Lesson Plan

Basics

PW SKILLS

List of Concepts Involved:

- Java Output
- Java Variables
- Java Identifiers
- Java Data Types
- Java Operators
- Java operators precedence

Topic 1: Java Output

Let us see the first programme in Java:
A 'Hello World in Java' programme is a simple programme that displays the 'Hello World in Java' on the screen.

Let us take a look at how the Java 'Hello World Java' programme works.

```
//  First Program
class HelloWorldJava {
    public static void main(String[] args) {
        System. out.println("Hello World Programme in
Java");
    }
}
```

Output

Hello World Programme in Java

How Does this Programme Work?

Compiler:- In computing, a compiler is a computer program that translates computer code written in one programming language into another language. The name "compiler" is primarily used for programs that translate source code from a high-level programming language to a lower level language to create an executable program.

1. // First Program

Any line in Java that begins with // is a comment. Comments are intended to help users reading the code understand the programme's intent and functionality. The Java compiler completely disregards it.

2. class HelloWorldJava

Every Java application starts with a class definition. The class in the programme is called HelloWorldJava, and its definition is as follows:

```
class HelloWorldJava {
…..
}
```

We have to keep in mind that every Java application has a class definition, and the class name should match the name of the file in Java.

3. Public static void main(String[] args) { ... }

This is the most common method. The main method is required in every Java application. All Java programs begin execution by calling main()
Let's understand the key terms:

a. Public: This is a visibility specifier or access specifier that defines the component's visibility. The term 'public' refers to a parameter or component that is visible to everyone.

b. Static: The keyword 'static' indicates that the method/ object/ variable that follows this keyword is static and can be invoked without the object or the dot (.) operator. The presence of the static keyword before the main method indicates that the main method is static.

c. Void: The keyword 'void' indicates that the method returns nothing.

d. Main: The keyword 'main' denotes the main method, which is the starting point for any Java programme. A Java programme's execution begins with the main method. The curly braces {} indicate start and end of main.

e. String[] args: The command line arguments are stored in the string array args.

4. System.out.println (IMPORTANT)

System. out. The println function is used to print messages to the screen. In
Java, the system is a class. The PrintStream class is represented by the parameters "Out" and "println." Println is a method, whereas "out" is an object.
The built-in method print() is used to display the string which is passed to it.
This output string is not followed by a newline, i.e. the next output will start on the same line.The built-in method println() is similar to print(), except that println() outputs a newline after each call.

Example Code:

```
public static void main(String[] args) {
     System.out.println("Hello World");
     System.out.println("Welcome to Physics Wallah");    }
```

Output:

Hello World
Welcome to Physics Wallah

Topic 2: Java Variables

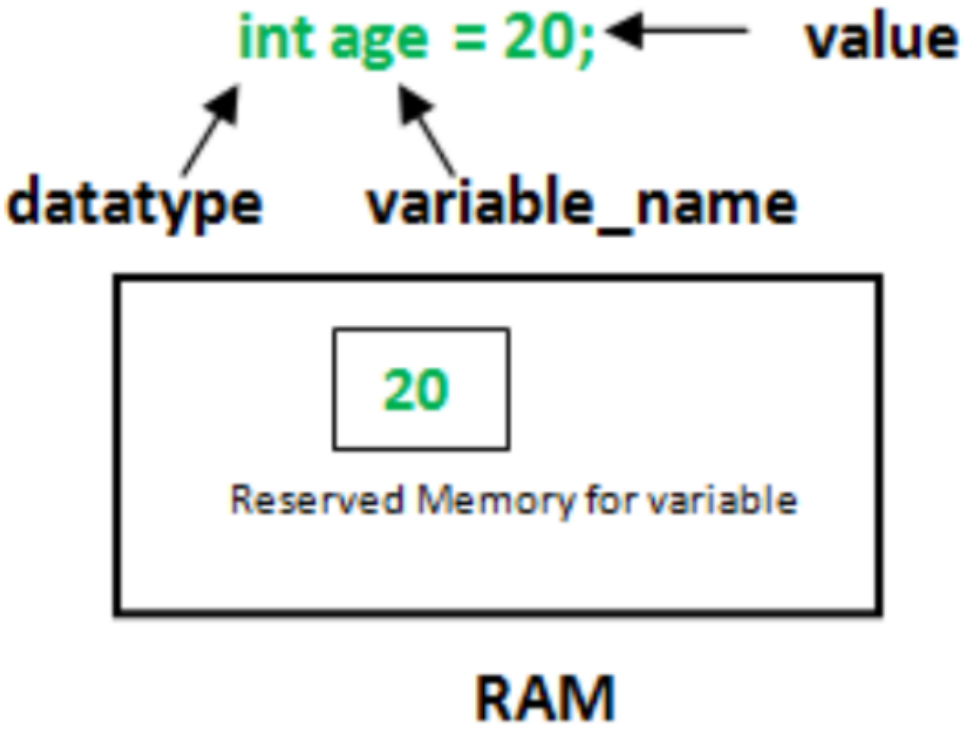
- A variable is the title of a reserved region allocated in memory. In other words, it may be a name of the memory location.
- Each variable should be given a unique name to indicate the storage area (identifier).
- A variable is a container which holds the value while the Java program is executed. A variable is assigned with a data type.

