# Java Foundations

**5-2**
**Understanding Conditional Execution**
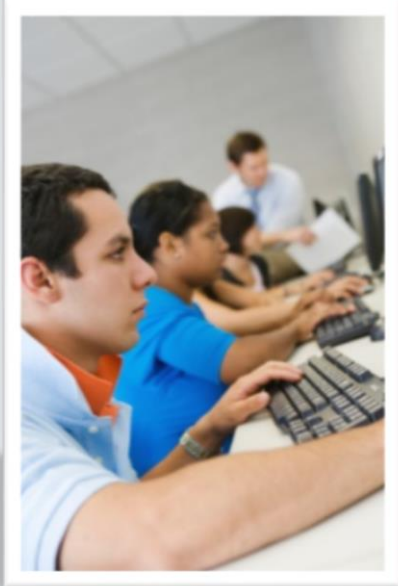
# Objectives

This lesson covers the following objectives:

- Describe conditional execution

- Describe logical operators

- Understand "short circuit" evaluation of logical operators

- Build chained `if` constructs

3

3

# Topics

- **Logical Operators**
- Short-Circuit Evaluation
- Ternary Operator
- Build Chained `if` Constructs

Boolean Expressions and if/else Constructs

Understanding Conditional Execution

switch Statement

Section 5
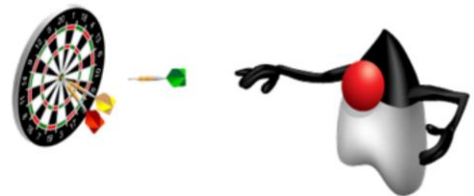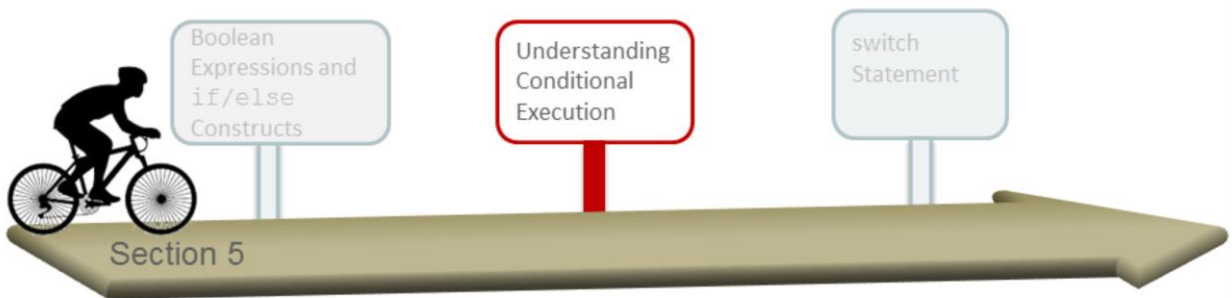
# When Multiple Conditions Apply

- What if a particular action is to be taken only if several conditions are true?

- Consider the scenario where a student is eligible for scholarship if the following two conditions are met:
  - Grade should be >= 88.
  - Number of days absent = 0.

# Handling Multiple Conditions

- Relational operators are fine when you're checking only one condition.

- You can use a sequence of `if` statements to test more than one condition.

```
if (grade >= 88) {

    if (numberDaysAbsent == 0) {

        System.out.println("You qualify for the scholarship.");

    }

}
```

JFo 5 -2
Understanding Conditional Execution

6

# Handling Multiple Conditions: Example

As demonstrated in the example:

- The sequence of `if` statements is hard to write, harder to read, and becomes even more difficult as you add more conditions.

- Fortunately, Java has an easy way to handle multiple conditions: **logical operators**.

JFo 5 -2
Understanding Conditional Execution

7

# Java's Logical Operators

You can use Java's three logical operators to combine multiple boolean expressions into one `boolean` expression.

| Logic Operator | Meaning |
|---|---|
| && | AND |
| \|\| | OR |
| ! | NOT |

# Three Logical Operators

| Operation | Operator | Example |
|---|---|---|
| If one condition AND another condition | && | `int i = 2;`<br>`int j = 8;`<br>`((i < 1) && (j > 6))` |
| If either one condition OR both conditions | \|\| | `int i = 2;`<br>`int j = 8;`<br>`((i < 1) \|\| (j > 10))` |
| NOT | ! | `int i = 2;`<br>`(!(i < 3))` |

The table in the slide lists the logical operators in the Java programming language. All examples yield a boolean result of false.

