

Java Foundations

boolean Expressions and if/else Constructs



Objectives

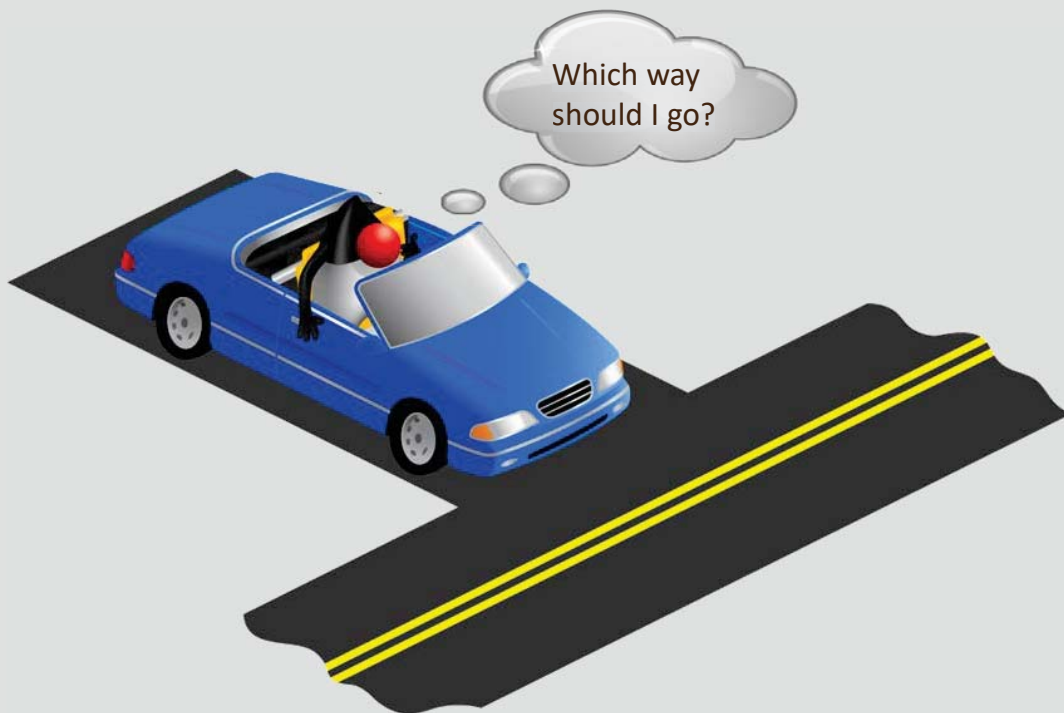
- This lesson covers the following objectives:
 - Declare, initialize, and use boolean variables
 - Compare boolean expressions using relational operators
 - Create an if statement
 - Create if/else constructs
 - Compare Strings



Making Decisions

- So far in the previous lessons, you saw different data types supported in Java
- boolean is another data type in Java that helps to add logic to a program
- It helps to make decisions

Making Decisions



Making Decisions

- Let's say that you're driving to the school
- You stop at an intersection
- And now you have to make a logical decision:
 - If I turn left, will it take me to the school?
 - If I go straight, will it take me to the school?
 - If I turn right, will it take me to the school?
- There are only two answers to each of these questions:
 - yes or no

Java's boolean Data Type

- It's basically the same in Java, where booleans will tell the program which is the best course of action to take
- In Java the values for the boolean data type are true and false, instead of yes and no
- You declare the boolean data type by using the boolean keyword

Using Java's boolean Data Type: Example

- Note: The value of a boolean variable is displayed as true or false

```
public static void main(String args[]) {  
    boolean passed, largeVenue, grade;   
    passed = true;  
    largeVenue = false;  
    grade = passed;  
    System.out.println(passed);  
    System.out.println(largeVenue);  
    System.out.println(grade);  
} //end method main
```

Declaring boolean variables

Assigning values to boolean variables

Printing values of boolean variables

boolean Data Type: Scenario

- What if you were driving a car that has an installed GPS system running on Java?
- Before you leave home, you ask the GPS system to take you to the school
- What simple code would you write to help you decide which way to turn?

boolean Data Type: Scenario

- Let's start

```
public static void main(String args[]) {  
    String left = "museum";  
    String straight = "gym";  
    String right = "restaurant";  
    boolean isLeft = false;  
    boolean isStraight = true;  
    boolean isRight = false;  
    System.out.println("Go straight ahead");  
} //end method main
```

