

Java Control Statements

Introduction to the Java Programming Language

Produced
by

Eamonn de Leastar
edeleastar@wit.ie

Department of Computing, Maths & Physics
Waterford Institute of Technology

<http://www.wit.ie>

<http://elearning.wit.ie>



Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



Essential Java

⊕ **Overview**

- ⊕ Introduction
- ⊕ Syntax
- ⊕ Basics
- ⊕ Arrays

⊕ **Classes**

- ⊕ Classes Structure
- ⊕ Static Members
- ⊕ Commonly used Classes

⊕ **Control Statements**

- ⊕ Control Statement Types
- ⊕ If, else, switch
- ⊕ For, while, do-while

⊕ **Inheritance**

- ⊕ Class hierarchies
- ⊕ Method lookup in Java
- ⊕ Use of this and super
- ⊕ Constructors and inheritance
- ⊕ Abstract classes and methods
- Interfaces

⊕ **Collections**

- ⊕ ArrayList
- ⊕ HashMap
- ⊕ Iterator
- ⊕ Vector
- ⊕ Enumeration
- ⊕ Hashtable

⊕ **Exceptions**

- ⊕ Exception types
- ⊕ Exception Hierarchy
- ⊕ Catching exceptions
- ⊕ Throwing exceptions
- ⊕ Defining exceptions
- Common exceptions and errors

⊕ **Streams**

- ⊕ Stream types
- ⊕ Character streams
- ⊕ Byte streams
- ⊕ Filter streams
- ⊕ Object Serialization

Overview

- ⊕ What are control statements
- ⊕ Different control statement types
- ⊕ if, if-else, and switch statement
- ⊕ for, while, and do-while statements

What are Control Statements?

- ⊕ Control statements are statements that control execution of other statements
 - ⊕ These other statements can be either selection or repetition statements
 - ⊕ Also called conditional and looping statements

Control Statement Types

- ⊕ There are two general types of control statements:
 - ⊕ Selection (conditional) statements
 - ⊕ if, if-else, and switch statements
 - ⊕ Repetition (looping) statements
 - ⊕ for, while, and do-while statements
- ⊕ There are also some other control statements specific to Java
 - ⊕ break, continue, and labels

Using if Statement

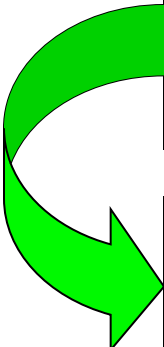
- ⊕ if statement is used for selecting whether or not to do something
- ⊕ The reserved word if is used with the expression in parentheses
 - ⊕ If the expression evaluates to true the statement after the expression will execute

```
if (condition)
{
    statement;
}
```

Example if Statement

```
int i = 1;
if(i > 0)
{
    System.out.println("Greater than zero");
}
```

=



```
int i = 1;
if(i > 0)
    System.out.println("Greater than zero");
```

```
int i = 3;
if(i > 2)
{
    System.out.println("Greater than zero");
    System.out.println("Greater than one");
    System.out.println("Greater than two");
}
```

Using if-else Statement

- ⊕ if-else statement is used for selection which one of two statements will execute
 - ⊕ If control statement condition evaluates to true the next statement (after if) executes
 - ⊕ If it evaluates to false the statement under else executes
- ⊕ Braces can be omitted if single statements used

```
if (condition)
{
    statement1;
}
else
{
    statement2;
}
```

=

```
if (condition)
    statement1;
else
    statement2;
```


Example if-else Statement

```
int i = 1;
if(i > 0)
    System.out.println("Greater than zero");
else
    System.out.println("Not greater than zero");
```

```
int i = 3;
if(i > 2)
{
    System.out.println("Greater than zero");
    System.out.println("Greater than one");
    System.out.println("Greater than two");
}
else
{
    System.out.println("Either equal to two");
    System.out.println("Or less than two");
}
```

Nested if statements

- ⊕ Any form of if statement can be nested
 - ⊕ There can be other if statements within of if statements

```
if (condition)
{
    if(nested_condition1)
    {
        nested_action1;
    }
    else
    {
        if(nested_condition2)
        {
            nested_action2;
        }
        else
        {
            nested_action3;
        }
    }
}
else
{
    action2;
}
```

