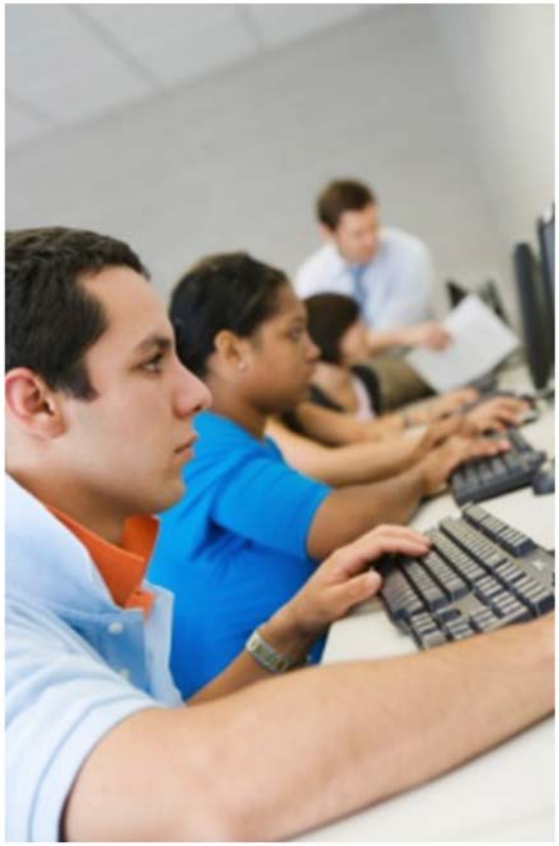

ORACLE
Academy

Oracle Academy
Java for AP Computer Science A

5-2

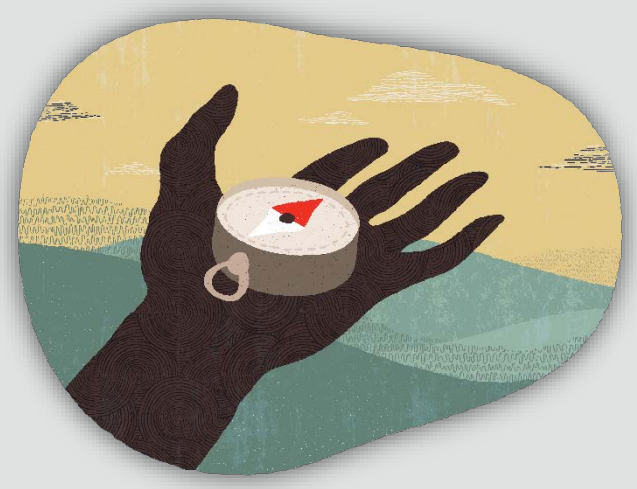
Understanding Conditional Execution



ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - Describe conditional execution
  - Describe logical operators
  - Understand “short circuit” evaluation of logical operators
  - Build chained if constructs



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.");
   }//endif
}//endif
```

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

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;
		int j = 8;
		((i < 1) && (j > 6))

If either one condition OR both conditions	**||**	int i = 2;
		int j = 8;
		((i < 1) || (j > 10))

NOT	**!**	int i = 2; (!(i < 3))

Applying Logical Operators

- You can write the previous example by using the logical AND operator as:

```
grade >= 88 && numberDaysAbsent == 0
```

Boolean Expression 1 — **Logical Operator** — **Boolean Expression 2**

- The logical operator allows you to test multiple conditions more easily, and the code is more readable

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.");
    }//endif
}//end method main
```

Evaluates to true if both boolean expressions are true

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**

Logical OR Operators: Example

```
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");
    }//endif
}//end method main
```

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

Logical NOT Operators

- Consider a scenario where a student is eligible for free tutoring if the following two conditions are met:
  - Grade < 88
  - Number of days absent < 3
- Use the ! logical operator

```
!madeFreeTutor && numberDaysAbsent < 3
```

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");
    }//endif
 }//end method main
```

Relational Operators and De Morgan

- Access the **Relational Operators and De Morgan** lesson from Section 5 of this course
  - Review the information on Page 1
  - Complete the Exercises on Pages 2 and 3

Exercise 1

- Create a new project and add the `WatchMovie.java` file to the 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**"

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

Skipping the Second AND Test

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

Left Expression

Right 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

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
		
		Example:
		int x = 2, y = 5, z = 0;
		
		z = (y < x) ? x : y;

Equivalent statements

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

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

Ternary Conditional Operator: Scenario

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

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

Ternary Conditional Operator: Example

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

```
int numberOfGoals = 1;

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

Ternary Conditional Operator: Example

- Advantage: Place the operation directly within an expression

```
int numberOfGoals = 1;

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

- Disadvantage: Can have only two potential results

```
(numberOfGoals==1 ? "goal" : "goals" : "More goals");
```

boolean — true — false — ???

Exercise 2

- Add the file `TernaryOperator.java` to the project you created for exercise 1
- Modify `TernaryOperator.java` to duplicate the logic given in the if/else statement by using the ternary operator

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
}//endif
```

Chaining if/else Constructs: Example

```
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);
   }//endif
}//end method main
```

Can if Statements Be Nested?

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

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

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

Understanding Nested if Statements

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

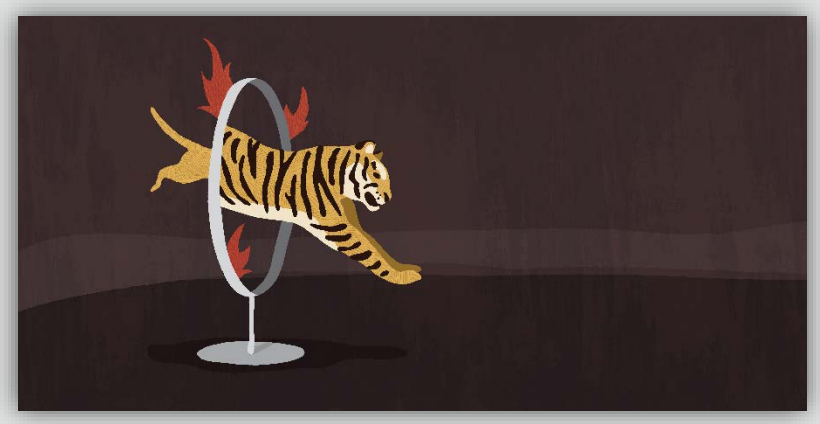
```
 if (tvType == "color") {
    if (size == 14) {
        discPercent = 8;
     }//endif
 }
 else {
    discPercent = 10;
}//endif
```

Exercise 3

- Add the file `ComputeFare.java` to the project you created for exercise 1
- 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




ORACLE
Academy