INTRODUCTION TO JAVA

```
public class classname{
        public static void main(String[] args){

        }
}
```

METHODS

This gives a well organised way of coding.The same code with plenty of lines can be divided to differrent blocks so that it becomes sensible than 100s of lines of code not arranged.It will present inside the class but outside the main method.

For ex:

```
public class sample{
        public static void main (Strings[] args){
                Helloworld();                           //call statement
        }


        public static void HelloWorld(){
                System.out.println("Hello world");
        }

}
```

PROCEDURAL DECOMPOSITION

It is the dividing of the millions of lines of code into small subdivisions(Divide and conquer method).Java does it by "**method**".
Method helps us to avoid redundancy of code.

Use camelcasing for class or method name.

***camelcasing** : start with small letter and continue starting other words by capital letter.
For ex: flyOff

DECLARATION OF VARIABLES

dataType variableName;

INITIALISATION

variableName = value;

SCOPE OF A VARIABLE

A variable declared within the curly braces of a method is called the l**ocal variable**.This cannot be accessed from outiside the method.
But a variables scope can be extended throughout the class by declaring it as a **class variable.**Since the value of the variable is a constant throughout the class it is called a **class constant**.

Syntax: **public static final dataType VARIABLENAME = value;**

for ex:`public static final double PI = 3.14;`

*java convention to use capital letters for class constants.

MATHEMATICAL OPERATIONS

When two numbers are subjected to any mathematical operation,the type of theresult will be that of one with largest bytes.If a float and an integer is added , the final result will be a float.

STRING CONCATENATION

When a string is concatenated with another data type,it returns a string.
For ex: 123 + “abc” returns “123abc”
10.6 + “123” returns “10.6123”
“” + 123 returns “123”
“is ” + true returrns “is true”

1 + 0 + “0” + 5 * 10 returns “1050” (intially it adds 1 and 0 as normal addition,but once it encounters a char then “+” will then act as a string concatenator.)

CASTING

Syntax: (resulting datatype)expression;

for ex: (int)(4/3) returns 1
while (int)4/3 also returns 1
but (int)(10/4.0) returns 2 while (int)10/4.0 returns 2.5 as it takes care of only numerator.

Casting has higher precedence than any other operator except paranthesis.

Syntax to convert int to double

(double)10; returns 10.0

BOOLEAN

Returns either true or false.
Uses logical operators like =,<,>,!= etc.
But th same method shows an error whe used to compare strings since they are oblects.
So here we use the “.equals” operation where one stirng is passed as a parameter.

“string”.equals(“string”)

***can be made case insensitive using “equalsIgnorecase” instead of simply “equals”.**

Similarly this method can be used for many purposes like to check starting letter or ending letter .
For ex:

“string”.startsWith(“s”)	-> true
“string”.endsWith(“t”)	->false
“string”.contains(“ring”)	->true
“string”.equalsIgnorecase(“STRING”)	->true
“string”.equals(“STRING”)	->false

METHOD OVERLOADING

It is the ability of a different methods to have the same name if the arguments are different.The datatypes of the oveloaded methods must be different or in different order.

*Static = no return type

JAVA MATHS

method header	**example**	**summary**
< any > abs(< any > x)	int x = Math.abs(-2);	returns the absolute value of x
double pow(double base, double exp)	double x = Math.pow(10,2);	returns the result of base to the power exp
double sqrt(double x)	double x = Math.sqrt(25.0);	returns the square root of x
double random()	double x = Math.random();	returns a randomly generated number between 0 and 1

```
Math.PI = 3.14159…
Math.E = 2.71828
```

STRINGS

Length
“stringname”.length();

To get substring,
“stringname”.substring(0,4);
“stringname”.substring(1,3);
as per index numbering 0 to n-1

To get index,

“stringname”.indexOf(“name”); returns starting index of matching string

method	example	example results	summary
boolean equals(String)	"hello".equals("good bye")	false	true if strings are identical, case sensitive
char charAt(int index)	"hello".charAt(2)	'l'	char at index
int lastIndexOf(String str)	"hello".lastIndexOf("l")	3	position of last occurrence of String
String toLowerCase()	"HELLO".toLowerCase()	"hello"	new string converted to all lowercase
String toUpperCase()	"hello".toUpperCasE()	"HELLO"	new string converted to all uppercase
String replace(char old, char new)	"hello".replace('l', 'x')	"hexxo"	new string with all occurrences of old replaced by new

RECURSION

The ability of a method to call itself.

*base case = smallest possible case
*recursive case