Expressions and Variables

- Mathematical expressions can be ...
 - Printed
 - Assigned to an int or double variable

```
System.out.println(2 + 2);  
int x = 2 + 2;
```

Expressions and Variables

- boolean expressions can be ...
 - Printed
 - Assigned to a boolean variable

```
System.out.println(x == 5);  
boolean isFive = x == 5;
```

Equality and Assignment

- == is a relational operator
- This operator tests to see if both sides of a boolean expression equal each other
- A boolean expression returns a value of true or false

```
x == 5
```

Equality and Assignment

- = is an assignment operator
- This operator assigns a value to a variable
- A boolean variable can be assigned whichever value a boolean expression returns

```
int x = 4;  
boolean isFive = x == 5;
```

Values in boolean Expressions

- Use == to test equality between primitive values
- boolean expressions may contain variables or hard-coded values

```
boolean res1 = 24 == 15;  
System.out.println("res1: " + res1);  
int value1 = 15;  
int value2 = 24;  
boolean res2 = value1 == value2;  
System.out.println("res2: " + res2);
```

Values in boolean Expressions

- Both expressions below return the same value:
 - If value1 and value2 hold the same value, the expression returns a true result
 - Otherwise, the expression returns false

```
boolean res1 = 24 == 15;  
System.out.println("res1: " + res1);  
int value1 = 15;  
int value2 = 24;  
boolean res2 = value1 == value2;  
System.out.println("res2: " + res2);
```

Relational Operators

- Use relational operators in boolean expressions that are used to evaluate if/else statements

Relational Operators

Condition	Operator	Example
Is equal to	==	<code>int i=1;</code> <code>(i == 1)</code>
Is not equal to	!=	<code>int i=2;</code> <code>(i != 1)</code>
Is less than	<	<code>int i=0;</code> <code>(i < 1)</code>
Is less than or equal to	<=	<code>int i=1;</code> <code>(i <= 1)</code>
Is greater than	>	<code>int i=2;</code> <code>(i > 1)</code>
Is greater than or equal to	>=	<code>int i=1;</code> <code>(i >= 1)</code>

Relational Operators: Example

- Note: Use the equal sign (=) to make an assignment and use the == sign to make a comparison and return a boolean

```
public static void main(String args[]) {  
    int a = 10;  
    int b = 20;  
    System.out.println(a == b);  
    System.out.println(a != b);  
    System.out.println(a > b);  
    System.out.println(a < b);  
    System.out.println(b >= a);  
    System.out.println(b <= a);  
}//end method main
```

For primitive
values == checks
for equality
testing



Exercise 1

- Import and open the `IfElseEx` project
- Modify `AgeValidity.java` to implement the following:
 - Have users enter their age
 - Declare a boolean variable, `drivingUnderAge`
 - Initialize `drivingUnderAge` to false
 - Write a boolean expression to check if the age entered by the user is less than or equal to 18, and then set `drivingUnderAge` to true
 - Print the value of `drivingUnderAge`

Conditional Statements

- Conditional statements let us choose which statement are executed next
- These decisions are based on boolean expressions (or conditions) that evaluate to true or false
- Conditional statements in Java are:
 - if statement
 - if/else statement
 - switch statement

