Shri Ganeshaya Namaha.

BASICS OF JAVA

● IMPORTANT TERMINOLOGY :—

① CLASS → Blueprint or template → defines properties & behaviours (TO CREATE MULTIPLE INSTANCES)

② OBJECT → Instance of a class

③ PACKAGE → Way to organise related classes & interfaces in Java (Mechanism for avoiding naming conflicts)

④ INHERITANCE → Allowing class to inherit properties of another class.

⑤ POLYMORPHISM → Objects of diff. classes → OBJECT OF 1 SUPER-CLASS

⑥ INTERFACE → Collections of abstract methods → SPECIFIES SET OF BEHAVIOURS

⑦ EXCEPTION → Prevents natural flow of code.

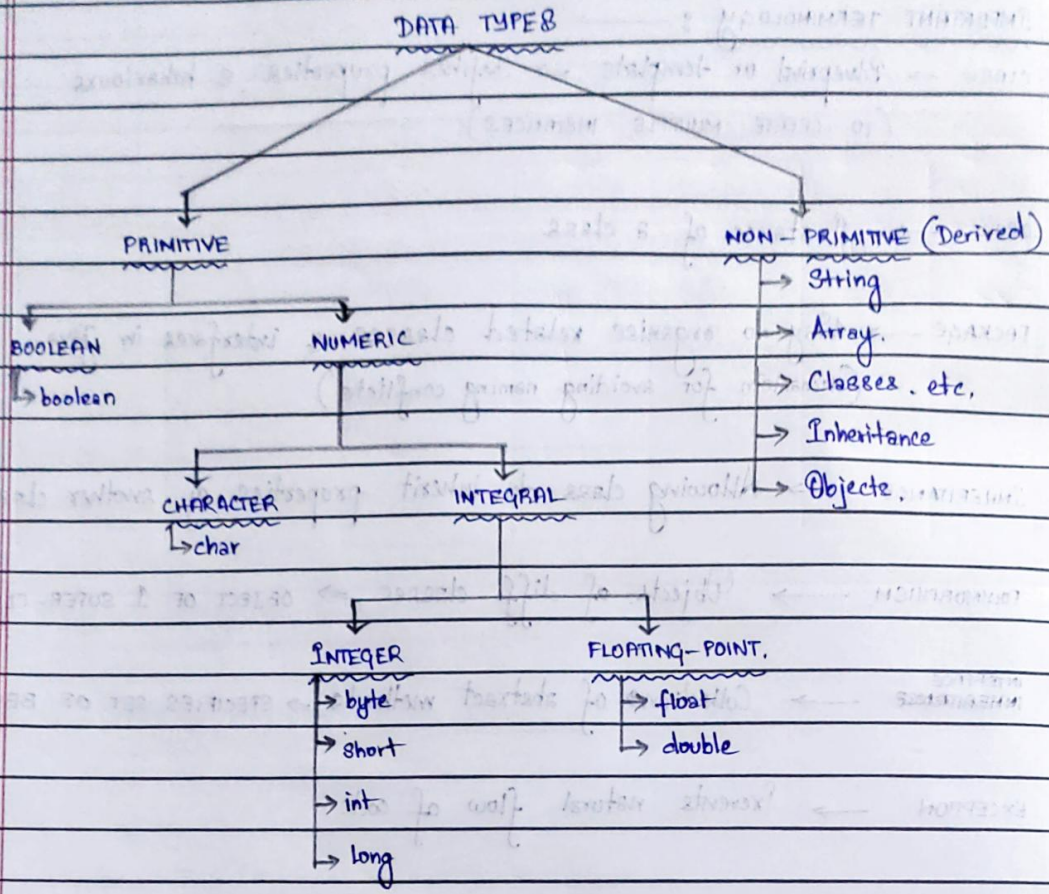
★ FIRST CODE :—

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

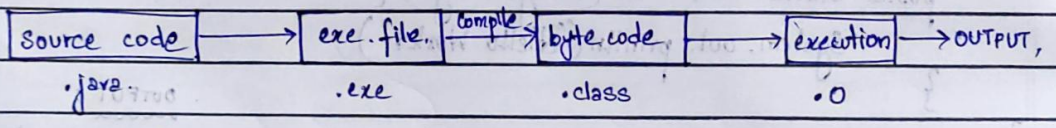
OUTPUT :
Hello World.

* 2 types of print statements ⇒ print & println

★ DATA TYPES IN JAVA :—



★ COMPILING & RUNNING :—



Byte code ⟶ language of virtual machine

Interpreter ⟶ bytecode by compiler ⇒ fed to interpreter

★ VARIABLES / IDENTIFIERS IN JAVA :

eg: int a, b, c ;
int d = 3, e, f = 5 ;
char x = 'a' ;

② SCOPE & LIFETIME OF VARIABLES ⟶

within the mentioned code block ⇒ lasts as long as code block.

③ TYPE CONVERSIONS ⟶

→ Implicit - Type conversions.
→ Explicit - Type conversions.

(A) IMPLICIT - TYPE CONVERSIONS :

* Done automatically by compiler * 2 conditions.
(i) Two variable types are compatible.
(ii) Destination type larger than source type

byte (1)
↓
bool ⟶ char ⟶ short ⟶ int ⟶ float ⟶ long ⟶ double
(1) (2) (2) (4) (4) (8) (8)

* Reverse order ⇒ incompatible.
(Done by Explicit type conversion)

(B) EXPLICIT - TYPE CONVERSIONS :

* Done manually

eg:

(i) int to byte
```
int i = 257
b = (byte) i ;
```

(ii) double to int
```
double d = 323.142;
i = (int) d ;
```

(iii) double to byte
```
double d = 323.46;
b = (byte) d ;
```

★ OPERATORS IN JAVA :—

OPERATORS

ARITHMETIC	UNARY	ASSIGNMENT	RELATIONAL	LOGICAL	TERNARY	BITWISE	SHIFT
+	-	=	==	&&	? : * ✓	||	<< *
-	! *	+=	!=	||		&	>> *
*	++	-=	>	!		^ * ✓	
/	--	*=	<	^ * ✓		!	
%	~ * ✓	/=	>=	bool.		~ * ✓	
		%=	<=				

⊙ * All important operators ⟶

① '~' BITWISE COMPLEMENT :

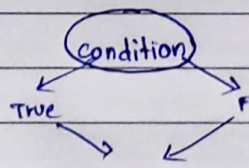
~(operand)

a = 5 ;

result = ~5 ; ⟶ OUTPUT ⇒ -6.

② '^' → XOR :

③ '?' : OPERATOR :



eg: max = (n1 > n2) ? n1 : n2 ;

④ '<<' LEFT SHIFT OPERATOR :——

* Shifts all of the bits in a value to the left specified number of times.

value << num.

⑤ '>>' RIGHT SHIFT OPERATOR :——

* Shifts all of the bits in a value to the right → specified no. of times.

value >> num

eg: shift.java.

```
class shift {
    public static void main (String[] args){
        int a = 5;
        System.out.println(a);
        a = a>>2;
        System.out.println(a);
        a = a<<2;
        System.out.println(a);
    }
}
```

OUTPUT

5
1
4.

⑥ UNSHIFTED RIGHT OPERATOR : >>>

a = a >>> 2 ;

no unshifted left.

★ TWO PARADIGMS :——

- Process-oriented.
- Object-oriented.

?: Ternary if-then-else

* ABSTRACTION.

* 3 OOPS principles
  - Encapsulation
  - Inheritance
  - Polymorphism.

* || & && → Shortcut operators.