

# Java Control Statements

An introduction to the Java Programming Language

---

Produced      Eamonn de Leastar ([edeleestar@wit.ie](mailto:edeleestar@wit.ie))  
by:          Dr. Siobhan Drohan ([sdrohan@wit.ie](mailto:sdrohan@wit.ie))



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



Centre for  
Technology-Enhanced Learning

# 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

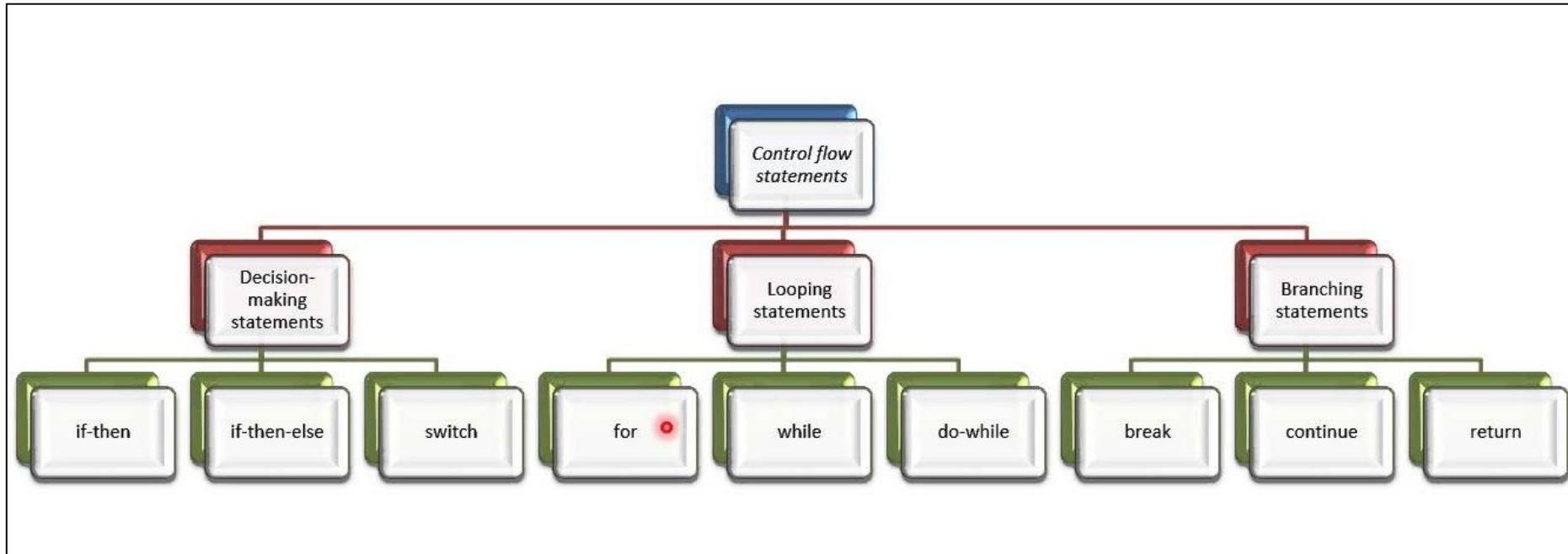
# Overview: Road Map

---

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

# What are Control Statements?

---



Control statements are statements that control execution of other statements

# Overview: Road Map

---

- Control Statement Types

- If, else, switch

- For, while, do-while

# if statement syntax

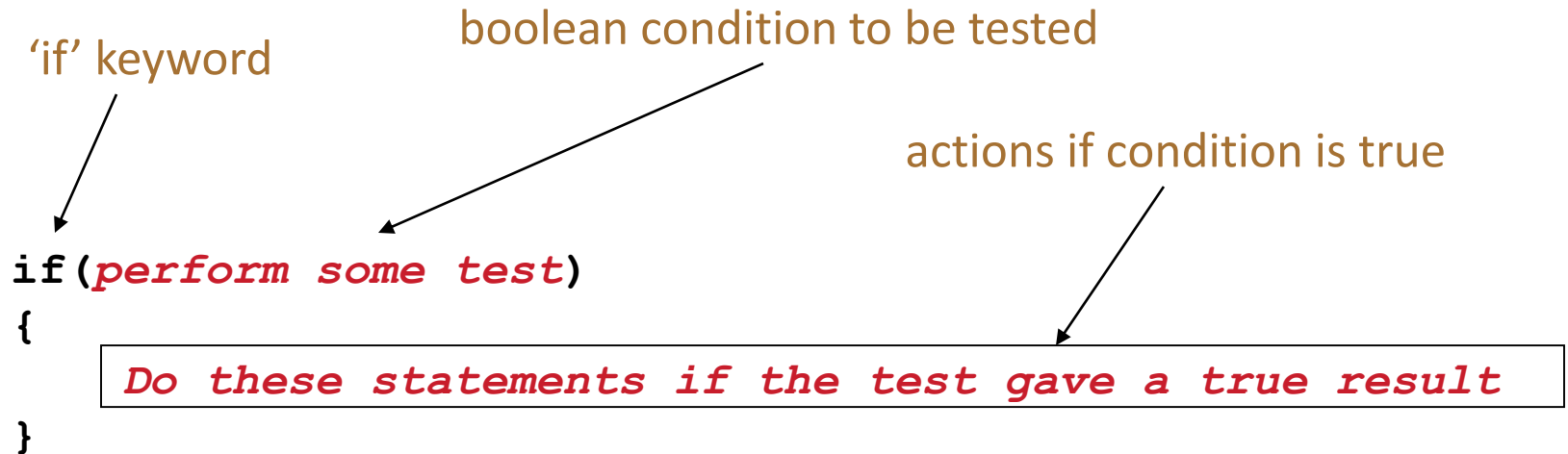
---

'if' keyword

