JAVA Programming Notes

Introduction to Java :-

Java is a popular - general purpose programming language with a wide range of applications.

Its' uses :-
- ↳ developing mobile / desktop applications
- ↳ big data processing.
- ↳ embedded systems etc. ....

Popularity :- Being one of the older programming languages, Java still has a huge demand in the current job market.

This course is an introduction to Java programming in the most attractive way possible.

learn a concept along with codes.

Basic Structure of Java code :-

```
class Main {
    public static void main (String[] args) {
        System.out.println ("Hello, World!");
    }
}
```

Before we begin with coding, we need to understand some points which are beneficial for a coder :-

1) Fear of coding :-

Programming may feel intimidating for beginners. If you give it time, you'll master it, no doubt.

2) Programming is a language :-

Similar to languages like English, programming is also a language in a sense.

You just need to learn the syntax.

Syntax means rules of grammar to learn English.

3) Math for Programming :-

Learning to code involves a lot of logical and trial-and-error, however, nothing beyond basic arithmetic.

Basics of Numbers and Text

There are a few things we need to understand first, even to write simple programs. We will cover the following concepts in this lesson.

In programming we can directly use some **fixed values** in our programs. These fixed values are called **literals**

1) Integers (int) ex → 2, -4, 3, 5, -10
2) Floating point numbers (double) ex → 1.5, 2.4
3) String (text) ex → "Python".

Quiz:-

1) "Java" is a ---- ?

A) int
B) String ✓
C) floating-point number
D) None

2) 9.0 is a --- ?

A) int
B) String
C) double ✓
D) None

Print Numbers and Texts

You might remember this from our Hello World program.

⇒ `System.out.println("Hello, World");`

- `System.out.println` — used to print on screen
- `("Hello, World")` — what has to be print
- `;` — end of statement

eg:

Program - to print a number:-

```
class Main {
    Public static void main (String[] args) {
        System.out.println (33);
    }
}
```

output: 33

Here, `System.out.println()` prints the content inside (), which is 33 in this program.

eg:- class Main{
    Public static void main (String[] args){
        System.out.println ("Java is easy.");
    }
}

output: Java is easy.

NOTE:- when we print strings, the outer quotation marks are not displayed.

⇒ COMMON MISTAKES

1) Errore because of quotation marks.
2) forgetting semi colon in the print statement.

NOTE:- we can include more than one print statement in our Java code.

NOTE: Two different print statements prints on two different lines by default.

eg:- class Main{
    Public static void main (String[] args){
        System.out.println ("I love Java");
    }
}

Introduction to Comments

In programming, we can make computers ignore certain parts of the program by using comments.

In Java, we use `//` (two forward slashes) to start a comment.

eg:

```
class Main {
    public static void main (String[] args)
    {
        // print Hello World
        System.out.println ("Hello World");
    }
}
```

here, // print Hello World is a comment. This part is completely ignored by the compiler.

NOTE: Comments are used as hints that we add to our program to make our code easier to understand.

They are mainly used for humans.

Quiz — What is the correct way to comment in Java?

Ans:- //

Java Variables

Variables are containers for storing data values. In Java, there are different type of variables, for ex-

1) String ⟶ "Java"
2) int ⟶ 2
3) float ⟶ 4.5
4) char ⟶ 'A', 'a'
5) boolean ⟶ True | False

Syntax

```
type variableName = value;
```

eg:

```
String name = "Ram"
System.out.println(name)
```

eg:

```
Public class Main {
    Public static void main (String[] args)
    {
        int myNum = 15;
        System.out.println(myNum);
    }
}
```

Identifiers

All Java variables must be identified with unique names. These unique names are called identifiers.

NOTE: It is recommended to use descriptive names in order to create understandable and maintainable code.

eg:- `int a = 10` → value

a → identifier.

int → data type

General rules for naming variable

1) Names can contain letters, digits, underscores, and dollar signs.
2) Names must begin with a letter.
3) Name should start with a lowercase letter and it cannot contain white space
4) Names can also start with $ and _.
5) Names are case sensitive.
6) Reserved words can't used as names. Such as int, boolean etc.

Java Data Types

A variable in Java must be a specified data type.

Data types are divided into two groups :-

1) Primitive data types
2) Non-primitive data types.

1) Primitive data types -
includes byte, short, int, long, float, double, boolean and char.

2) Non-primitive data types -
such as string, Arrays and Classes.

data type	Size	Description
Byte	1 byte	Stores whole numbers from -128 to 127.
short	2 bytes	Stores whole number from -32,768 to 32,767
int	4 bytes	Stores whole number from -2147483648 to 2147 483647

long	8 bytes	Stores whole numbers from -9223 372 036 854 775 808 to 9223 372 036 854 775 807.
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits.
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits.
boolean	1 bit	Stores true or false values.
char	2 bytes	Stores a single character/ letter or ASCII values.

JAVA TYPE CASTING

Type casting is when you assign a value of one primitive data type to another type.

In Java, there are two types of casting.

1) Widening Casting (automatically) :-
converting a smaller type to a larger type size.

⇒ byte → short → char → int → long → float → double

2) Narrowing casting (manually) :-
converting a larger type of to a smaller size type.

⇒ double → float → long → int → char → short → byte

Widening Casting

eg:-

```
Public class Main {
    Public static void main (String[] args) {
        int myInt = 9;
        double myDouble = myInt;
        // automatic casting int to double

        System.out.println (myInt);    //output → 9
        System.out.println (myDouble); // output → 9.0
    }
}
```

Eg:-

Narrowing Casting

```
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);    // output 9
    }
}
```

Java Operators

operators are used to perform operations on variable and values.

Java divides the operators into the following groups -

1) Arithmetic operators
2) Assignment operators
3) Comparison operators
4) Logical operators
5) Bitwise operators.

operator → + Name → Addition

eg:-
```
Public class Main {
    Public static void main (String[] args)
    {
        int x = 5;
        int y = 3;
        System.out.println (x+y);
    }
}
```

operator → - Name → Subtraction

eg:-
```
Public class Main {
    Public static void main (String [] args)
    {
        int x = 5;
        int y = 3;
        System.out.println (x-y);
    }
}
```

operator → * Name → multiplication

eg:- System.out.println (x * y);

operator → / name → Division

eg:- System.out.println (x/y);

operator → % Name → Modulus

eg:- System.out.println(x%y);

operator → ++ Name → Increment

eg:-
```
int x = 5;
++x;
System.out.println(x);
```
// output 6

operator → -- Name → Decrement

eg:-
```
int x = 5;
--x;
System.out.println(x);
```
// output 4

Java Assignment operator

eg:-
```
int x = 10;
```

eg:-
```
int x = 10;
x += 5;
```
// output is

[same as].

other assignment operators :-

=	%=	>>=
+=	&=	<<=
-=	|=	
*=	^=	
/=		

Java comparison operator

eg:-

```
int x = 5;                    // returns true
int y = 3;
System.out.println(x>y);
```

other comparison operators :-

==	>	>=
!=	<	<=

Java logical Operators

operator → logical and (&&)

eg:-

```
int x = 5;
System.out.println(x>3 && x<10);
// returns true. because both the conditions are true.
```

operator :- Logical or (||) :-

eg :- int x = 5;
System.out.println(x>3 || x<4);
// returns true because one of the conditions satisfied.

operator :- logical not (!) :-

eg :- int x = 5;
System.out.println(!(x>3 && x<10));
// returns false as it reverse the result.