Université Concordia University
CENTRE FOR CONTINUING EDUCATION

INTRODUCTION TO PROGRAMMING WITH JAVA - CEJV416

Lecture 5

Boolean and Relational Operators

Control Structures

if, if/else, switch

Control

- In software the term control is about choosing which instruction to carry out based on a Boolean operation
- Boolean operations are expressions that have only two possible outcomes, true or false
- We express Boolean operations using the relational operators

Relational Operators

Operator	Name	Returns a true value if…
==	Equality	Both operands are equal.
!=	Inequality	The left and right operands are not equal.
>	Greater Than	The left operand is greater than the right operand.
<	Less Than	The left operand is less than the right operand.
>=	Greater Than Or Equal	The left operand is greater than or equal to the right operand.
<=	Less Than Or Equal	The left operand is less than or equal to the right operand.

How to compare primitive data types

- Use the *relational operators* to create a *Boolean expression* that compares two operands and returns a boolean value that is either true or false.
- When comparing two numeric operands that are not of the same type, Java will convert the less precise operand to the type of the more precise operand before doing the comparison.
- By definition, a boolean variable evaluates to a boolean value of true or false.

Examples of Boolean expressions


```
int a = 1;
System.out.println(a + 2 <= 10); //true
System.out.println(a != 0);      //true

char c = 'y';
System.out.println(c == 'z');    //false

double x = 2.3;
System.out.println(x >= 1);      //true


String r = null;
System.out.println(r != null);   //false
```

Control Structures

- All programs can be written in terms of three control structures
  1. Sequence Structure
  2. Selection Structure
  3. Repetition Structure

The Selection Structure

- A selection structure allows a program to choose amongst alternative courses of action
- The first selection structure is `if`

if


```
if (grade >= 60) {
   System.out.println("Passed");
}
```

- Selection structures require logical expressions
- The logical expression must be inside of parenthesis

if/else


```
if (value > 10) {
   System.out.println("Too Large");
} else {
   System.out.println("Too Small");
}
```

Example


```
if ( myScore > yourScore ) {
    System.out.println ("I won");
} else {
    System.out.println ("I will win next time!");
}
```

```
if  (sales > minimum) {
     salary = salary + bonus;
}
```

```
if  (sales > minimum)
     salary = salary + bonus;
```

The syntax of the if/else statement


```
if (booleanExpression) {statements}
[else if (booleanExpression) {statements}] ...
[else {statements}]
```

Example


```
if  (numberOfPeople < 50)
    System.out.println("Less than 50 people");
else if  (numberOfPeople < 100) {
    System.out.print("At least 50 and ");
    System.out.println("less than 100 people");
}
else if  (numberOfPeople < 200) {
    System.out.print("At least 100 and ");
    System.out.println("less than 200 people");
}
else
    System.out.println("At least 200 people");
```

Exercise 10 - Hi Five or HiEven

- Write a program that prompts the user to enter an integer. If the number is a multiple of 5, print HiFive. If the number is divisible by 2, print HiEven.

```
run:
Enter an integer:
15
HiFive
BUILD SUCCESSFUL (total time: 5 seconds)
```

```
run:
Enter an integer:
8
HiEven
BUILD SUCCESSFUL (total time: 2 seconds)
```

```
run:
Enter an integer:
10
HiFive
HiEven
BUILD SUCCESSFUL (total time: 2 seconds)
```

An if statement that contains two blocks of code


```
double discountPercent = 0.0;
if (customerType.equals("R"))
{                                       // start block
    discountPercent = .1;
    shippingMethod = "UPS";
}                                       // end block
else if (customerType.equals("C")
{                                       // start block
    discountPercent = .2;
    shippingMethod = "Bulk";
}                                       // end block
else
    shippingMethod = "USPS";
```

Nested if statements


```
if (customerType.equals("R"))
{                                       // begin nested if
    if (subtotal >= 100)
        discountPercent = .2;
    else
        discountPercent =.1;
}                                       // end nested if
else
    discountPercent = .4;
```

Logical operators

Operator	Name	Description
`&&`	And	Returns a true value if both expressions are true. This operator only evaluates the second expression if necessary.
`||`	Or	Returns a true value if either expression is true. This operator only evaluates the second expression if necessary.
`!`	Not	Reverses the value of the expression.

How to use the logical operators


```
subtotal >= 250 && subtotal < 500
timeInService <=4 || timeInService >= 12

(subtotal >= 250 && subtotal < 500) ||
isValid == true

!(counter++ >= years)
```

- You can use the *logical operators* to create a Boolean expression that combines two or more Boolean expressions.
- By default, Not operations are performed first, followed by And operations, and then Or operations.
- Since the && and || operators evaluate the second expression only if necessary, they can be referred to as *short-circuit operators*.

Truth Table for Operator !

p	!p
true	false
false	true

Example (assume age = 24, gender = 'M')
!(age > 18) is false, because (age > 18) is true.
!(gender != 'M') is true, because (grade != 'M') is false.

Truth Table for Operator &&

p1	p2	p1 && p2
false	false	false
false	true	false
true	false	false
true	true	true

Example (assume age = 24, gender = 'F')

(age > 18) && (gender == 'F') is true, because (age > 18) and (gender == 'F') are both true.

(age > 18) && (gender != 'F') is false, because (gender != 'F') is false.

Truth Table for Operator ||

p1	p2	p1 || p2
false	false	false
false	true	true
true	false	true
true	true	true

Example (assume age = 24, gender = 'F')

(age > 34) || (gender == 'F') is true, because (gender == 'F') is true.

(age > 34) || (gender == 'M') is false, because (age > 34) and (gender == 'M') are both false.

