Operators

- Operators are symbols that performs operations on variables or values.

  Categories:
  1. Arithmetic Operators (+, -, *, /, %)
  2. Relational Operators (==, !=, >,<,>=,<=)
  3. Logical Operators (&&, ||, !)

- Arithmetic Operators
  - Used for basic mathematical operations

```
public class ArithmeticExample {
    public static void main(String[] args) {
        int a=20,b=10;
        System.out.println("Addition: "+(a+b));
        System.out.println("Subtraction: "+(a-b));
        System.out.println("Multiplication: "+ (a*b));
        System.out.println("Division: "+ (a/b));
        System.out.println("Modulus: "+ (a%b));
    }
}
```

- Relational Operators
  - Used to compare values

```
public class RelationalExample {
    public static void main(String[] args) {
        int a = 10,b = 5;
        System.out.println("a == b: "+ (a==b));
        System.out.println("a != b: "+ (a!=b));
        System.out.println("a > b : "+ (a > b));
        System.out.println("a < b : "+ (a < b));
    }
}
```

- Logical Operators
  - Used to combine multiple conditions

```
public class LogicalExample {
    public static void main(String[] args) {
        int age = 20;
```

```
        boolean hasId = true;

        // AND Operator
        // T T -> True
        System.out.println(age >= 18 && hasId);

        // OR Operator
        // F F -> False
        System.out.println(age < 18 || hasId);

        // NOT Operator
        // !(T) -> False
        // !(F) -> True
        System.out.println(!(age >= 18));
    }
}
```

Control Statements(if statement, if-else statement, if-else-if ladder)

- Control statements allow us to control the flow of execution.

- if Statement:
  - It is used to execute a block of code if a condition evaluates to true.
    SYNTAX:
    if(CONDITION){
    // Code to execute if the condition is true
    }

```
public class IfExample {
    public static void main(String[] args) {
        int number = 10;

        // 10 > 0 -> True
        if(number > 0){
            System.out.println("The Number is positive");
        }
    }
}
```

- if-else statement
  - It provides an alternative path of execution when the condition is false

SYNTAX:

```
if(condition){
        // Code to execute if the condition is true
} else {
        // Code to execute if the condition is false
}
```

```java
public class IfExample {
    public static void main(String[] args) {
        int number = -10;

        // - 10 > 0 -> False
        if(number > 0){
            System.out.println("The Number is positive");
        } else {
            System.out.println("The Number is negative");
        }
    }
}
```

- if-else-if Ladder
  - It is used to test multiple conditions

SYNTAX:

```
if(condition1){
                // Code to execute if condition1 is true
}else if (condition2) {
                // Code to execute if condition2 is true
}else {
                // Code to execute if none of the conditions are true
}
```

```java
public class IfElseIfExample {
    public static void main(String[] args) {
        int marks = 70;

        if (marks >= 90){
            System.out.println("Grade: A+");
        } else if (marks >= 80) {
            System.out.println("Grade: A");
        } else if (marks >= 70) {
            System.out.println("Grade: B");
        } else if (marks >= 60) {
            System.out.println("Grade: C");
        } else {
            System.out.println("Please Attempt Again");
        }
    }
}
```

Switch Case Statements

- Are a control structure used to simplify the decision making when multiple options exists.

SYNTAX:

```
switch(expression){
                    case value1:
                      // code block
                      break;

                      case value2:
                      // Code block
                      Break;

                      // more cases

                      Default:
                      //Default block of code
}
```

```java
public class SwitchCaseExampleClass {
    public static void main(String[] args) {
        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;

                case 4:
                    System.out.println("Thursday");
                    break;
                default:
                    System.out.println("Invalid Input!!!!!!");
            }

    }
}
```

Explanation:

Day variable value is 3 and it matches with case 3, so Wednesday is printed.

The break prevents execution of further cases.

The default is optional but it executes when no other case matches.

Looping Statements

It is used to execute a block of code repeatedly.

Types of Loops:

1. For loop
2. While loop
3. Do-while loop

- For loop
  - It is used when the number of iterations are known.

SYNTAX:

```
for(initialization; condition; update){
        // CODE BLOCK
}
```

```java
public class ForLoopExample {
    public static void main(String[] args) {
        for (int i = 1; i <= 50; i++){
            System.out.println("Count: "+ i);
        }
    }
}
```

Explanation:

Initialization: int i =1;

Condition: i <= 50

Update: i++

While loop

- It is used when the condition is checked before each iteration

SYNTAX:

```
While(condition) {
        // code block
}
```

```java
public class WhileLoopExample {
    public static void main(String[] args) {
        int i = 1;
        while(i <= 50){
            System.out.println("Count: "+ i);
            i++;
        }
    }
}
```

while loop: when number of iteration we don't know then we use while loop

- Do-while loop
  - It guarantees the execution of the loop body at least once.

SYNTAX:

do {

// Code Block

} while(condition);

```
public class DoWhileLoopExample {
    public static void main(String[] args) {
        int i = 1;

        do {
            System.out.println("Count: "+ i);
            i++;
        } while(i <= 5);
    }
}
```

Note:

1. Use for loops when number of iterations are known
2. Use while loops when the condition needs to be checked before execution
3. Use do-while loops when we want the loop execute for at least once.
4. Use switch for multiple conditional branches.