## Java Syntax

## System.out.println()

Inside the main() method, we can use the println() method to print a line of text to the screen:

public class Main {

public static void main(String[] args) {

System.out.println("Hello World");

}

}

## Java Comments

Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

## Single-line Comments

Single-line comments start with two forward slashes (//).

public class Main {

public static void main(String[] args) {

// This is a comment

System.out.println("Hello World");

}

}

## Java Multi-line Comments

Multi-line comments start with /\* and ends with \*/.

public class Main {

public static void main(String[] args) {

/\* The code below will print the words Hello World

to the screen, and it is amazing \*/

System.out.println("Hello World");

}

}

## Java Variables

Variables are containers for storing data values.

In Java, there are different **types** of variables, for example:

* String - stores text, such as "Hello". String values are surrounded by double quotes
* int - stores integers (whole numbers), without decimals, such as 123 or -123
* float - stores floating point numbers, with decimals, such as 19.99 or -19.99
* char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
* boolean - stores values with two states: true or false

## Declaring (Creating) Variables

To create a variable, you must specify the type and assign it a value:

*type variableName = value;*

## Java Data Types

As explained in the previous chapter, a [variable](https://www.w3schools.com/java/java_variables.asp) in Java must be a specified data type:

int myNum = 5; // Integer (whole number)  
float myFloatNum = 5.99f; // Floating point number  
char myLetter = 'D'; // Character  
boolean myBool = true; // Boolean  
String myText = "Hello"; // String

Data types are divided into two groups:

* Primitive data types - includes byte, short, int, long, float, double, boolean and char
* Non-primitive data types - such as [String](https://www.w3schools.com/java/java_strings.asp), [Arrays](https://www.w3schools.com/java/java_arrays.asp) and [Classes](https://www.w3schools.com/java/java_classes.asp) (you will learn more about these in a later chapter)

## Java Type Casting

## Widening Casting

Widening casting is done automatically when passing a smaller size type to a larger size type:

public class Main {  
 public static void main(String[] args) {  
 int myInt = 9;  
 double myDouble = myInt; // Automatic casting: int to double  
  
 System.out.println(myInt); // Outputs 9  
 System.out.println(myDouble); // Outputs 9.0  
 }  
}

## Narrowing Casting

Narrowing casting must be done manually by placing the type in parentheses in front of the value:

public class Main {  
 public static void main(String[] args) {  
 double myDouble = 9.78d;  
 int myInt = (int) myDouble; // Manual casting: double to int  
  
 System.out.println(myDouble); // Outputs 9.78  
 System.out.println(myInt); // Outputs 9  
 }  
}

## Java Conditions and If Statements

## The if Statement

Use the if statement to specify a block of Java code to be executed if a condition is true.

if (*condition*) {  
 *// block of code to be executed if the condition is true*  
*}*

if (20 > 18) {  
 System.out.println("20 is greater than 18");  
}

## The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

if (*condition*) {  
 *// block of code to be executed if the condition is true*  
*}* else {  
 *// block of code to be executed if the condition is false*  
*}*

## The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

if (*condition1*) {  
 *// block of code to be executed if condition1 is true*  
*}* else if (*condition2*) {  
 *// block of code to be executed if the condition1 is false and condition2 is true*  
*}* else {  
 *// block of code to be executed if the condition1 is false and condition2 is false*  
*}*

# Java Switch

## Java Switch Statements

Instead of writing **many** if..else statements, you can use the switch statement.

The switch statement selects one of many code blocks to be executed:

switch(*expression*) {  
 case x:  
 *// code block*  
 break;  
 case y:  
 *// code block*  
 break;  
 default:  
 *// code block*  
*}*

int day = 4;  
switch (day) {  
 case 1:  
 System.out.println("Monday");  
 break;  
 case 2:  
 System.out.println("Tuesday");  
 break;  
 case 3:  
 System.out.println("Wednesday");  
 break;  
 case 4:  
 System.out.println("Thursday");  
 break;  
 case 5:  
 System.out.println("Friday");  
 break;  
 case 6:  
 System.out.println("Saturday");  
 break;  
 case 7:  
 System.out.println("Sunday");  
 break;  
}

# Java While Loop

## Java While Loop

The while loop loops through a block of code as long as a specified condition is true:

while (*condition*) {  
 *// code block to be executed*  
*}*

## The Do/While Loop

The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

do {  
 *// code block to be executed*  
*}*  
*while* (*condition*);

# Java For Loop

## Java For Loop

When you know exactly how many times you want to loop through a block of code, use the for loop instead of a while loop:

for (*statement 1*; *statement 2*; *statement 3*) {  
 *// code block to be executed*  
*}*

The example below will print the numbers 0 to 4:

for (int i = 0; i < 5; i++) {  
 System.out.println(i);  
}