Truth Tables


```
answer = (6 > 4) && (7 < 9);
```

The result would be true

```
answer = (6 > 9) || (7 < 9);
```

The result would be true

When combining with && every condition must be true

When combining with || only one of the conditions must be true

Short Circuit Evaluation

Java uses short circuit evaluation

```
answer = (7 > 9) && (3 < 4);
```

Once it is determined that 7 > 9 is false the second logical expression is skipped

```
answer = (5 > 3)  || (3 > 4);
```

Once it is determined that 5 > 3 is true the second logical expression is skipped

Example

- Write a java statement that check the followings:
  - If the number is between 0 and 100, and its dividable by 3

- if ((number > 0) && (number <100) && (number%3==0))

Exercise 11 - A Simple Math Learning Tool

- Witte a program to let a first grader practice additions. The program randomly generates two single-digit integers number1 and number2 and displays a question such as "What is 7 + 9?" to the student. After the student types the answer, the program displays a message to indicate whether the answer is true or false.

```
run:
What is 2 + 7?
9
2 + 7 = 9 is true
BUILD SUCCESSFUL (total time: 8 seconds)
```

```
run:
What is 3 + 0?
5
3 + 0 = 5 is false
BUILD SUCCESSFUL (total time: 4 seconds)
```

Special Case - String

- Strings are not primitives, they are objects
- Relational operators cannot be used on objects
- If an object is intended for a Boolean expression it must have a method to carry out the operation

Method	Description
`equals(String)`	Compares the current String object with the String object specified as the argument and returns a true value if they are equal. This method makes a case-sensitive comparison.
`equalsIgnoreCase(String)`	Works like the equals method but is not case-sensitive.

How to compare strings

- To test whether two strings contain the same string value, you must call one of the methods of the String object.
- To test whether a string is null, you can use the equality operator (==) or the inequality operator (!=) with the null keyword.

How to compare strings

- A string object is a *reference type*, not a primitive data type. So instead of containing data, a string variable refers to (or points to) the data, which is in another location of computer memory.
- If you use the equality or inequality operator to compare two string variables, Java tests to see whether the two strings refer to the same String object. If they do, the expression is true.

Expressions that compare two string values


```
firstName.equals("Frank")          // equal to a string literal
firstName.equalsIgnoreCase("Frank")
                                   // equal to a string literal
firstName.equals("")               // equal to an empty string

!lastName.equals("Jones")          // not equal to a string
literal
!code.equalsIgnoreCase(productCode)
                        // not equal to another string variable

firstName == null                  // equal to a null value
firstName != null                  // not equal to a null value
```

Code that tests whether two strings refer to the same object


```
Scanner sc = new Scanner(System.in);
System.out.print("Enter string1: ");
String string1 = sc.next();
System.out.print("Enter string2: ");
String string2 = sc.next();

if (string1 == string2)
            // this will be false no matter what you enter
    System.out.println("string1 = string2");
else
    System.out.println("string1 not = string2");
```

Example


```
String s = new String("test");
String t = new String("test");
System.out.println(s == t);      //false
System.out.println(s.equals(t)); //true
```

The syntax of the switch statement


```
switch (switchExpression)
{
    case label1:
        statements
        break;
    [case label2:
        statements
        break;] ...
    [default:
        statements
        break;]
}
```

A switch statement with a default label


```
switch (productID)
{
    case 1:
        productDescription = "Hammer";
        break;
    case 2:
        productDescription = "Box of Nails";
        break;
    default:
        productDescription = "Product not found";
        break;
}
```

How to code switch statements

- Can only be used with an expression that evaluates to one of these integer types: char, byte, short, or int and, since Java 1.7, Strings
- Case labels represent the literal integer values of the expression or a String, and these labels can be coded in any sequence.
- The switch statement transfers control to the appropriate case label.

How to code switch statements

- If control isn't transferred to one of the case labels, the optional default label is executed.
- If a case label doesn't contain a break statement, code execution will fall through to the next label. Otherwise, the break statement ends the switch statement.

A switch statement that uses a string


```
switch (productCode)
{
    case "hm01":
        productDescription = "Hammer";
        break;
    case "bn03":
        productDescription = "Box of Nails";
        break;
    default:
        productDescription = "Product not found";
        break;
}
```

A switch statement that falls through case labels


```
switch (dayOfWeek)
{
    case 2:
    case 3:
    case 4:
    case 5:
    case 6:
        day = "weekday";
        break;
    case 1:
    case 7:
        day = "weekend";
        break;
}
```

Exercise 12

- Ask a user enter a province code and print the province name. Print "Not found" for any invalid province code. Do it once with if-else and once with switch

- AB Alberta
- BC British Columbia
- MB Manitoba
- NB New Brunswick
- NL Newfoundland
- NS Nova Scotia
- NT Northwest Territories
- NU Nunavut
- ON Ontario
- PE Prince Edward Island
- QC Quebec
- SK Saskatchewan
- YT Yukon

Exercise 13-1
An Improved Math Learning Tool

- Write a program to teach a first grade child how to learn subtractions. The program randomly generates two single-digit integers number1 and number2 with number1 >= number2 and displays a question such as "What is 9 – 2?" to the student. After the student types the answer, the program displays whether the answer is correct.

Exercise 13-2
Body Mass Index

- Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters. The interpretation of BMI for people 16 years or older is as follows:

BMI	Interpretation
Below 18.5	Underweight
18.5-24.9	Normal
25.0-29.9	Overweight
Above 30.0	Obese

Write a program that prompts the user to enter his/her weight and height. Then your program should calculate the BMI and tell the user his/her BMI and its interpretation.