10

More on Conditionals

Objectives

After completing this lesson, you should be able to:

- Use a `ternary` statement
- Test equality between strings
- Chain an `if/else` statement
- Use a `switch` statement
- Use the NetBeans debugger

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)`

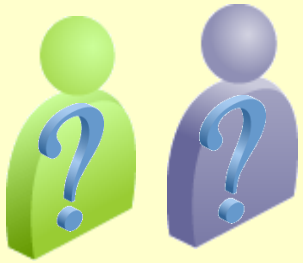
Testing Equality Between `String` variables

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.");
        }
    }
}
```



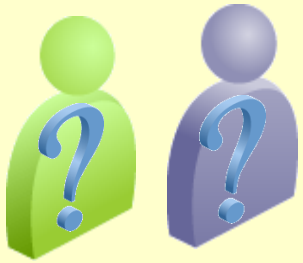
Testing Equality Between `String` variables

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.");
        }
    }
}
```



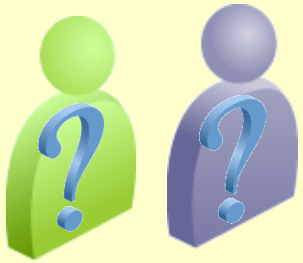
Testing Equality Between `String` variables

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.");
        }
    }
}
```



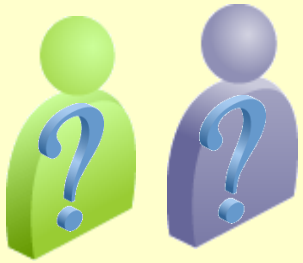
Testing Equality Between `String` variables

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

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

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

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

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 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

```
if (month == 1 || month == 3 || month == 5 || month == 7
     || month == 8 || month == 10 || month == 12) {
    System.out.println("31 days in the month.");
}
else if (month == 2) {
    if(!isLeapYear){
       System.out.println("28 days in the month.");
   }else System.out.println("29 days in the month.");
}
else if (month ==4 || month == 6 || month == 9
        || month == 11) {
    System.out.println("30 days in the month.");
}
else
    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

1. Open the project `Exercise_10-2` in NetBeans.

In the `Order` class:

2. Complete the `calcDiscount` method so it determines the discount for three different customer types:
   - Nonprofits get a discount of 10% if total > 900, else 5%.
   - Private customers get a discount of 7% if total > 900, else 0%.
   - Corporations get a discount of 8% if total < 500, else 5%.

In the `ShoppingCart` class:

3. Use the `main` method to test the `calcDiscount` method.

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`

```
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) {
             System.out.println("28 days in the month.");
          } else
             System.out.println("29 days in the month.");
          break;
      case 4: case 6: case 9: case 11:
          System.out.println("30 days in the month.");
          break;
      default:
          System.out.println("Invalid month.");
}
```

`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 variable 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) {
            System.out.println("28 days in the month.");
        } else
            System.out.println("29 days in the month.");
```

Known values

Exercise 10-3: Using `switch` Construct

1. Continue editing **`Exercise_10-2`** or open **`Exercise_10-3`**.

In the `Order` class:

2. Rewrite `calcDiscount` to use a `switch` statement:
   - Use a ternary expression to replace the nested if logic.
   - For better performance, use a `break` statement in each case block.
   - Include a `default` block to handle invalid `custType` values.

In the `ShoppingCart` class:

3. Use the main method to test the `calcDiscount` method.

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.

Topics

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

Working with an IDE Debugger

Most IDEs provide a debugger. They are helpful to solve:

- Logic problems
  - (Why am I not getting the result I expect?)
- Runtime errors
  - (Why is there a `NullPointerException`?)

Debugger Basics

- Breakpoints:
  - Are stopping points that you set on a line of code
  - Stop execution at that line so you can view the state of the application
- Stepping through code:
  - After stopping at a break point, you can “walk” through your code, line by line to see how things change.
- Variables:
  - You can view or change the value of a variable at run time.
- Output:
  - You can view the System output at any time.

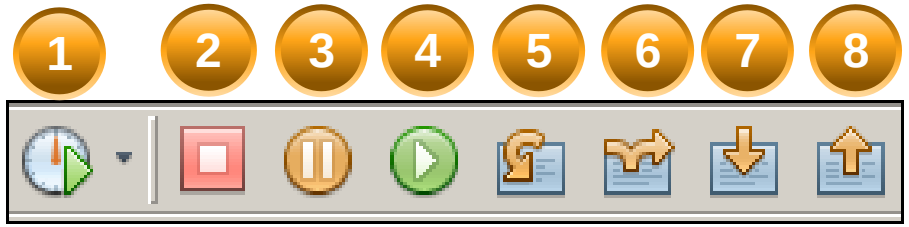
Setting Breakpoints

- To set breakpoints, click in the margin of a line of code.
- You can set multiple breakpoints in multiple classes.

```
3   public class DebugTestIfElse {
4       public static void main(String[] args) {
5           int month =11;
6           boolean isLeapYear = true;
7
8           if(month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12){
9               System.out.println("31 days in the month.");
10          }
11          else if(month == 2){
12              if(!isLeapYear){
13                  System.out.println("28 days in the month.");
14              }
15              else{
16                  System.out.println("29 days in the month.");
17              }
18          }
19          else if(month == 4 || month ==6 || month == 9 || month ==11){
20              System.out.println("30 days in the month.");
21          }
22          else{
23              System.out.println("Invalid month");
```

The Debug Toolbar

1. Start debugger
2. Stop debug session
3. Pause debug session
4. Continue running
5. Step over
6. Step over an expression
7. Step into
8. Step out of



Viewing Variables

Breakpoint

Current line of execution

```
8   switch(month){
9       case 1: case 3: case 5: case 7:
10      case 8: case 10: case 12:
11          System.out.println("31 days in the month.");
12          break;
13      case 2:
14          if(!isLeapYear){
15              System.out.println("28 days in the month.");
16          }
17          else{
18              System.out.println("29 days in the month.");
19          }
20          break;
21      case 4: case 6: case 9: case 11:
```

DebutTestSwitch > main > switch (month) > case 2:

Output | Watches | Variables | Breakpoints

Name	Type	Value
Static		
args	String[]	#72(length=0)
month	int	2
isLeapYear	boolean	true

Value of variables

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

Practices Overview

- 10-1: Using Conditionals
- 10-2: Debugging