JAVA OPERATORS

1. **Ternary Operator**

**Variable = Boolean expression? value to assign if true: value to assign if false**

**example:**

**int number1 = 40;**

**int number2 = 20;**

**int biggerNumber = (number1 > number2) ? number1 : number2;**

**//Compares both numbers and returns which one is bigger**

**Implicit Promotion (Widening)**

Implicit promotion occurs when the Java compiler automatically converts a smaller data type to a larger one to avoid data loss. This is also known as type widening. It happens without the programmer needing to do anything explicitly, hence it's "implicit." Java promotes data types in a specific order based on their size:

* byte → short → int → long → float → double

example: int a = 100; long b = a; // Implicit promotion from int to long

In the example above, a is of type int and is implicitly promoted to long because long can hold a larger range of values than int. No explicit cast is needed here.

Rules for Implicit Promotion:

* The new type must be capable of holding all the possible values of the original type.
* There is no loss of information during the conversion

--------------------------------------------------------------------------------------------

Explicit Promotion (Narrowing)

Explicit promotion, also known as **type narrowing**, occurs when you want to convert a larger data type to a smaller one. This conversion is not done automatically by the compiler because it can lead to **data loss** or **precision loss**. Therefore, you need to explicitly cast the data type.

example: double x = 9.78; int y = (int) x; // Explicit promotion from double to int (narrowing)

In the above example, the double x is explicitly cast to an int, which leads to the loss of the decimal part (9.78 becomes 9). Without the cast (int), this would result in a compilation error, as the compiler cannot automatically narrow the type.

**Rules for Explicit Promotion:**

* You need to explicitly tell the compiler to perform the conversion using a **cast**.
* Data loss or truncation can occur.

----------------------------------------------------------------------- ----------------------------------

JAVA CONDITIONAL STATEMENT

**Switch Statement**

The switch statement in Java is a control flow statement that allows you to execute different blocks of code based on the value of an expression. It is an alternative to using multiple if-else statements when dealing with a fixed set of possible values.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a chat

Description automatically generated

**Important Note:**

**Break Statement: If you forget the break statement, the code will continue executing the next case even if it doesn't match (known as "fall-through").**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a chat

Description automatically generated**

**Using default: Always include a default case to handle unexpected values.**

-------------------------------------------------------------------------------------------------------------

**Switch Expression:**

In Java, **switch expressions** are an enhancement to the traditional switch statement, introduced in **Java 12 (as a preview feature)** and finalized in **Java 14**. Switch expressions allow you to return a value directly from the switch and provide a more concise and flexible syntax.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A white background with black text

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Other Statements**

**If statement**

**If – else statements**

**If – else if – else statements**

JAVA LOOPS

**While Loop**

while loop is a control flow statement that allows code to be executed repeatedly based on a given condition. The loop continues to execute as long as the condition is evaluated as true. When the condition becomes false, the loop terminates.

A screenshot of a computer

Description automatically generated

Condition: This is a boolean expression evaluated before each iteration of the loop. If the condition is true, the loop body is executed. If it is false, the loop terminates.

Loop Body: The block of code inside the { } is repeatedly executed as long as the condition is true.

A screenshot of a phone

Description automatically generated

A screenshot of a computer code

Description automatically generated

**A screenshot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer code

Description automatically generated**

**Enhanced For Loop**

Also known as the "for-each" loop, it simplifies iterating over collections like arrays and other Iterable types (like ArrayList, HashSet, etc.). This loop is especially useful when you don't need to know the index of each item and just want to perform an action on each element.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated