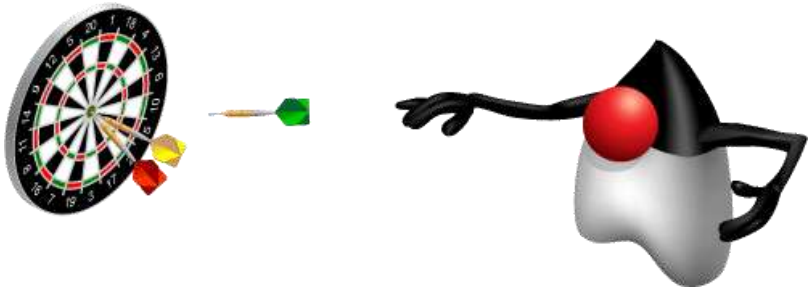
More on Conditionals

Objectives

- After completing this lesson, you should be able to:
  - Correctly use all of the conditional operators
  - Test equality between string values
  - Chain an `if/else` statement to achieve the desired result
  - Use a `switch` statement to achieve the desired result
  - Debug your Java code by using the NetBeans debugger to step through code line by line and view variable values



Topics

- Relational and conditional operators
- More ways to use `if/else` statements
- Using a `switch` statement
- Using the NetBeans debugger

Review: Relational Operators

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

Testing Equality Between `String` variables

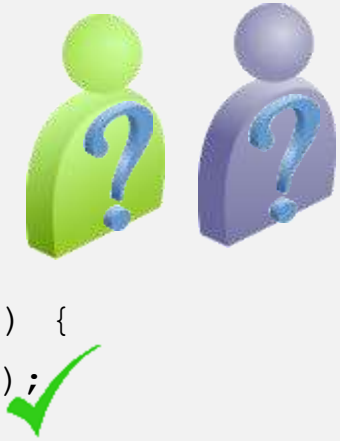
- Example:

```
•     public class Employees {

•       public String name1 = "Fred Smith";
•       public String name2 = "Sam Smith";
•
•       public void areNamesEqual() {
•           if (name1.equals(name2)) {
•             System.out.println("Same name.");
•         }
•           else {
•           System.out.println("Different name.");
•           }
•         }
•     }
```

Testing Equality Between `String` variables

- Example:

```
•     public class Employees {

•       public String name1 = "Fred Smith";
•       public String name2 = "fred smith";
•
•       public void areNamesEqual() {
•           if (name1.equalsIgnoreCase(name2)) {
•             System.out.println("Same name.");
•         }
•           else {
•           System.out.println("Different name.");
•           }
•         }
•     }
```



Testing Equality Between `String` variables

- Example:

```
•     public class Employees {

•       public String name1 = "Fred Smith";
•       public String name2 = "Fred Smith";
•
•       public void areNamesEqual() {
•           if (name1 == name2) {
•             System.out.println("Same name.");
•         }
•           else {
•           System.out.println("Different name.");
•           }
•         }
•     }
```

Testing Equality Between `String` variables

Example:

```
public class Employees {

  public String name1 = new String("Fred Smith");
  public String name2 = new String("Fred Smith");

  public void areNamesEqual() {
      if (name1 == name2) {
        System.out.println("Same name.");
      }
      else {
        System.out.println("Different name.");
      }
    }
  }
```

Common Conditional 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 another condition	`||`	`int i = 2;`
		`int j = 8;`
		`((i < 1) || (j > 10))`

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

Ternary Conditional Operator

Operation	Operator	Example

If `some condition` is true, assign the value of `value1` to the result. Otherwise, assign the value of `value2` to the 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;
}
```

Using the Ternary Operator

- Advantage: Usable in a single line

```
int numberOfGoals = 1;
String s = (numberOfGoals==1 ? "goal" : "goals");

System.out.println("I scored " +numberOfGoals +" "
                                        +s );
```

- 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 10-1: Using the Ternary Operator

- In this exercise, you use a ternary operator to duplicate the same logic shown in this `if/else` statement:

