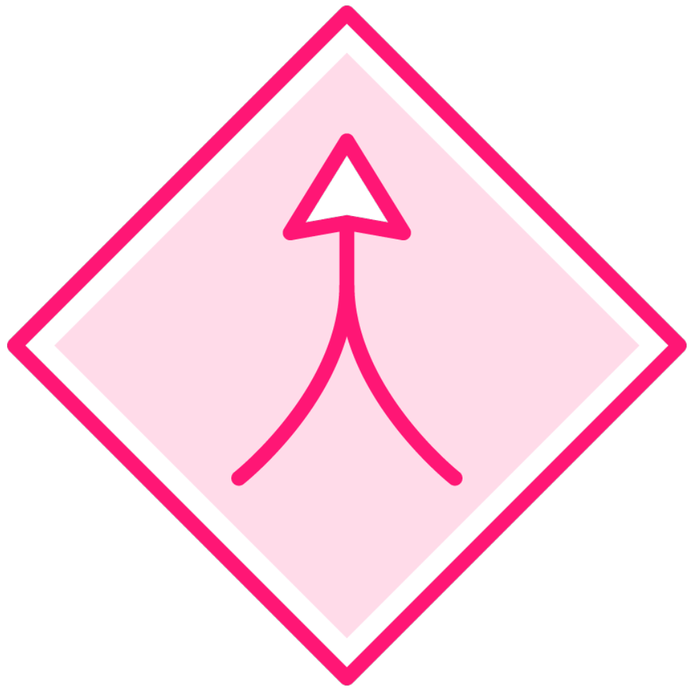
## Chaining if-else

- Evaluated in order top-to-bottom
- First to test true is executed

```
if ( condition-1 )
  true-statement-1 ;
else
  true-statement-2 ;
          .
          .
          .
else if ( condition-N )
  true-statement-N ;
else
  false-statement ;
```

# Chaining if-else

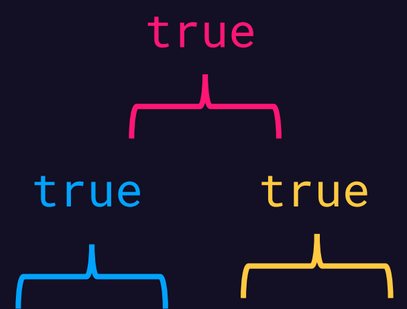
```
int value1 = 10;
int value2 = 40;
if (value1 > value2)
    System.out.println(“value 1 is bigger”);
else
    System.out.println(“value 2 is bigger”);
else
    System.out.println(“value 1 and value 2 are equal”);
```



## Logical operators

- Produce a single true or false result from two true or false values
- May combine two relational tests
- May combine two Boolean variables

```
int a = 20, b = 14, c = 5;
```

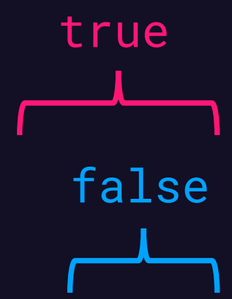



```
if ( a > b & b > c )
     System.out.println(“a is greater than c”);
```

## Logical Operators

| | Operator | What Resolves to True |
|---|---|---|
| And | & | |
| | | |
| | | |
| | | |

```
boolean done = false;
```

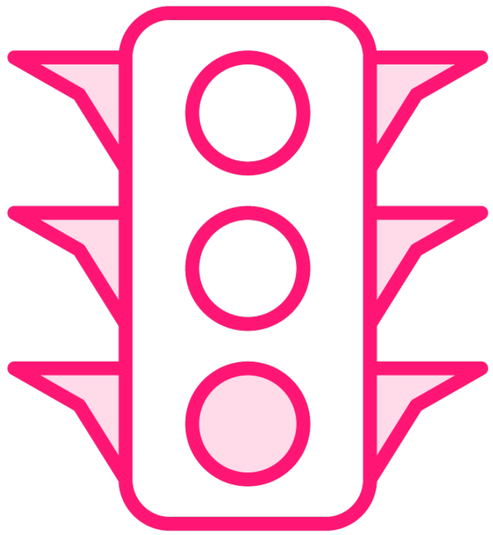



```
if ( ! done )
     System.out.println(“Keep going!”);
```

## Logical Operators

| | Operator | What Resolves to True |
|---|---|---|
| And | & | true & true |
| Or | \| | |
| | | |
| | | |

# Conditional Logical Operators

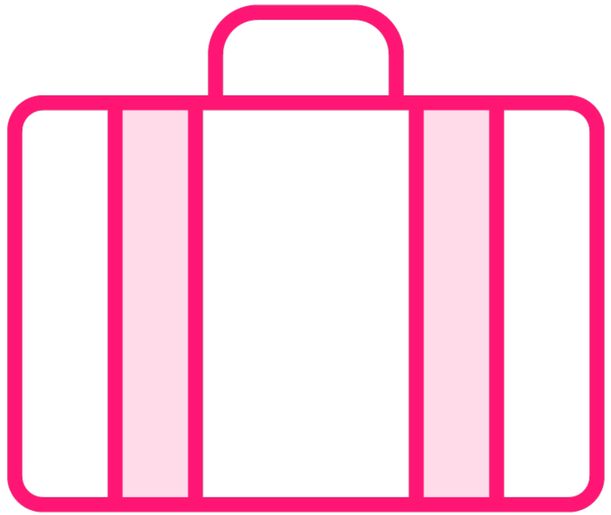


**Similar to standard logical operators**

**Right side executes only when needed**

- && executes right only when left is true
- || executes right only when left is false

| | Operator | What Resolves to True |
|---|---|---|
| And | && | |
| Or | \|\| | |



