✱ YouTube → Faisal Memon

Package=(Folder):-

A Package in Java is way to group class and interfaces like folders on your computer. It hepls to organize code, avoid name conflicts and control access.

✱ Types:

① Built-in Packages (Predefined) - like Java.util, Java.io, Java.Sql.

② User-defined Packages:- The ones we create ourselves.

Example → File 1:-

```
Package myPackage;
Public class MyClass {
  Public void Show() {
    System.Out.Println("Hello from
      Faisal Memon Class2 !");
  }
}
```

⊙ File-2:

```
import myPackage.MyClass;
Public class Test {
Public static void main(String[] args){
    MyClass obj = new MyClass();
        obj.show();
  }
}
```

✱ Main Points:

- ⊙ Package = Folder for Java classes
- ⊙ Helps in code Management
- ⊙ Avoids naming conflicts
- ⊙ Controls access

→ Rules of Variable Declaration.

- ⊙ Case-Sensitive (age ≠ Age)
- ⊙ Must begin with a letter, '_' or '$' (but prefer letters)

- Cannot start with a digit
- No spaces or special symbols (%, #, @, etc)
- Cannot be a Java keyword (e.g, int, class etc)
- Can contain letters, digits, '_' or '$' after the first character
- Use meaningful names (avoid x1, x2, a2)
- Use lowerCamelCase for normal variables (totalMarks)
- Use UPPER_CASE_WITH_UNDERSCORES for constants (MAX_SPEED)

```
Public static void main(String[] args) {
    // VALID Variable names
    int age = 25;
    int studentCount = 100;
    int _score = 95; // legal but not recommended
    int $total = 500; // legal
    int gearRatio = 6;
    final int MAX_SPEED = 120; // constant naming convention
}
```

```
// Printing Valid Variables
System.out.println("Age:" + age);
                   ("StudentCount:"+studentCount);
                   ("Gear Ratio:" + gearRation);
                   ("Max Speed: "+MAX_SPEED);
    }
}
```

✱ INVALID Variables names:

```
int 2fast = 50; // Cannot start with a digit
int Student-Count = 40; // hyphens not allowed
int total marks = 100; // speces not allowed
int for = 5; // 'for' is a keyword
int @rate = 10; '@' not allowed
int class = 1;  reserved keyword
```

✱ Example Showing Case-Sensisitivity

```
int number = 10;
int Number = 20;
```

Note: "Both are different Variables."

What Are Data Types

A data type defines what kind of data a Variable can Store.

Example
int age = 25; → age is an integer
String name = "Faisal"; → name is text

* Think of it as the "Shape" or "Category" of the data.

Types :→

① → Primitive Data Types: "These are the most types. directly stored in memory"

Type	Size	Example
byte	8-bit	byte b = 10;
Short	16-bit	Short s = 200;
int	32-bit	int age = 25;
long	64-bit	long views = 1_000_000L;
Float	32-bit (decimal)	Float pi = 3.14f;
double	64-bit (decimal)	double price = 99.99;
Char	16-bit Unicode	char grade = 'A';
boolean	1-bit (true/false)	boolean isActive = true;

Note:

A bit is the smallest unit of data in a computer. It can have only two possible values – 0(off) or 1(on).

Example

```
Public class DataTypeExample {
    Public static Void main (String[]args) {
        byte age = 25;
        Short marks = 32000;
        Int Salary = 50000;
        long distance = 12345678 9L;
        Float price = 45.6f;
        double pi = 3.14159;
        char grade = 'A';
        boolean isJavafun = true;
    System.out.println("Age: " +age);
    System.out.println("Java fun ? "+ isJavaFun);
    }
}
```

Output:

```
Age:25
Java fun ? True.
```

✡ Unicode:

Unicode is a Standard System that gives a unique number to every character used in any language - Hindi, English, Chinese, Arabic, emojis etc.

" It helps computers Understand and display text properly across all language.

Example

Character	Unicode Value
A	\u0041
a	\u0061
0	\u0030
@	\u0040
अ	\u0905
😊	(Emoji) \u1F60A

Example :

```
Public class UnicodeExample {
    Public static void main (String [] args) {
        char ch1 = '\u0041';
        char ch2 = '\u0905';
    System.out.println("Unicode \u0041 =" + ch1);
System.out.println("Unicode \u0905 =" + ch2);
    }
}
```

Output:- Unicode \u0041 = A
Unicode \u0905 = अ

(2) Non-Primitive Data Types:

These are created by the Programmer or Java itself.

✱ Types of Non-Primitive Data Types

(1) String → Sequence of characters (text)

Example:

```
String name = "faisal"
```

(2) Array → Collection of same data type Values

Example:

```
int [] numbers = {10, 20, 30, 40};
```

(3) Class → Blueprint or template for creating objects.

