

Module 1 - Lecture 3

Expressions, Statements, Blocks, and Branches



Review

- Java overview
- IntelliJ overview
- Variables
 - Declaration vs Initialization
 - Naming best practices
- Arithmetic operators and expressions
- Type conversion



What is a Program?

- Data
- **Behavior** -> Today's focus



Expressions and Statements

- An **expression** is a construct made up of variables, operators, and method invocations.
- An expression **evaluates to a single value**.
- A **statement** forms a complete unit of execution.
- Think of expressions as words and statements as sentences. In that case, **code blocks** are paragraphs.



Code Blocks

- Code that belongs together can be written in blocks.
- What does this do?

```
{  
    int length = 5;  
    int width = 10;  
    int area = length * width;  
}
```



Scope

A variable's **scope** defines where in the program that the variable exists (i.e. can be referenced). When code execution reaches a point where a variable is no longer referenceable, the variable is said to be **out of scope**.

Rules of Scope:

1. Variables declared inside of a method or block `{..}` are only available within that block.
2. Blocks can be nested within other blocks. Therefore, if a variable is declared outside of a block, it is accessible within the inner block.



Methods

- A **method** is a named block of code.
- A method can take multiple parameters and return zero or one result.
- A method has a declaration, which is made up of a few components in a certain order.
 - The name and parameters make up a **method signature**.

<Access Modifier> <Return Type> <Name> <Parameters>

Examples:

```
public double divide (int num1, int num2)
```

```
public void main (String[] args)
```



Let's Code!

Boolean Expressions

In programming, we often want to conditionally execute sections of code. Before we can do that we need to know how to check when we should run a section of code.

A **boolean expression** is an expression that produces a boolean value (**true** or **false**) when evaluated.



Comparison Operators

Given X = 5

OPERATOR	DESCRIPTION	COMPARING	YIELDS
==	IS EQUAL TO	X == 8	FALSE
		X == 5	TRUE
!=	IS NOT EQUAL TO	X != 8	TRUE
		X != 5	FALSE
>	IS GREATER THAN	X > 8	FALSE
<	IS LESS THAN	X < 8	TRUE
>=	GREATER THAN OR EQUAL TO	X >= 8	FALSE
<=	LESS THAN OR EQUAL TO	X <= 8	TRUE



Boolean (Logical) Operators

NOT

A	!A
FALSE	TRUE
TRUE	FALSE

**BOOLEAN
EXPRESSION**

A statement which evaluates to a single boolean value.

Given A is TRUE and B is FALSE,
Evaluate the expression

AND

A	B	A && B
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

OR

A	B	A B
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

(A && B) || (A && !B)



(TRUE && FALSE) || (TRUE && !FALSE)



(TRUE && FALSE) || (TRUE && TRUE)



FALSE || (TRUE && TRUE)



FALSE || TRUE



TRUE



Boolean (Logical) Operators cont...

BOOLEAN EXPRESSION

A statement which evaluates to a single boolean value.

XOR

A	B	$A \wedge B$
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE



Challenge (breakout)

Let's assume a car can be powered by gasoline, diesel, or electricity.

Using boolean expressions, how would you determine if a car is a hybrid
i.e. at least two power sources are used?

No conditional logic allowed!

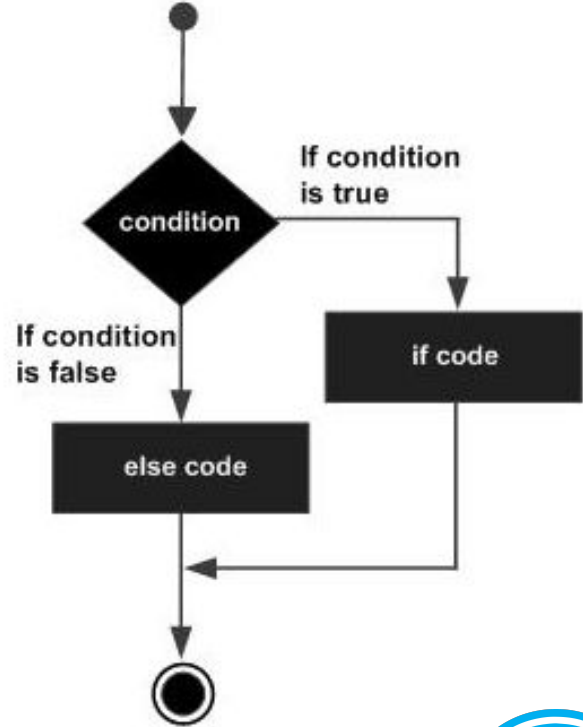


If Statements

Conditional blocks allow a program to take a different path depending on some condition(s) determined while the program runs.

Syntax:

```
if (boolean expression) {  
    <if_code_here>  
}  
else {  
    <else_code_here>  
}
```



Let's Code!

QUESTIONS?