Syntax for Declaring a Variable:

Type variable_name [= value];
The variable_name is the name of a variable. We can initialize the variable by specifying an equal sign and a value (Initialization is optional).However, the compiler never assigns a default value to an uninitialized local variable in Java.

Creating Variables example:

```
int rate = 40;
```



Here, rate is an int data type variable with the value 40 assigned to it.
The variable can only hold integer values, as indicated by the int data type.
In the example, we assigned a value to the variable during the declaration process. However, it is not required. Variables can be declared and assigned separately. As an example,

```
int rate;
rate = 40;
```

Change Values of Variables

A variable's value can be changed in the programme. Hence, it is named variable. As an example,

```
int rate = 50;
System.out.println(rate);  // 50
rate = 60;
System.out.println(rate);   // 60
```

Initially the value of rate was 50 ,now it is changed to 60.

Java Naming Conventions

All Java components require names. In Java, there are several points to remember while naming the variable. They are as follows -

- Variable names should not begin with a number. As an example
  int 2var; // 2var is an invalid variable .

- Whitespace is not permitted in variable names. As an example,
  int cricket score; // invalid variables.
  There is a gap bw cricket and score.

- A Java keyword (reserved word) cannot be used as a variable name. For example int float; //invalid expression as float is defined as a keyword in java.

- Here, we should use variable names with more than one word with all lowercase letters for the first word and capitalization of the first letter of each subsequent word. Take, for example, cricketScore. This is called camelcase.

- When creating variables, It's preferable to give them meaningful names-age,earning,value for example, make more sense than variable names like a, e, and v.

- Use all lowercase letters when creating one-word variable names. It's preferable to use physics rather than PHYSICS or pHYSICS, for example.

Topic 3: Java Identifiers

Java identifier is a name given to a package, class, interface, method, or variable. All Java components require names. Names used for classes, variables, and methods are called identifiers.

In Java, there are several points to remember about identifiers. They are as follows -

- **Rule 1 -** All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
- **Rule 2 -** After the first character, identifiers can have any combination of characters.
- **Rule 3 -** A keyword cannot be used as an identifier.
- **Rule 4 -** The identifiers are case sensitive.
- **Rule 5 –** Whitespace is not permitted.
- Examples of legal identifiers: rank, $name, _rate, __2_mark.
- Examples of illegal identifiers: 102pqr, -name.

Topic 4: Java Data Types

Data types specify the different sizes and values that can be stored in the variable. Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.Therefore, by assigning different data types to variables, we can store integers, decimals, or characters in these variables.

There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes, Strings,Interfaces, and Arrays.

Primitive data types

1. **Boolean Type**

- The Boolean data type can have two values– true or false.
- They are typically used in true/false situations.

For example,

```
Boolean flag=true;
```

2. **Byte Type**

- Values for the byte data type range from -128 to 127 (8-bit signed two's complement integer).
- To save memory, it is used instead of int when it is certain that the value of a variable will be between -128 and 127.

For example,

```
byte range=105;
```

3. Short Type

- In Java, the short data type can have values ranging from -32768 to 32767 (16-bit signed two's complement integer).
- If the value of a variable is certain to be between -32768 and 32767, it is used instead of other integer data types (int, long).

For example,

```
short loss=-50;
```

4. Int Type

- Values for the int data type range from -2^31 to 2^31-1 (32-bit signed two's complement integer).
- You can use an unsigned 32-bit integer if you're using Java 8 or later. This will have a value of 0 as the minimum and a value of 232-1 as the maximum. For more information, see 'How to Use an Unsigned Integer in Java 8?'.

For example,

```
int profit=5000;
```

5. Long Type

- Values for the long data type range from -2^63 to 2^63-1 (64-bit signed two's complement integer).
- You can use an unsigned 64-bit integer with a minimum value of 0 and a maximum value of 264-1 if you're using Java 8 or later.