boolean condition to be tested

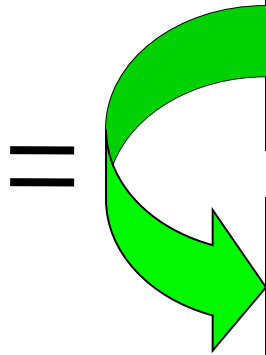
actions if condition is true

```
if(perform some test)  
{  
    Do these statements if the test gave a true result  
}
```

The diagram illustrates the syntax of an if statement. It shows the code: `if(perform some test)`, `{`,  `Do these statements if the test gave a true result`, and `}`. Three labels with arrows point to specific parts: 'if' keyword points to the `if` keyword; boolean condition to be tested points to the `(perform some test)` part; and actions if condition is true points to the indented block of code inside the curly braces.

# Example if Statement

---



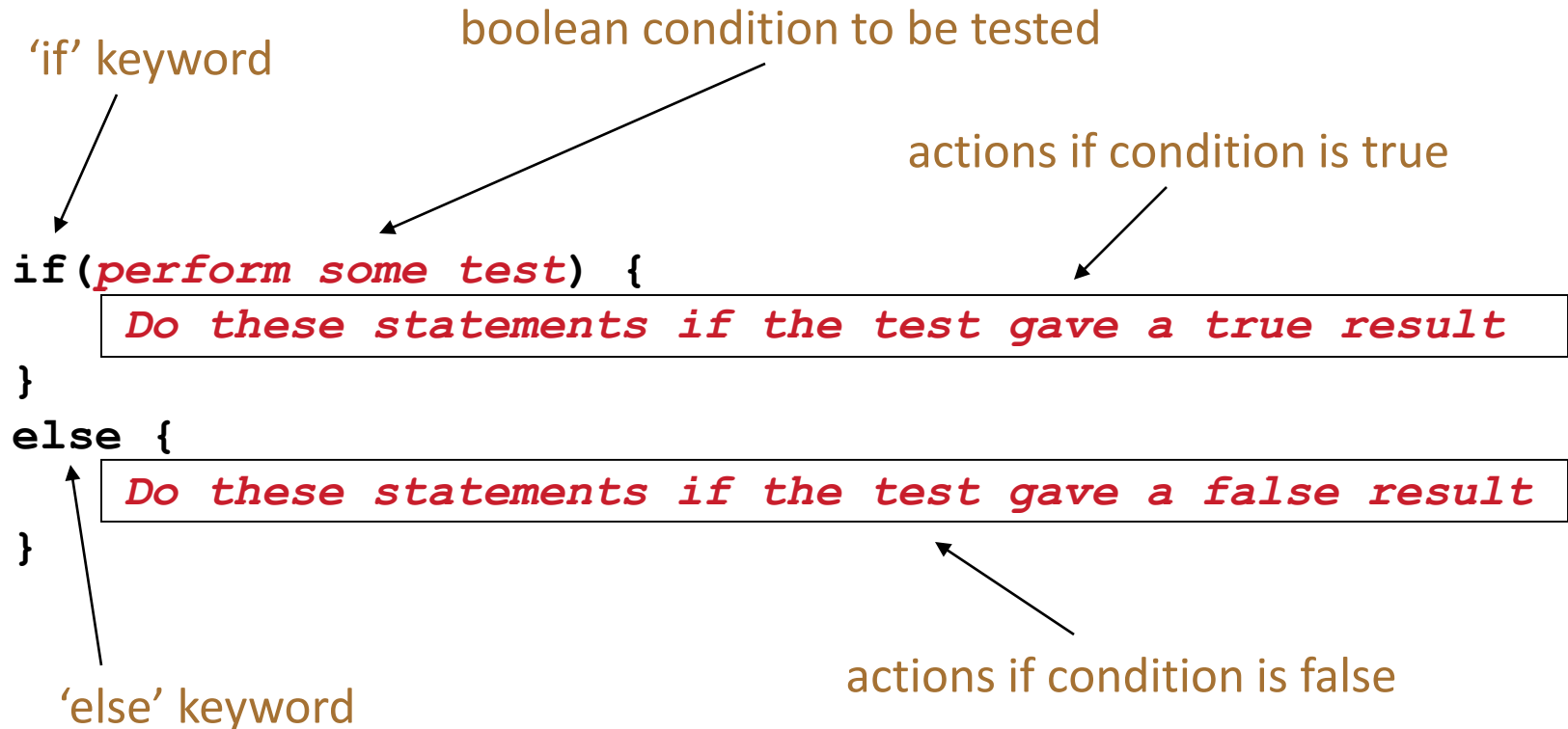
```
int i = 1;
if(i > 0)
{
    System.out.println("Greater than zero");
}
```

```
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");
}
```

