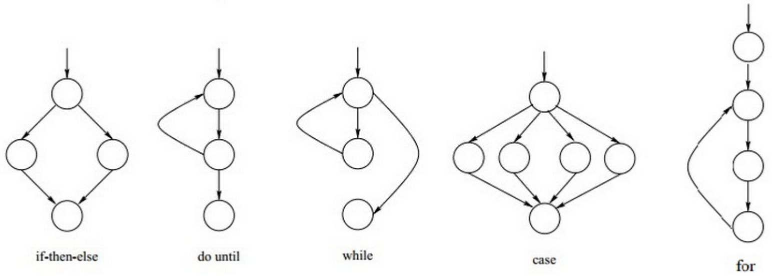
Control Structures in Java
if-else and switch

Lecture 7
CGS 3416 Spring 2020

January 28, 2020

Control Flow

Control flow refers to the specification of the order in which the individual statements, instructions or function calls of an imperative program are executed or evaluated



Types of Control Flow

Flow of control through any given function is implemented with three basic types of control structures:

- **Sequential**: Default mode. Statements are executed line by line.
- **Selection**: Used for decisions, branching – choosing between 2 or more alternative paths.
  - if
  - if - else
  - switch
  - conditional statements
- **Repetition**: Used for looping – repeating a piece of code multiple times in a row.
  - while
  - do - while
  - for

The function construct, itself, forms another way to affect flow of control through a whole program. This will be discussed later in the course.

Logical Operators

Java has Boolean operators for combining expressions. Each of these operators returns a Boolean value: a true or a false.

!x // the NOT operator (negation) – true if x is false

x && y // the AND operator – true if both x and y are true

x || y // the OR operator – true if either x or y (or both) are true

x ∧ y // the EXCLUSIVE OR operator – true if exactly one operand is true and one is false

These operators will be commonly used as test expressions in selection statements or repetition statements (loops).

Relational Operators

The comparison operators in Java work much like the symbols we use in mathematics. Each of these operators returns a Boolean value: a true or a false.

```
x == y     // x is equal to y
x != y     // x is not equal to y
x <y      // x is less than y
x <= y     // x is less than or equal to y
x >y      // x is greater than y
x >= y     // x is greater than or equal to y
```

Examples of Expressions

```
(x >0 && y >0 && z >0) // all three of (x, y, z)
                          are positive

(x <0 ||y <0 ||z <0) // at least one of the three
                       variables is negative

( numStudents >= 20 && !(classAvg <70)) // there are
         at least 20 students and the class average
         is at least 70

( numStudents >= 20 && classAvg >= 70) // means the
         same thing as the previous expression
```

Short Circuit Evaluation

- The && and ||operators also have a feature known as short-circuit evaluation.
- In the Boolean AND expression (X && Y), if X is false, there is no need to evaluate Y (so the evaluation stops). Example:

```
        (d != 0 && n / d >0)
```

- Notice that the short circuit is crucial in this one. If d is 0, then evaluating (n / d) would result in division by 0 (illegal). But the "short-circuit" prevents it in this case. If d is 0, the first operand (d != 0) is false. So the whole && is false.
- Similarly, for the Boolean OR operation (X ||Y), if the first part is true, the whole thing is true, so there is no need to continue the evaluation. The computer only evaluates as much of the expression as it needs. This can allow the programmer to write faster executing code.

if Statement

- The most common selection statement is the if statement. Basic syntax:

```
      if (boolean expression)
      {
            statement(s)
      }
```

- The condition is always a boolean expression. This means that it must be an expression that evaluates to a true or a false.

if - else Statements

- The if statement can also have an else clause. This is sometimes known as an if/else statement. Basic syntax:

```
if (boolean expression)
      {
            statement(s)
      }
      else
      {
            statement(s)
      }
```

- In both of these formats, the set braces can be left out if the "body" of the if or the else is a single statement. Otherwise, the block is needed.

Examples

-
  ```
   if (grade >= 68)
       System.out.print("Passing");
  ```

-
  ```
  if (x == 0)
       System.out.println("Nothing here");
  else
       System.out.println("There is a value");
  ```

Examples

-
  ```
  if (y != 4)
  {
       System.out.println("Wrong number");
       y = y * 2;
       counter++;
   }
  else
   {
       System.out.println("That's it!");
       success = true;
   }
  ```

Examples

- Be careful with ifs and elses. If you don't use { }, you may think that you've included more under an if condition than you really have.

```
if (val <5)
     System.out.println("True");
else
     System.out.println("False");
     System.out.println("Too bad!");
```

- Indentation is only for people! It improves readability, but means nothing to the compiler.

Some Common Errors

What's wrong with these if-statements? Which ones are syntax errors and which ones are logic errors?

-
  ```
  if (x == 1 ||2 ||3)
        System.out.print("x is a number in the
                     range 1-3");
  ```

-
  ```
  if (x >5) && (y <10)
        System.out.print("Yahoo!");
  ```

-
  ```
  if (response != 'Y' ||response != 'N')
        System.out.print("You must type Y or N
                  (for yes or no)");
  ```

The switch Statement

A switch statement is often convenient for occasions in which there are multiple cases to choose from. The syntax format is:

```
switch (expression)
{
    case constant:
        statements
    case constant:
        statements

    ...(as many case labels as needed)

    default:            // optional label
        statements
}
```

The switch Statement

- The switch statement evaluates the expression, and then compares it to the values in the case labels. If it finds a match, execution of code jumps to that case label.
- The values in case labels must be constants, and may only be types char, byte, short, or int. From Java 7 onwards, Strings and Enum types are also allowed.
  - This also means the case label must be a literal or a variable declared to be constant (with final).
  - You may not have case labels with regular variables, floating point literals, operations, or function calls
- If you want to execute code only in the case that you jump to, end the case with a break statement, otherwise execution of code will "fall through" to the next case.

The Conditional Operator

There is a special operator known as the conditional operator that can be used to create short expressions that work like if/else statements.

- **Format:**
  `boolean_expr ?  true_expr :  false_expr`
- **How it works:**
  - The boolean expression is evaluated for true/false value. This is like the test expression of an if-statement.
  - If the expression is true, the operator returns the true expression value.
  - If the test expression is false, the operator returns the false expression value.
- Note that this operator takes three operands. It is the one ternary operator in the Java language

Some Examples

```
 System.out.print( (x >y) ?  "x is greater than y" :
               "x is less than or equal to y");

// Note that this expression gives the same result as
the following

     if (x >y)
          System.out.print("x is greater than y");
     else
          System.out.print("x is less than or equal
                                to y");
```

Some Examples

```
 (x <0 ?  value = 10 :  value = 20);

// this gives the same result as:

    value = (x <0 ?  10 :  20);

// and also gives the same result as:
    if (x <0)
         value = 10;
    else
         value = 20;
```