## Block statement

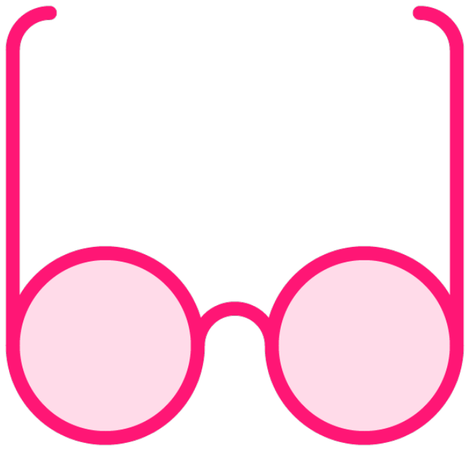
- Groups statements together
- Creates a compound statement
- Enclose statements in opening and closing brackets

```
{
  statement-1;
  statement-2;
       .
       .
       .
  statement-N;
}
```

# Block Statement

```
int v1 = 10, v2 = 4;
final int diff;
if (v1 > v2)
     diff = v1 – v2;
     System.out.println(“v1 is bigger than v2, diff = “ + diff);
}
else
     diff = v2 – v1;
     System.out.println(“v1 is not bigger than v2, diff = “ + diff);
}
```

# Block Statement and Variable Scope



**Variable scope**

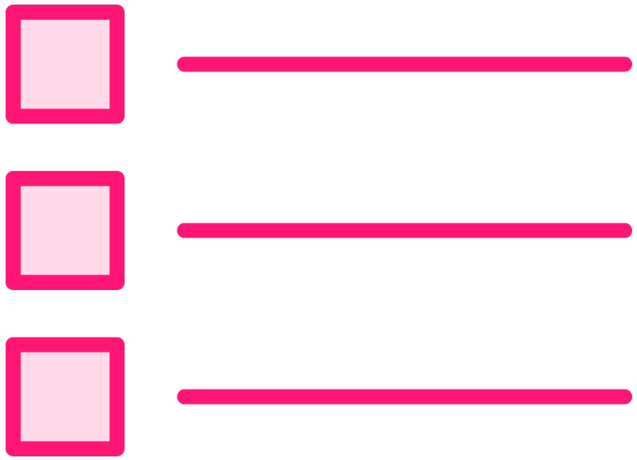
- Describes range of visibility

**Variable declared within a block statement**

- Scope limited to that block
- In other words, the variable is not visible outside of the block

# Block Statement and Variable Scope

```
double students = 30.0d, rooms = 4.0d;
if(rooms > 0.0d) {
    System.out.println(students);
    System.out.println(rooms);
    double avg = students / rooms
    System.out.println(avg);
}
System.out.println(avg);
```

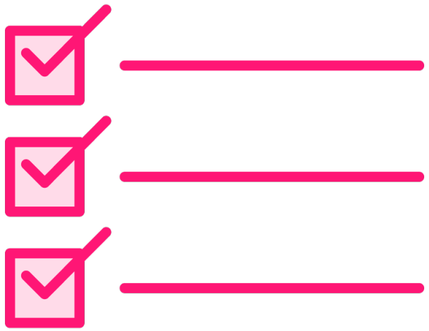


## Switch

- Test value against multiple matches
- Transfers control based on match

```
switch (value-to-test) {
  case matching-value-1:
    statements
    break;
  .
  .
  .
  case matching-value-N:
    statements
    break;
  default:
    statements
}
```

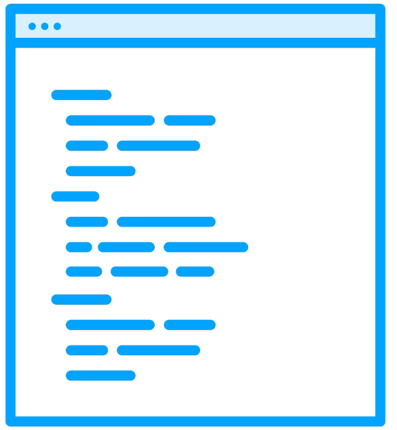
# Using Switch



**Primitive types supported**

byte, short, int, long

char



**A match can have multiple statements**

End each match with break

Otherwise will “fall through”
to next match

# Summary

## Conditional assignment

- Return value based on condition

## If-else

- Conditionally executes a statement
- Else clause is optional
- Can chain if-else statements together

# Summary

## Relational operators

- Compare one value to another

## Logical operators

- Produce a single true or false result from two true or false values

## Conditional logical operators

- Similar to standard logical operators
- Only execute right side when needed

# Summary

## Block statement

- Group statements together
- Variables declared within a block are not visible outside of block

## Switch

- Test value against multiple matches
- Transfers control based on match
- Be sure to end each match with break