Oops Fundamentals

1. How to create an object in java?

Ans → To create an object in Java, you need to follow these steps:

1. Define a class: The first step is to define a class that represents the object you want to create. A class is a blueprint for creating objects, and it defines the properties and behaviors of the object.

public class Person {

String name;

int age;

}

1. Instantiate the class: Once you have defined the class, you can create an instance of it by using the "new" keyword followed by the name of the class:

Person person1 = new Person();

This creates a new instance of the Person class and assigns it to the variable "person1".

1. Set the properties of the object: You can set the properties of the object by using the dot notation:

person1.name = "John";

person1.age = 25;

1. Use the object: Once you have created and initialized the object, you can use it in your program:

System.out.println("Name: " + person1.name);

System.out.println("Age: " + person1.age);

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

Ans → In Java, the "new" keyword is used to create an instance of a class or to allocate memory for an object. When you use the "new" keyword, you are telling the Java virtual machine to create a new object of the specified class and allocate memory for it.

For example, let's say you have a class called "Person" and you want to create a new object of that class. You would use the following code:

Person person1 = new Person();

Here, the "new" keyword is used to create a new instance of the "Person" class and allocate memory for it. The "person1" variable is then used to refer to this new object.

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

Ans → In Java, there are three types of variables: local variables, instance variables, and static variables.

1. Local Variables: Local variables are declared inside a method, constructor, or block and are accessible only within that scope. They do not have a default value and must be initialized before use. Once the block or method is exited, the local variable is destroyed.

public void exampleMethod() {

int x = 10; // local variable

}

1. Instance Variables: Instance variables are declared within a class but outside of any method, constructor, or block. They are also known as member variables or attributes of a class. Instance variables are created when an object is created and are accessible throughout the class. They have default values assigned based on their data types.

public class Person {

String name; // instance variable

int age; // instance variable

}

1. Static Variables: Static variables are declared with the "static" keyword and are also known as class variables. They are shared among all instances of a class and are stored in memory throughout the life of the program. They are accessed using the class name rather than an object reference.

public class Counter {

static int count = 0; // static variable

}

4. What is the difference between the Instance variable and the Local variable?

Ans → The main differences between instance variables and local variables in Java are as follows:

1. Scope: Instance variables are declared inside a class but outside of any method, constructor, or block and have class-level scope. They can be accessed throughout the class by any method, constructor, or block. Local variables, on the other hand, are declared inside a method, constructor, or block and have block-level scope. They can be accessed only within that block or method.
2. Lifetime: Instance variables are created when an object of a class is created and are destroyed when the object is destroyed. They have a lifetime that is tied to the object's lifetime. Local variables, on the other hand, are created when a method, constructor, or block is entered and are destroyed when the method, constructor, or block is exited. They have a shorter lifetime than instance variables.
3. Initialization: Instance variables are initialized with default values when an object is created, which can be overridden with explicit initialization. Local variables do not have default values and must be explicitly initialized before use.
4. Memory allocation: Instance variables are stored in memory for each object of the class that is created. Local variables are stored on the stack and are allocated memory when the method, constructor, or block is entered.

5. In which area memory is located for Instance Variable and Local variable?

Ans → In Java, instance variables are stored in the heap memory area, while local variables are stored in the stack memory area.

When an object is created, its instance variables are allocated memory on the heap. Each object has its own set of instance variables, which are separate from those of other objects of the same class.

Local variables, on the other hand, are created when a method or block is entered and are stored on the stack. The stack is a special region of memory that is used for temporary storage of data during the execution of a method or block. When the method or block is exited, the local variables are destroyed and their memory is reclaimed.

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 parameters. When a method is overloaded, it has the same name as another method in the class, but it takes a different number or type of parameters.

In Java, method overloading is achieved by defining multiple methods with the same name but different parameter lists. The parameter lists must differ in either the number or type of parameters, or both.