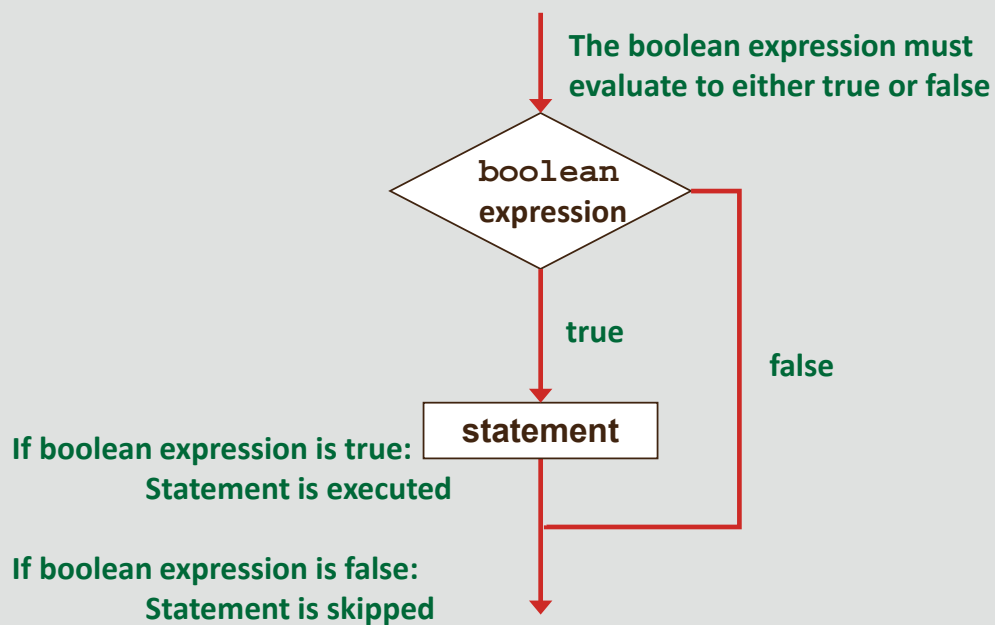
Understanding the if Statement

- An if statement consists of a boolean expression followed by one or more statements
- Syntax:

boolean expression

```
if ( <some condition is true> ){  
    //Statements will execute if the boolean  
    //expression is true  
} //endif
```

Understanding the if Statement



Using boolean Expressions in if Statements

```
public static void main(String args[]) {  
    String left = "museum";  
    String straight = "gym";  
    String right = "restaurant";  
  
    if (left == "gym") {  
        System.out.println("Turn Left");  
    }//endif  
  
    if (straight == "gym") {  
        System.out.println("Drive Straight");  
    }//endif  
  
    if (right == "gym") {  
        System.out.println("Turn Right");  
    }//endif  
}//end method main
```

This block is executed

Executing a Block of Code

1. A code block isn't needed for one statement to be executed by an if statement.
– Here's an example:

```
daysInFeb = 28;  
if(isLeapYear)  
    daysInFeb = 29;  
    System.out.println(year + " is a leap year");
```

Only this statement is carried out

Executing a Block of Code

2. However, it's always recommended that you use code blocks, even if there's only one statement to execute in the block

```
daysInFeb = 28;
if(isLeapYear){
    daysInFeb = 29;
    System.out.println(year + " is a leap year");
} //endif
```

This block will be executed

2

if Statement: Examples

```
public static void main(String args[]) {
    int grade = 85;

    if (grade > 88) {
        System.out.println("You made the Honor Roll.");
    } //endif

    if (grade <=88) {
        System.out.println("You are eligible for tutoring.");
    }//endif

} //end method main
```

Second if
statement

- Output:

You are eligible for tutoring.



Exercise 2

- Import and open the IfElseEx project
- Modify the ChkOddEven.java to implement the following:
 - Input a number between 1 and 10
 - Use if statements
 - Test whether a number is odd or even
- The program should generate the following output:
 - Enter a number: 7
 - The num is odd 7

Choosing Between Two Alternatives

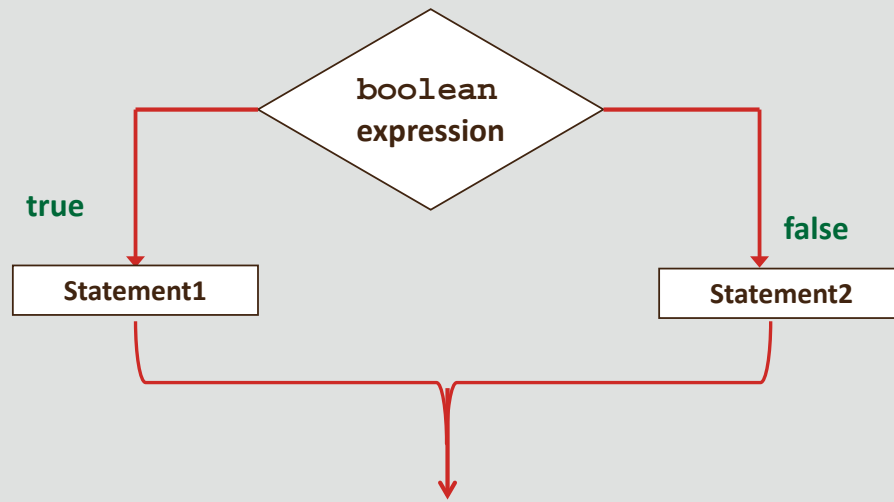
- If you want to choose between two alternatives you use the if/else statement
- Syntax:

```
boolean expression
if ( <some condition is true> ) {
    // do something
}
else {
    // do something different
} //endif
```