Example:

```
Class Car {
    String color;
    int speed;
}
```

④ Object → Instance of a class

Example:

```
Car myCar = new Car();
myCar.color = "Red";
myCar.speed = 120;
```

⑤ Interface: → Like a blueprint of a class (contains only method declarations)

Example

```
interface Animal {
    void sound();
}
```

✱ This is faisal bhai's order — Remember it.

① Primitive holds value directly

② Non-Primitive holds reference (address) of object.

✤ Control Flow

Control flow means the order in which statements, instructions, or function calls are executed in a program.

✤ 3 Main Types of Control Flow Statements:

Type	Description
① Decision Making	Choose one path from many
② Looping	Repeat a block of code
③ Jumping	Jump from one point to another

① Decision Making Statements:

ⓐ if statement.

```
if (age >= 18) {
    System.out.println("you can vote");
}
```

(b): if-else statement:

```
if (Marks >= 40) {
    System.out.println("Pass");
} else {
    System.out.println("Fail");
}
```

(c):- else-if ladder

```
if(marks >=90){
 System.out.println("Grade A");
} else if(mark >=75){
    system.out.println("Grade B");
}else{
  System.out.println("Grade C");
}
```

(d) → Switch statement:-

```
int day = 3;
switch (day) {
   case 1:
System.out.println("Monday"); break;
```

```
Case 2;
System.out.println("Tuesday"); break;
  Case 3;
System.out.println("Wednesday");
    break;
default:
System.out.println("Invalid day");
  }
```

(2):→ Looping Statements

(a) for →
```
for(int i=1; i<=5; i++) {
    System.out.println(i);
}
```

(b) While loop:
```
int i = 1;
while (i<=5) {
  System.out.println(i);
    i++;
}
```

(c): do-while loop

```
int i = 1;
do {
  System.out.println(i);
    i++;
} while (i<=5);
```

Note: "while checks first, do-while runs first."

(3) Jumping statements:-

(a) break: → Stops the loop or switch.

```
for (int i=1; i<=5; i++){
    if (i==3)
      break;
  System.out.println(i);
  }
```

(b) → Continue: → Skips the current iteration

```
for(int i=1; i<=5; i++){
  if (i==3) Continue;
System.out.println(i);
}
```

(c) **return:** Exits from a method

```
public int sum(int a, int b) {
    return a+b;
}
```

Additional loops

Enhanced for loop: –

Simplified loop to iterate all elements of array or collection without using index.

Example with Array:

```
int[] numbers = {10, 20, 30, 40};
for (int num : numbers) {
    System.out.println(num);
}
```

Output:

```
10
20
30
40
```

✱ What is a Method in Java?

A Method in Java is a block of statements that performs a specific task.

- It is also called function
- Method help in reusing code.

⊙ Advantages of Methods:

1. Organizes code
2. Reuses code
3. Makes program readable and maintainable.

⊙ Example of a Method:

```
Public class Demo {
  // Method to add two numbers
  Public Static int add (int a, int b) {
      return a+b;
  }
  Public Static void main (String [] args) {
      int Sum = add (10, 20); // method call
      System.Out.println ("Sum =" + sum);
Output= Sum = 30 }
```

✱ Types of Methods in Java:

① :- With return type & with parameters

```
int add (int a, int b){
    return a+b;
}
```

② With return type & without parameters

```
int getNumber(){return 100;}
```

③ Without return type & with parameters:

```
Void printSum(int a, int b){
System.out.println(a+b);
}
```

④ Without return type & without parameters

```
Void greet(){
System.out.println("Hello!");}
```

How is our Code Running!

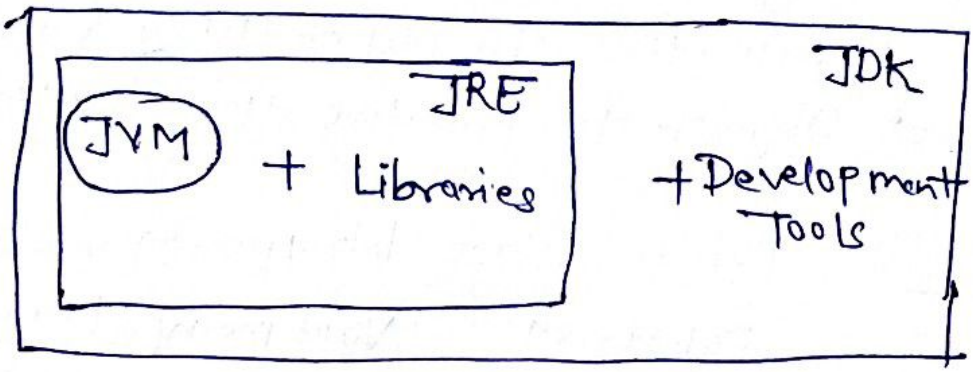
Source Code .Java → Compiler → Byte code .Class

Compilation

Native code ← JVM

Execution

JDK → Java Development Kit



⊙ Operators in Java:

Symbols that tell compiler to perform some operation

sum = a + b

operants — Operators

① Arithmetic Operators

Arithmetic operators are used to perform Mathematical calculation like addition subtraction, Multiplication, etc.

Types of Arithmetic Operators

Operator	Symbol	Example	Result
Addition	'+'	10+5	15
Subtraction	'—'	10−5	5
Multiplication	'*'	10*5	50
Division	'/'	10/5	2
Modules(Remainder)	'%'	10%3	1

Example

```
Public class ArithmeticExample {
    public static void main(String [] args) {
        int a=10, b=3;
        System.out.println("a+b=" +(a+b));
        — — — — ("a-b=" +(a-b));
        — — — — ("a*b=" +(a*b));
        — — — — ("a/b=" +(a/b));
        — — — — ("a%b=" +(a%b);
    }
}
```

Unary Operators:

Unary operators are operators that work on a single operand.

- They perform operations like increment decrement negation etc.

① Pre Increment:

Pre - Increment means Increase the value first, then use it.

⊙ Symbol : ++a.

Example:

```
Public class PreIncrementExample {
    Public static void main(String [] args) {
        int x = 10;
        System.out.println("Original x =" + x);
        int y = ++x; // x is increased first, then assigned to y
        System.out.println("After ++x : x = " + x);
        ——  ——  ——  (y);
    }
}
```

Output =

```
x = 10
x = 11
y = 11
```

Key Points:

1. Pre-increment updates the variable before using it.
2. Both the variable and the assigned value increase immediately.
3. Mostly used inside loops or expressions.

⇒ Relational Operators

Relational operators are used to compare two value.

. The Result is always a boolean value (true or false)

⊙ Types of Relational operators:

Operator	Meaning	Example	Result
==	Equal	5==5	true
!=	Not equal to	5!=3	true
>	Greater then	10>5	true
<	Less then	5<10	true
>=	Greater then or equal to	10>=10	true
<=	Less than or equal to	5<=5	true

Key Points:

① Result is always true or false.
② Used in condition (like if, while, for loops)
③ Work with numbers, characters

Logical Operators:

Logical operators are used to combine two or more conditions.
- They always return a boolean result (true and false)

Explanation:

① Logical AND (&&)

Returns true only if both conditions are true.

→ System.out.println(5>3 && 10>5) //true

② Logical OR (||):

- Returns true if at least one condition is true.

→ System.out.println(5>3 || 10<5); //true

③ Logical NOT (!) Reverses the result.

System.out.println(!(5>3)); //false

Key points:

1. Logical operators work only with boolean values.
2. They are mostly used in conditions and loops.
3. && and || in Java use short-circuit evaluation.
   - For && if the first condition is false then second is not checked.
   - For ||, if the first condition is ~~false~~ true then second is not checked.

Assignment operator:

Assignment operators are used to assign value to variables.

- The most common operator = (simple assignment).

Example `int a = 10; // assigns 10 to variable 'a'`

operator	Example	Meaning
=	a = 5	Assigns value 5 to a
+=	a += 5	Add 5 to a
-=	a -= 3	Subtracts 3 from a
*=	a *= 2	Multiplies a by 2
/=	a /= 2	Divides a by 2
%=	a %= 3	Stores remainder when a is divided by 3

Example : Program:

```
Public class AssignmentExample {
  Public static void main(String[] args) {
    int a = 10;
  System.out.println("Initial value of a:" +a);
     a += 5; // a = a+5
  System.out.println("after a +=5:" +a);
     a -= 3; // a-3
     System.out.println("After a -= 3:" +a);
     a *= 2; // a = a*2
     System.out.println("After a *=2:" +a);
  a /= 4; // a = a/4
     System.out.println("After a /= 4:" +a);
       a %= 3; // a = a%3;
     System.out.println("After a %= 3: +a);
          }
     }
```

Output:

```
Initial value of a = 10;
After a += 5:   15
After a -= 3:   12
After a *= 2:   24
After a /= 4:   6
After a %= 3:   0
```

Key Points

1. = assigns the value
2. Compound assignment (+=, -=, etc) is shorthand for arithmetic with assignment.
3. Works with all numeric types (int, float, etc).