In this example, you use the logical AND operator because both boolean expressions must be true to make the student eligible for a scholarship.

Logical AND operator:

- Combined condition is true if and only if both boolean expressions are true.
- Combined condition is false if either or both of the boolean expressions are false.

# Logical AND Operator: Example

```
public static void main(String[] args) {
    int numberDaysAbsent = 0;
    int grade = 95;
    if (grade >= 88 && numberDaysAbsent == 0) {
        System.out.println("You qualify for the scholarship.");
    }
    else {
        System.out.println("You do not qualify for the scholarship.");
    }
}
```

*Evaluates to true if both boolean expressions are true.*

This example illustrates the logical AND operator. For the output to be displayed as "You qualify for the scholarship," the two conditions must be true. That is, the grade must be greater than or equal to 88, and the number of days absent must be equal to zero.

## Logical OR Operators

- Consider a scenario where a student is eligible for a sports team if one of the following two conditions are met:
  - Grade >= 70
  - Number of days absent < 5

- In this case, you can use the logical OR operator to connect the multiple boolean expressions.

```
grade >=70  ||  numberDaysAbsent < 5
```
Boolean Expression 1    Logical Operator    Boolean Expression 2

Combined condition is true if either or both of the boolean expressions are true.

Combined condition is false if both of the boolean expressions are false.

# Logical OR Operators: Example

```java
public static void main(String[] args) {
    int numberDaysAbsent = 3;
    int grade = 85;
    if (grade >= 70 || numberDaysAbsent < 5) {
        System.out.println("You qualify for a sports team");
    }
    else {
        System.out.println("You do not qualify for a sports team");
    }
}
```

*Evaluates to true if either of the boolean expressions evaluates to true.*

This example illustrates using the logical OR operator. In this example, "You qualify for a sports team" is displayed even if one of the conditions is true. That is, the grade must be >= 70 or the number of days absent   must be less than five.

This example illustrates the logical ! operator. Because the grade is equal to 65, `!madeFreeTutor` is true because `madeFreeTutor` is false.

The combined expression evaluates to true and the following output is displayed: `"You qualify for free tutoring help."`

## Logical NOT Operators

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

    int numberDaysAbsent = 2;

    int grade = 65;

    boolean madeFreeTutor = grade >= 88;

    if (!madeFreeTutor && numberDaysAbsent < 3) {

        System.out.println("You qualify for free tutoring
        help");

    }

}
```

This example illustrates the logical ! operator. Because the grade is equal to 65, `!madeFreeTutor` is true because `madeFreeTutor` is false**.**

The combined expression evaluates to true and the following output is displayed: **"**`You qualify for free tutoring help.`**"**
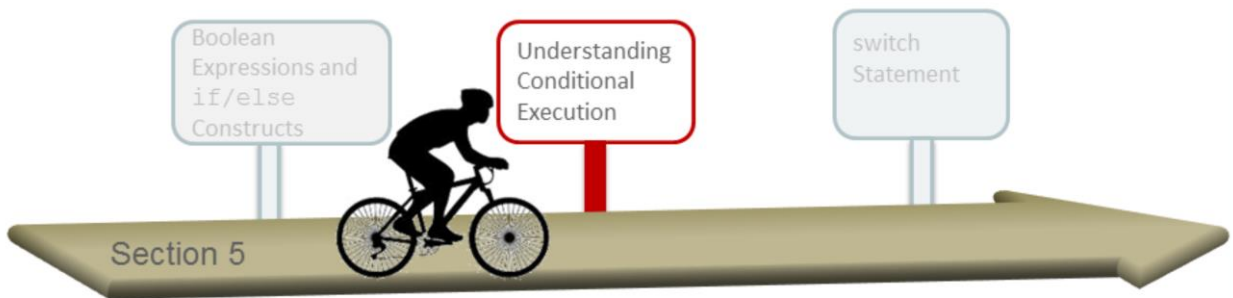
# Exercise 1

- Import and open the ConditionalEx project.

- Modify WatchMovie.java to watch a movie that meets the following two conditions:

  - The movie price is greater than or equal to $12.
  - The movie rating is equal to 5.

- Display the output as "I'm interested in watching the movie."

- Else display the output as "I am not interested in watching the movie."

# Topics

- Logical Operators

- **Short-Circuit Evaluation**

- Ternary Operator

- Build Chained `if` Constructs

Boolean Expressions and `if/else` Constructs

Understanding Conditional Execution

switch Statement

Section 5

17

# Skipping the Second AND Test

- The && and || operators are short-circuit operators.

- If the 1st expression (on the left) is false, there is no need to evaluate the 2nd expression (on the right).