if block

else block

Understanding if/else Statements



If boolean expression is true: Statement1 is executed
If boolean expression is false: Statement2 is skipped

if/else Statements: Example 1

```
String forecast;  
double temperature = getTemperature();  
  
if (temperature <= 32.0) {  
    forecast = "SNOW";  
}  
else {  
    forecast = "RAIN";  
} //endif
```

This block is
executed



30.3 °F

if/else Statements: Example 2

```
String forecast;  
double temperature = getTemperature();  
  
if (temperature <= 32.0) {  
    forecast = "SNOW";  
}  
else {  
    forecast = "RAIN";  
} //endif
```

This block is
executed



40,2 °F

if/else Statements: Example 3

- You can replace the two if statements with an if/else statement
- The if/else statement is more efficient because only one comparison is being made

```
public static void main(String args[]) {  
    int grade = 85;  
    if (grade > 88) {  
        System.out.println("You made the Honor Roll.");  
    }  
    else {  
        System.out.println("You passed.");  
    } //endif  
} //end method main
```




Exercise 3

- Import and open the `IfElseEx` project
- Examine `AgeCheck.java`:
 - The program has a logic problem
 - For some values, it prints the wrong answer
 - Find the problems and fix them. (You may need to run the program a few times and try different values to see which ones fail)
 - Replace the two if statements with an if/else statement



Exercise 4

- Import and open the `IfElseEx` project
- Examine `ShoppingCart.java`
- Use an if/else statement to implement the following:
 - Declare and initialize a boolean variable, `outOfStock`
 - If `quantity > 1`, change the message variable to indicate plural
 - If an item is out of stock, inform the user that the item is unavailable, else print the message and the total cost

Comparing Variables

- When you compare values by using boolean expressions, you need to understand the nuances of certain data types
- Relational operators such as `==` are ...
 - Great for comparing primitives
 - Terrible for comparing Strings (and other objects)
- Let's examine why

Comparing Primitives

- The value `z` is set to be the sum of `x + y`
- When a boolean expression tests the equality between `z` and the sum of `x + y`, the result is `true`

```
int x = 3;  
int y = 2;  
int z = x + y;  
  
boolean test = (z == x + y);  
System.out.println(test);           //true
```

Comparing Strings

- The value z is set to be the concatenation of x + y
- When a boolean expression tests the equality between z and the concatenation of x + y, the result is false

```
String x = "Ora";  
String y = "cle";  
String z = x + y;  
  
boolean test = (z == x + y);  
System.out.println(test);           //false
```

Why?



Why Are There Contradictory Results?

- Primitives and objects are stored differently in memory
 - Strings are given special treatment
 - This is discussed later in the course
- As a result ...
 - == compares the values of primitives
 - == compares the objects' locations in memory
- It's much more likely that you'll need to compare the content of Strings and not their locations in memory

How Should You Compare Strings?

- You should almost never compare Strings using ==
- Instead, compare Strings using the equals() method
 - This method is part of the String class
 - It accepts one String argument, checks whether the contents of Strings are equal, and then returns a boolean
 - There is also a similar method, equalsIgnoreCase()

```
String x = "Ora";  
String y = "cle";  
String z = x + y;  
boolean test = z.equals(x + y);  
System.out.println(test);           //true
```

Exercise 5



- Import and open the IfElseEx project
- Examine StringEquality.java
- Use an if and an if/else statement:
 - Declare a String variable name
 - Have the user input a value for the name
 - Check whether the name is “Moe,” and then print “You are the king of rock and roll”
 - Otherwise print “You are not the king”
 - Don’t use ==

Summary

- In this lesson, you should have learned how to:
 - Declare, initialize, and use boolean variables
 - Compare primitive values using relational operators
 - Create an if statement
 - Create if/else constructs
 - Compare Strings