# if-else statement syntax

---





# Example if-else Statement

---

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

Braces can be omitted if single statements used.

```
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");
}
```

# More if statement syntax

---

```
if(condition1...perform some test)
```

```
{
```

*Do these statements if condition1 gave a true result*

```
}
```

```
else if(condition2...perform some test)
```

```
{
```

*Do these statements if condition1 gave a false result and condition2 gave a true result*

```
}
```

```
else
```

```
{
```

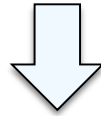
*Do these statements if both condition1 and condition2 gave a false result*

```
}
```

# The switch statement – pattern one

---

Pre Java 7: can switch on int and char.  
Post Java 7: can also switch on String



```
switch(expression) {  
    case value: statements;  
                break;  
    case value: statements;  
                break;  
    further cases possible  
    default: statements;  
            break;  
}
```

A *switch* statement can have any number of **case** labels.

# The switch statement – pattern two

---

```
switch(expression) {  
    case value1:  
    case value2:  
    case value3:  
        statements;  
        break;  
    case value4:  
    case value5:  
        statements;  
        break;  
    further cases possible  
    default:  
        statements;  
        break;  
}
```

The **break** statement stops execution “falling through” into the next label’s statements.

The **default** case is optional.

# The switch statement – example 1

---

```
switch(day) {  
    case 1:  dayString = "Monday";  
             break;  
    case 2:  dayString = "Tuesday";  
             break;  
    case 3:  dayString = "Wednesday";  
             break;  
    case 4:  dayString = "Thursday";  
             break;  
    case 5:  dayString = "Friday";  
             break;  
    case 6:  dayString = "Saturday";  
             break;  
    case 7:  dayString = "Sunday";  
             break;  
    default: dayString = "invalid day";  
             break;  
}
```

# The switch statement – example 2

---

```
switch(dow.toLowerCase()) {  
    case "mon":  
    case "tue":  
    case "wed":  
    case "thu":  
    case "fri":  
        goToWork();  
        break;  
    case "sat":  
    case "sun":  
        stayInBed();  
        break;  
}
```



# The switch statement – example 3

---

```
switch (group){  
    case 'A':  
        System.out.println("10.00 a.m ");  
        break;  
    case 'B':  
        System.out.println("1.00 p.m ");  
        break;  
    case 'C':  
        System.out.println("11.00 a.m ");  
        break;  
    default:  
        System.out.println("Enter option A, B or C only!");  
}
```