```
b = (x != 0) && ((y / x) > 2);
```

Left Expression          Right Expression

The evaluation is performed in left-to-right order and halts as soon as the result is known. This means that the expression on the right side won't be evaluated if it isn't necessary.

# Skipping the Second AND Test

```
b = (x != 0) && ((y / x) > 2);
       Left           Right
    Expression     Expression
```

- If $x$ is 0 then $(x \ != \ 0)$ is false.

- For the `&&` operator, because it doesn't matter whether $((y/x)>2)$ is `true` or `false`, the result of this expression is `false`.

- So Java doesn't continue evaluating $((y/x)>2)$.

# Skipping the Second OR Test

- If the 1st expression (on the left) is `true`, there is no need to evaluate the 2nd expression (on the right).

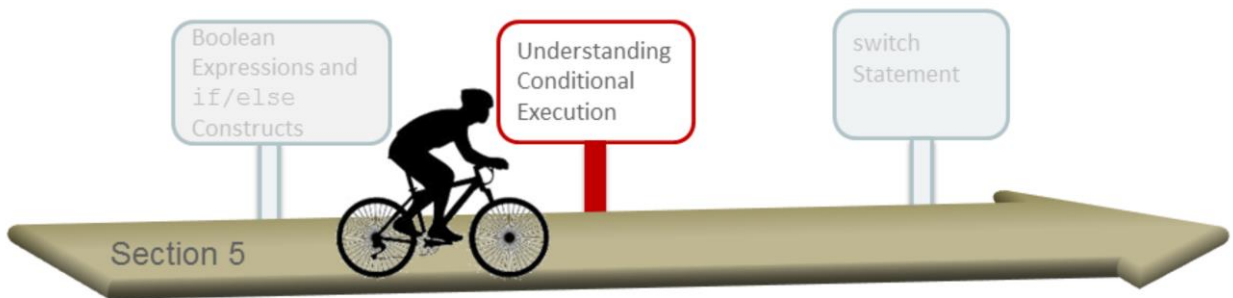- Consider this example:

```
boolean b = (x <= 10) || (x > 20);
```

Left Expression      Right Expression

- If (x<=10) is true, then (x>20) is not evaluated because it doesn't matter if (x>20) is true or false.

- The result of this expression is `true`.

# Topics

- Logical Operators

- Short-Circuit Evaluation

- **Ternary Operator**

- Build Chained `if` Constructs

Boolean Expressions and if/else Constructs

Understanding Conditional Execution

switch Statement

Section 5

# What Is a Ternary Conditional Operator?

| Operation | Operator | Example |
|---|---|---|
| If condition is true, assign result = value1. Otherwise, assign result = value2. **Note:** value1 and value2 must be the same data type. | ?: | result=condition ? value1 : value2<br><br>Example:<br>int x = 2, y = 5, z = 0;<br><br>z = (y < x) ? x : y; |

Equivalent statements

```
z = (y < x) ? x : y;
```

```
if(y<x){
    z=x;
}
else{
    z=y;
}
```

The ternary operator is a conditional operator that requires three operands. It has a more compact syntax than an `if/else` statement.

Use the ternary operator instead of an `if/else` statement if you want to make your code shorter. There are three operands in the slide example:

- `(y < x)`: This boolean expression (condition) is being evaluated.
- `? x`: If `(y < x)` is true, `z` will be assigned the value of `x`.
- `: y`: If `(y < x)` is false, `z` will be assigned the value of `y`.

In the slide example, `z = 5`.

## Ternary Conditional Operator: Scenario

Assume that you're playing a soccer game and you're tracking the goals as follows:

```java
public static void main(String args[]) {
        int numberOfGoals = 5;
        String s;
        if (numberOfGoals == 1) {
            s = "goal";
        }
        else {
            s = "goals";
        }
        System.out.println("I scored " + numberOfGoals + " " + s);
}
```

Based on the number of goals scored, these examples will print the appropriate singular or plural form of "goal." The operation is compact because it can yield only two results based on a boolean expression.

# Ternary Conditional Operator: Example

A similar result is achieved with the ternary operator by replacing the entire `if/else` statement with a single line.