```
01      int x = 4, y = 9;
02      if ((y / x) < 3){
03          x += y;
04      }
05      else x *= y;
```

Topics

- Relational and conditional operators
- More ways to use `if/else` statements
- Using a `switch` statement
- Using the NetBeans debugger

Java Puzzle Ball

- Have you played through **Basic Puzzle 12**?

Consider the following:

What happens `if` the ball strikes the blade?

Java Puzzle Ball Debrief

- What happens `if` the ball strikes the blade?
  - `if` the ball strikes the blade:
    - Transform the ball into a blade
  - `if` the ball is a blade `&&` it strikes the fan:
    - The ball is blown in the direction of the fan
  - `if` the ball is a blade `&&` it strikes any object other than the fan `||` blade:
    - Destroy that object
    - Transform the ball back into a ball

Handling Complex Conditions with a Chained `if` Construct

- The chained `if` statement:
  - Connects multiple conditions together into a single construct
  - Often contains nested `if` statements
  - Tends to be confusing to read and hard to maintain

Determining the Number of Days in a Month

```
• 01  if (month == 1 || month == 3 || month == 5 || month == 7
• 02       || month == 8 || month == 10 || month == 12) {
• 03     System.out.println("31 days in the month.");
• 04  }
• 05  else if (month == 2) {
• 06     if(!isLeapYear){
• 07        System.out.println("28 days in the month.");
• 08    }else System.out.println("29 days in the month.");
• 09  }
• 10  else if (month ==4 || month == 6 || month == 9
• 11         || month == 11) {
• 12     System.out.println("30 days in the month.");
• 13  }
• 14  else
• 15     System.out.println("Invalid month.");
```

Chaining `if/else` Constructs

Syntax:

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

Exercise 10-2: Chaining `if` Statements

- In this exercise, you write a `calcDiscount` method that determines the discount for three different customer types:
  - Nonprofits get a discount of 10% if total > 900, else 8%.
  - Private customers get a discount of 7% if total > 900, else no discount.
  - Corporations get a discount of 8% if total > 500, else 5%.

Topics

- Relational and conditional operators
- More ways to use `if/else` statements
- Using a `switch` statement
- Using the NetBeans debugger

Handling Complex Conditions with a `switch` Statement

- The `switch` statement:
  - Is a streamlined version of chained `if` statements
  - Is easier to read and maintain
  - Offers better performance

Coding Complex Conditions: `switch`

```
01 switch (month) {
02      case 1: case 3: case 5: case 7:
03      case 8: case 10: case 12:
04          System.out.println("31 days in the month.");
05          break;
06      case 2:
07          if (!isLeapYear) {
08             System.out.println("28 days in the month.");
09          } else
10             System.out.println("29 days in the month.");
11          break;
12      case 4: case 6: case 9: case 11:
14          System.out.println("30 days in the month.");
15          break;
16      default:
17          System.out.println("Invalid month.");
• 18 }
```

`switch` Statement Syntax

Syntax:

```
switch (<variable or expression>) {
   case <literal value>:
      //code_block1
      [break;]
    case <literal value>:
      // code_block2
      [break;]
     default:
       //default_code
{
```

When to Use `switch` Constructs

- Use when you are testing:
  - Equality (not a range)
  - A *single* value
  - Against fixed known values at compile time
  - The following data types:
    - Primitive data types: `int`, `short`, `byte`, `char`
    - `String` or `enum` (enumerated types)
    - Wrapper classes (special classes that wrap certain primitive types): `Integer`, `Short`, `Byte` and `Character`

Only a single (value can be tested.

```
switch (month) {
    case 1: case 3: case 5: case 7:
    case 8: case 10: case 12:
        System.out.println("31 days in the month.");
        break;
    case 2:
        if (!isLeapYear) {
```

Known values

Exercise 10-3: Using `switch` Construct

- In this exercise, you modify the `calcDiscount` method to use a `switch` construct, instead of a chained `if` construct:
  - Use a ternary operator instead of a nested `if` within each case block.

Quiz

- Which of the following sentences describe a valid case to test in a `switch` construct?

a. The `switch` construct tests whether values are greater than or less than a single value.
b. Variable or expression where the expression returns a supported `switch` type.
c. The `switch` construct can test the value of a `float`, `double`, `boolean`, or `String`.
d. The `switch` construct tests the outcome of a `boolean` expression.

Summary

- In this lesson, you should have learned how to:
  - Use a `ternary` statement
  - Test equality between strings
  - Chain an `if/else` statement
  - Use a `switch` statement
  - Use the NetBeans debugger