For example,

```
long profit=455559990;
```

6. Double Type

- The double data type is a 64-bit floating-point data type with double precision.
- It should never be used for exact values like currency.

For example,

```
double height=12.5;
```

7. Float Type

- The float data type is a 32-bit single-precision floating-point value. If you're curious, you can learn more about single-precision and double-precision floating-point.
- It should never be used for precise values like money.

For example,

```
float depth=-32.3f;
```

8. Char Type

- It's a Unicode 16-bit character.
- The char data type has a minimum value of 'u0000' (0) and a maximum value of 'uffff'.

For example,

```
char temp='a';
```

MCQs

1. Compiler assigns a default value to an uninitialized local variable in Java Programming.
   Whether this statement is true or false ?

   a) True
   b) False

Ans b) false

Explanation:

In java, it's compulsory to initialize any local variable before using it because compilers don't assign any default value to variables.

2. Which of the following data type can store the longest decimal number ?

Options:

a) boolean
b) double
c) float
d) long

Ans b) double

Explanation:

Out of all given options, only float and double can hold decimal numbers and double' is the longest data type with 64-bit defined by Java to store floating-point values.

3. Which of the following cannot be stored in character data type?

Options:

a) Special symbols
b) Letter
c) String
d) Digit

Ans c) String

Explanation:

String is a collection of characters and is stored in a variable of String data type.

Topic 5: Java Operators

Operators in Java can be classified into 6 types:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Unary Operators
6. Bitwise Operators

1.Java Arithmetic operators:

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The following table lists the arithmetic operators:
Assume integer variable num1 holds 20 and variable num2 holds 30, then:

Operator	Description and Example
+ Addition	**Adds values on either side of the operator. Example: num1 + num2 will give 50**
- Subtraction	**Subtracts right hand operand from left hand operand Example: num1 - num2 will give -10**

*** Multiplication**	**Multiplies values on either side of the operator Example: num1 * num2 will give 600**
/ Division	**Divides left hand operand by right hand operand Example: num1 / num2 will give 0.67**
% Modulus	**Divides left hand operand by right operand and returns remainder Example: num2 % num1 will give 10**
++ Increment	**Increases the value of operand by 1 Example: num2++ gives 31**
-- Decrement	**Decreases the value of operand by 1 Example: num1-- gives 19**

Example:

```
class Main {
  public static void main(String[] args) {

    // declare variables p and q
    int p = 20, q = 10;
    int result;

    // addition operator
    result=p+q;
    System.out.println(result);
    // subtraction operator
    System.out.println(p - q);
// we can directly perform subtraction in print statement,no
need to // use result variable here

    // multiplication operator
    System.out.println(p * q);

    // division operator
    System.out.println(p / q);

    // modulo operator
    System.out.println(p % q);
  }
}
```

Output

30
10
200
2
0

2.Java Relational Operators:

Operator	Description	Example
==	Is Equal To	4 == 5 returns false
!=	Not Equal To	1!= 5 returns true
>	Greater Than	1 > 5 returns false
<	Less Than	2 < 5 returns true
>=	Greater Than or Equal To	4 >= 5 returns false
<=	Less Than or Equal To	1 <= 5 returns true

```
class Main {
  public static void main(String[] args) {
      // create variables
    int p = 10, q = 15;
    // == operator
    System.out.println(p == q);  // false
    // ≠ operator
    System.out.println(p ≠ q);  // true
      // > operator
    System.out.println(p > q);  // false
     // < operator
    System.out.println(p < q);  // true
    // ⩾ operator
    System.out.println(p ⩾ q);  // false
   // ⩽ operator
    System.out.println(p ⩽ q);  // true
  }
}
```

Output

false
true
false
true
false
true

3.Java Logical Operators

Logical operators are used to check whether an expression is true or false. They are used in decision making.

Operator	Example	Meaning
&& (Logical AND)	expression1 && expression2	true only if both expression1 and expression2 are true
|| (Logical OR)	expression1 || expression2	true if either expression1 or expression2 is true
! (Logical NOT)	!expression	true if expression is false and vice versa

Example code:

```
class Main {
  public static void main(String[] args) {

    // && operator
    int p=15,q=10,r=5;
    System.out.println((p > q) && (p > r));  // true
    System.out.println((p > q) && (p < r));  // false

    // || operator
    System.out.println((r < q) || (p < q));  // true
    System.out.println((p > q) || (q > r));  // true
    System.out.println((p < q) || (p < r));  // false

    // ! operator
    System.out.println(!(p == q));  // true
    System.out.println(!(p > q));  // false
  }
}
```

Output

true
false
true
true
false
true
false

4.Java assignment operators:

The assignment operator = assigns the value of its right-hand operand to a variable, a property, or an indexer element given by its left-hand operand.
Let's see some more assignment operators available in Java.