```java
int numberOfGoals = 1
System.out.println("I scored " +numberOfGoals +" "
+(numberOfGoals==1 ? "goal" : "goals") );
```

As you can see, the ternary operator can be useful to reduce the number of lines of code, but it can make your code difficult to read and so it isn't ideal for nested statements.
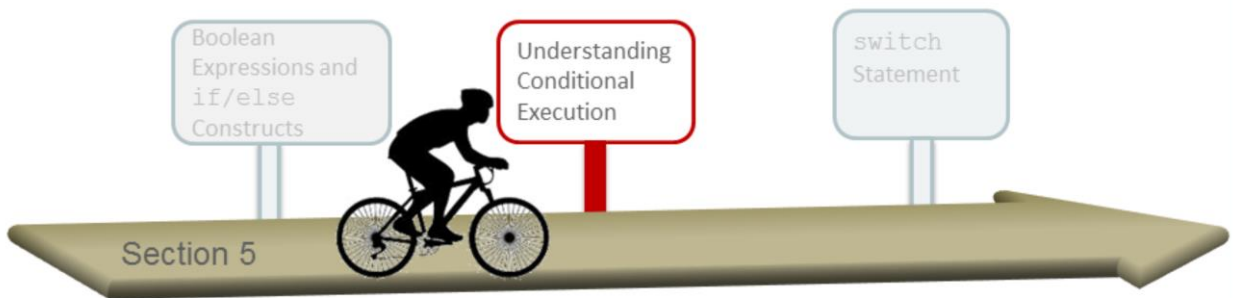
# Exercise 2

- Import and open the `ConditionalEx` project.
- Modify `TernaryOperator.java` to duplicate the logic given in the `if/else` statement by using the ternary operator.

# Topics

- Logical Operators

- Short-Circuit Evaluation

- Ternary Operator

- **Build Chained `if` Constructs**

Boolean Expressions and if/else Constructs

Understanding Conditional Execution

switch Statement

Section 5

# Handling Complex Conditions with a Chained `if` Construct

The chained `if` statement:

- Connects multiple conditions together into a single construct

- Tends to be confusing to read and hard to maintain

# Chaining `if`/`else` Constructs

- You can chain `if` and `else` constructs together to state multiple outcomes for several different expressions.

- Syntax:

```
if (<condition1>) {
    //code_block1
}
else if (<condition2>) {
    // code_block2
}
else {
    // default_code
}
```

The syntax for a chained if/else construct is shown in the slide example:

- Each of the conditions is a boolean expression.
- `code_block1` represents the lines of code that are executed if condition1 is true.
- `code_block2` represents the lines of code that are executed if condition1 is false and condition2 is true.
- `default_code` represents the lines of code that are executed if both conditions evaluate to false.

**Note:** Multiple else `if` statements can be evaluated. The else statement is optional.

**Java Foundations Course    S5L2- #**

# Chaining if/else Constructs: Example

```java
public static void main(String args[]) {
    double income = 30000, tax;
    if (income <= 15000) {
        tax = 0;
    }
    else if (income <= 25000) {
        tax = 0.05 * (income - 15000);
    }
    else {
        tax = 0.05 * (income - (25000 - 15000));
        tax += 0.10 * (income - 25000);
    }
}
```

This example demonstrates chaining if/else constructs to test multiple conditions. The else statement is executed if all conditions are false.

# Can if Statements Be Nested?

- In Java, an if statement can be present inside the body of another if statement.

```java
if (tvType == "color") {
    if (size == 14) {
        discPercent = 8;
    }
    else {
        disPercent = 10;
    }
}
```

- In this example, the else statement is paired with the if statement (size==14).

In a nested if statement:

It's very important to be sure which else construct goes with which if construct. This indentation greatly aids the clarity of the code for a reader.

In this example, if the outer if statement is true, then the inner if statement is executed.

# Understanding Nested `if` Statements

```java
if (tvType == "color") {
    if (size == 14) {
        discPercent = 8;
    }
}
else {
    discPercent = 10;
}
```

In this example, the `else` statement is paired with the outer `if` statement (`TVType=="color"`).

# Exercise 3

- Import and open the `ConditionalEx` project.
- Examine `ComputeFare.java`.
- Implement the following using `if/else` constructs:
  - Declare an integer variable, age.
  - Have the user enter the value for age.
- Using a chained if construct, compute the fare based on the age according to these conditions:
  - If age is less than 11, then fare=3$.
  - If age is greater than 11 and less than 65, then fare=5$.
  - Else for all other ages, then fare=3$.

# Summary

In this lesson, you should have learned how to:

- Describe conditional execution

- Describe logical operators

- Understand "short circuit" evaluation of logical operators

- Build chained `if` constructs