**Assignment Oops Fundamentals**

1. How to Create an Object in Java?

Ans :In Java, you create an object by first defining a class and then using the `new` keyword to instantiate that class.

Example:

class MyClass {

// Class body

}

public class Main {

public static void main(String[] args) {

MyClass obj = new MyClass(); // Creating an object of MyClass

}

}

2 . What is the use of the `new` keyword in Java?

Ans :The `new` keyword in Java is used to create new objects. It allocates memory for the new object on the heap and initializes the object with default or specified values.

Example:

MyClass obj = new MyClass();

3. What are the different types of variables in Java?

Ans :Java has three main types of variables:

1. Local Variables: Defined inside methods or blocks and can only be accessed within that method or block.

2. Instance Variables: Defined within a class but outside any method, and accessible by all methods of that class. Each instance has its own copy of these variables.

3. Static Variables: Defined within a class using the `static` keyword and shared across all instances of the class.

4. What is the difference between Instance Variables and Local Variables

Ans :- nstance Variables: Defined in the class, outside any method, and each object gets its own copy. They remain in memory as long as the object exists.

- Local Variables: Defined within methods, constructors, or blocks. Their scope is limited to the block they are declared in, and they are removed from memory when the method/block execution is completed.

5. In which area is memory allocated for instance variables and local variables?

Ans : Instance Variables: Memory is allocated in the heap when the object is created.

- Local Variables: Memory is allocated in the stack when the method is called, and it is freed once the method execution is completed.

6 . What is Method Overloading?

Ans : Method Overloading is a feature in Java that allows a class to have multiple methods with the same name but different parameter lists. It enables methods to perform similar operations but with different inputs.

Example:

class MathOperations {

// Method to add two integers

public int add(int a, int b) {

return a + b;

}

// Overloaded method to add three integers

public int add(int a, int b, int c) {

return a + b + c;

}

}

public class Main {

public static void main(String[] args) {

MathOperations math = new MathOperations();

System.out.println(math.add(5, 10)); // Calls first method

System.out.println(math.add(5, 10, 15)); // Calls overloaded method

}

}