# 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

# Overview: Road Map

---

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

# for loop - pseudo-code

---

General form of a for loop

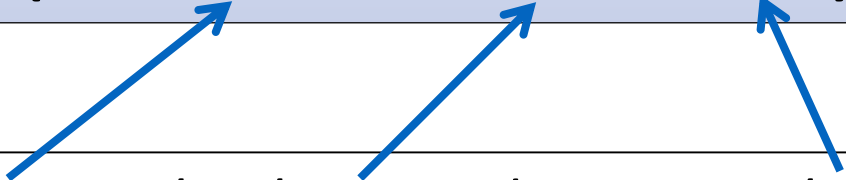
```
for(initialization; boolean condition; post-body action)  
{  
    statements to be repeated  
}
```

# for loop - syntax

---

```
for(int i = 0; i < 4; i++)
```

```
for(initialization; boolean condition; post-body action)  
{  
    statements to be repeated  
}
```



# for loop - example

---

```
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++;  
}
```

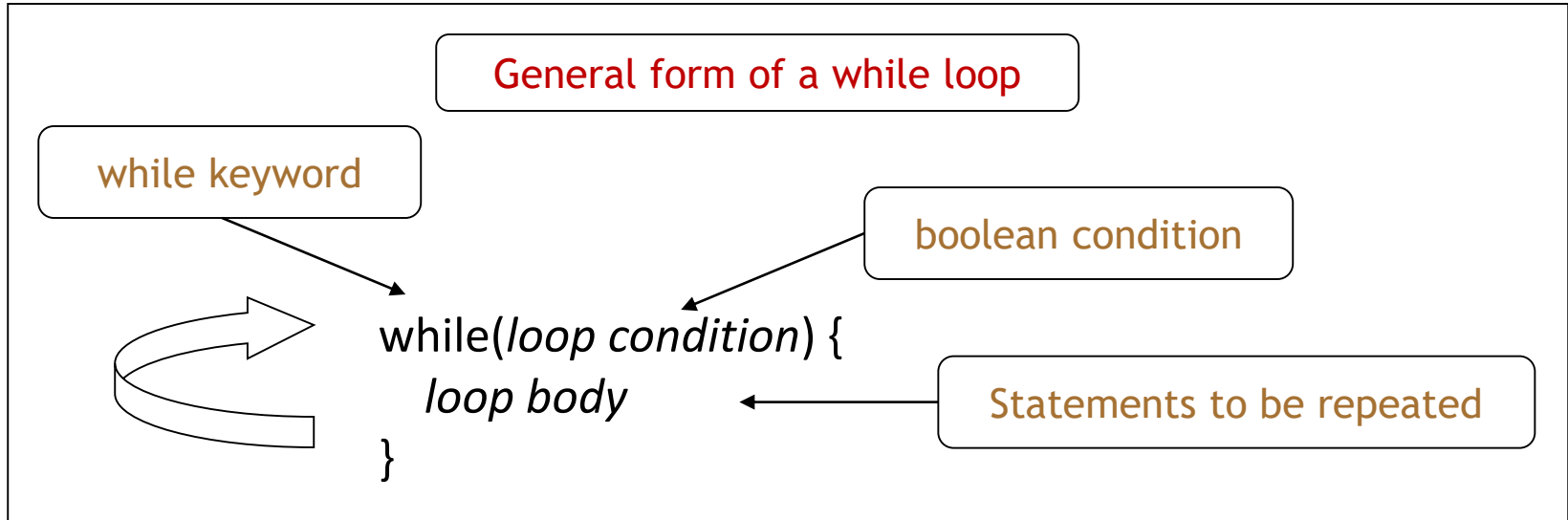


Console

```
Infinite loop  
Infinite loop
```

# while loop - pseudo code

---



## Pseudo-code expression of the actions of a while loop

while we wish to continue, do the things in the loop body



# while loop

---

```
Declare and initialise loop control variable (LCV)
while(condition based on LCV)
{
    "do the job to be repeated"
    "update the LCV"
}
```

This structure should always be used

# while - example

---

```
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

# do-while

---

- ⊕ Similar to the while statement:
  - ⊕ Condition is evaluated at the end of the statement
  - ⊕ block is executed at least once

```
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

# Topics covered in this lecture:

## ⊕ **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



Any questions?