Operator	Example	Equivalent to
=	p = q;	p = q;
+=	p += q;	p = p + q;
-=	p -= q;	p = p - q;
*=	p *= q;	p = p * q;
/=	p /= q;	p = p / q;
%=	p %= q;	p = p % q;

```
class Main {
  public static void main(String[] args) {

    int p = 10;
    int q;
```

```
    // assign value using =
    q = p;
    System.out.println(q);  // value of q is 10

    // assign value using =+
    q += p;
    System.out.println(q);  // value of q is 20

    // assign value using =*
    q *= p;
    System.out.println(q); // value of q is 200
  }
}
```

Output

10
20
200

5.Java Unary Operators:

Java unary operators are the types that need only one operand to perform any operation like increment, decrement, negation, etc

Operator	Meaning
+	**Unary plus:** not necessary to use since numbers are positive without 0-using it
-	**Unary minus:** inverts the sign of an expression
++	**Increment operator:** increments value by 1
--	**Decrement operator:** decrements value by 1
!	**Logical complement operator:** inverts the value of a boolean

For example:

```
class Main {
  public static void main(String[] args) {

        // initialize p
         int p = 5;
         System.out.println("Post-Increment Operator");
         System.out.println(p++); // 5
         // p's value is incremented to 6 after returning
         // current value i.e; 5

         // initialized to 0
         int q = 5;
         System.out.println("Pre-Increment Operator");
         System.out.println(++q);   //6
        // q is incremented to 6 and then it's value is
         // returned
  }
}
```

Output:

Post-Increment Operator
5
Pre-Increment Operator
6

6.Java Bitwise operators:

Operator	Description
~	Bitwise Complement
<<	Left Shift
>>	Right Shift
>>>	Unsigned Right Shift
&	Bitwise AND
^	Bitwise exclusive OR

Example code:

```
class Main {
  public static void main(String[] args) {

    // initialize p
        int p = 5;
        System.out.println(p<<2);
// Shifting the value of p towards the left two positions
        }
     }
```

Output: 20

Explanation:

Shifting the value of p towards the left two positions will make the leftmost 2 bits to be lost. The value of p is 5. The binary representation of 5 is 00000101.
After 2 left shifts , binary representation of p will be 00010100 which is equivalent to 20.

Topics 6:Java Operator Precedence and Associativity

Operator precedence determines the order in which the operators in an expression are evaluated.

Now, take a look at the statement below:

int ans = 10 - 4 * 2;

What will be the value of variable ans? Will it be (10 - 4)*2, that is, 12? Or it will be 10 - (4 * 2), that is, 2?

When two operators share a common operand, 4 in this case, the operator with the highest precedence is operated first.

In Java, the precedence of * is higher than that of -. Hence, the multiplication is performed before subtraction, and the value of variable ans will be 2.

Associativity of Operators in Java

If the precedence of all operators in an expression is the same? In that instance, the second quality associated with an operator, associativity, comes into existence.

Associativity specifies the order in which operators are executed, which can be left to right or right to left. For example, in the phrase p = q = r = 10, the assignment operator is used from right to left. It means that the value 10 is assigned to r, then r is assigned to q, and at last q is assigned to p. This phrase can be parenthesized as (p = (q = (r = 10)).

Operator Precedence in Java (Highest to Lowest)

Precedence order

Category	Operators	Associativity
Postfix	++ - -	Left to right
Unary	++ - -+ - ! ~	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >> >>>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left

Example 1.

Let say we have 4 variables of type int ; p,q,r,s
s = p-++r-++q;
is equivalent to

Answer: s = p-(++r)-(++q);

Explanation: The operator precedence of prefix ++ is higher than that of - subtraction operator.

Example 2. What does the following code fragment print?

```
System.out.println(4 + 2 + "pqr");
System.out.println("pqr" + 4 + 2);
```

Answer: 6pqr and pqr42, respectively.

Explanation: The + operator is left-to-right associative, whether it is string concatenation or addition.

Example 3. What is the result of the following code fragment? Explain.

```
boolean p = false;
boolean q = false;
boolean r = true;
System.out.println(p == q == r);
```

Answer: It prints true.

Explanation: The equality operator is left-to-right associative, so p == q evaluates to true and this result is compared to r, which yields true.