### [**DAY-1-JAVA**](https://docs.google.com/document/d/1Jn4j43qxbsUATx4t1uwsALCnI9A9m-kSL2zw8uptYNw/edit?usp=sharing)**notes**

Let start the day-2-java

**Day-2-java**

[**Classification of operators in java**](#_8zb3h5t9w1lw) **1**

[**Ways to Collect Input Dynamically in Java**](#_a22ruc5o9mxu) **1**

[**Scanner Class in Java**](#_o0ggqw4o1ksg) **1**

[**Commonly Used Methods**](#_tcvmnnk6vjzt) **1**

[**How to Use the Scanner Class**](#_re3w4obt5en8) **2**

[**Control Statements in Java**](#_tn9fwvfz25a1) **2**

[**1. Conditional Statements**](#_d8ohwmel69al) **2**

[**2. Looping Statements**](#_j8cw9m6hqr4k) **3**

[**3. Jump Statements**](#_o3m7ry5ti8h3) **4**

### **Classification of operators in java**

* **Unary**: Works with 1 operand (e.g., ++, --, !).
* **Binary**: Works with 2 operands (e.g., +, >, &&).
* **Ternary**: Works with 3 operands (e.g., condition ? true : false).

### **Ways to Collect Input Dynamically in Java**

* **Scanner**: Most common for console input.
* **BufferedReader**: Efficient for large input.
* **Console**: Secure input for passwords.
* **Command-Line Arguments**: Pass input when running the program.

### **Scanner Class in Java**

The **Scanner** class is part of the java.util package and is used to take input from the user or other sources (like files). It provides various methods to read data of different types (e.g., int, float, string).

### **Commonly Used Methods**

| **Method** | **Description** |
| --- | --- |
| nextInt() | Reads an integer input. |
| nextFloat() | Reads a float input. |
| nextDouble() | Reads a double input. |
| nextBoolean() | Reads a boolean input (true/false). |
| nextLine() | Reads an entire line of text. |
| next() | Reads a single word (up to a space). |

### **How to Use the Scanner Class**

1. **Import the Class**:  
   Use import java.util.Scanner;.
2. **Create a Scanner Object**:  
   Example: Scanner scanner = new Scanner(System.in);
3. **Call Methods**:  
   Use appropriate methods like nextInt(), nextLine(), etc., to read user input.
4. **Close the Scanner**:  
   Always use scanner.close() after input operations to free resources.

### **Control Statements in Java**

Control statements determine the flow of execution in a Java program. They are grouped into three types:

### **1. Conditional Statements**

Used to execute code based on conditions.

* **if**: Executes a block if a condition is true.  
  if (x > 0) {

System.out.println("Positive number");

}

* **if-else**: Executes one block if true, another if false.  
  if (x > 0) {

System.out.println("Positive");

} else {

System.out.println("Negative");

}

* **if-else-if**: Tests multiple conditions.  
    
  if (x > 0) {

System.out.println("Positive");

} else if (x == 0) {

System.out.println("Zero");

} else {

System.out.println("Negative");

}

* **switch**: Selects code to execute based on a value.  
  switch (day) {

case 1: System.out.println("Monday"); break;

case 2: System.out.println("Tuesday"); break;

default: System.out.println("Invalid day");

}

### **2. Looping Statements**

Used to repeat a block of code.

**for**: Executes a block a fixed number of times.  
for (int i = 1; i <= 5; i++) {

System.out.println(i);

}

**while**: Repeats as long as a condition is true.  
int i = 1;

while (i <= 5) {

System.out.println(i);

i++;

}

**do-while**: Executes the block at least once before checking the condition.  
int i = 1;

do {

System.out.println(i);

i++;

} while (i <= 5);

### **3. Jump Statements**

Used to alter the normal flow of control.

* **break**: Exits a loop or switch.  
  for (int i = 0; i < 5; i++) {

if (i == 3) break;

System.out.println(i);

}

* **continue**: Skips the current iteration.  
  for (int i = 0; i < 5; i++) {

if (i == 2) continue;

System.out.println(i);

}

* **return**: Exits from a method and optionally returns a value.  
  public int add(int a, int b) {

return a + b;

}