Example Nested Statement

```
int i = 3;
if(i > 2)
{
    System.out.println("Greater than zero");
    System.out.println("Greater than one");
    System.out.println("Greater than two");
}
else
{
    if(i == 2)
        System.out.println("Equal to two");
    else
        System.out.println("Less than two");
}
```

Using switch Statement

- ⊕ switch statement is used when the execution of statements depends on different values of a variable or expression
 - ⊕ The variable must be of type (or expression must evaluate to the type of) byte, char, short or int
- ⊕ Uses switch keyword

```
switch (expression)
{
    case value1:
        expression1;
    case value2:
        expression2;
        expression3;
    default:
        expression4;
}
```

Using break statement

- ⊕ Allows to stop execution after executing an expression
 - ⊕ Other expressions from other cases are not executed
 - ⊕ Execution continues after the switch statement

```
switch (expression)

    case value1:
        expression1;
        break;
    case value2:
        expression2;
        break;
    default:
        expression3;
}
```

Example switch statement...

```
int i = 2;
switch(i)
{
    case 1: System.out.println("1");
    case 2: System.out.println("2");
    case 3: System.out.println("3");
    default: System.out.println("default");
}
```



Console

```
2
3
default
```

```
int i = 2;
switch(i)
{
    case 1: System.out.println("1"); break;
    case 2: System.out.println("2"); break;
    case 3: System.out.println("3"); break;
    default: System.out.println("default");
}
```



Console

```
2
```

...Example switch statement

```
int i = 2;
switch(i)
{
    case 1:
        System.out.println("1");
        break;
    case 2:
    case 3:
        System.out.println("2");
        System.out.println("or");
        System.out.println("3");
        break;
    default:
        System.out.println("default");
}
```



Console

```
2
or
3
```

Using for Statement

- ⊕ for statement is used for looping, i.e. repetition in a code
- ⊕ Consists of for keyword and parentheses containing:
 - ⊕ Index declaration and initialization section
 - ⊕ Index test section
 - ⊕ Index increment section

```
for (initialization; test; increment)
{
    statement;
}
```


Example for Statement

```
for(int i=1; i<5; i++)  
{  
    System.out.println("Index is equal to " + i);  
}
```



Console

```
Index is equal to 1  
Index is equal to 2  
Index is equal to 3  
Index is equal to 4
```

```
int i=1;  
for(;;)  
{  
    System.out.println("Infinite loop");  
    if(i==2) break;  
    i++;  
}
```



Console

```
Infinite loop  
Infinite loop
```

Using while Statement

- ⊕ while statement is used for repeating statements while some condition applies
- ⊕ Condition is evaluated before the statement
 - ⊕ Condition evaluates to either true or false
 - ⊕ If condition evaluates to true statement executes
 - ⊕ If condition evaluates to false statement does not execute, and evaluation proceeds after the block
- ⊕ Uses while keyword

```
while (condition)
{
    statement;
}
```

Example while Statement

```
int i=1;
while (i < 5)
{
    System.out.println(i);
    i++;
}
```



Console

```
1
2
3
4
```

```
int i=5;
while (i < 5)
{
    System.out.println(i);
    i++;
}
```



Console

Using do-while Statement

- ⊕ Similar to the while statement
 - ⊕ Condition is evaluated at the end of the statement
 - ⊕ Block is executed at least once
- ⊕ Uses do and while keywords

```
do
{
    statement;
} while (condition);
```

Example do-while Statement

```
int i=1;
do
{
    System.out.println(i);
    i++;
}while (i < 5);
```



Console

```
1
2
3
4
```

```
int i=5;
do
{
    System.out.println(i);
    i++;
}while (i < 5);
```

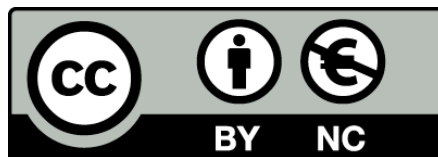


Console

```
5
```

Summary

- ⊕ What are control statements
- ⊕ Different control statement types
- ⊕ if, if-else, and switch statement
- ⊕ for, while, and do-while statements



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

