**Oops Fundamental**

1. How to create an object in java?

**1.First you declare a class**

A class is like a blueprint that specifies the object's properties (fields) and behaviors (methods).

1. **Instantiate the Object:**

Once you have the class defined, you can create an object (an instance of that class) using the new keyword followed by the class name and a set of parentheses.

Example : suppose we have a Test class

Then its object is

**Test t1=new Test();**

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

The new keyword in Java is used to create an instance (an object) of a class. When you use new, you are allocating memory for an object and calling a constructor to initialize that object.

**Memory Allocation:** When you use new, Java allocates memory to store the object's data (fields) and sets up any necessary resources. This memory allocation is typically done on the heap, which is where objects are usually stored in Java.

**Constructor Invocation:** After allocating memory, the new keyword invokes a constructor to initialize the object. Constructors are special methods in a class that are used to set initial values for the object's fields or perform any necessary setup actions.

**3.Reference Assignment:** The result of using new is a reference to the newly created object. This reference is assigned to a variable, allowing you to access and manipulate the object through that variable.

Ex: **Test t1=new Test();**

1. What are the different types of variables in java?

In Java, variables are classified into different types based on their scope, lifetime, and usage.

**Local Variables:**

* Defined within a method, constructor, or a block.
* Limited to the scope where they are declared.
* Must be initialized before use.

Example: public void exampleMethod() {

int localVar = 10; // Example of a local variable

}

**Instance Variables (Non-Static Fields):**

* Belong to an instance of a class (object).
* Each object has its own set of instance variables.
* Initialized with default values when an object is created.

**Example :-**public class MyClass {

int instanceVar; // Example of an instance variable

}

**Static Variables (Class Variables):**

* Belong to the class rather than any specific instance.
* Shared among all instances of the class.
* Initialized once (at class initialization) and retains its value across instances.
* Example:

public class MyClass {

static int staticVar; // Example of a static variable

}

1. What is the difference between instance and local variable in java?

| **Instance Variable** | **Local Variable** |
| --- | --- |
| They are defined in class but outside the body of methods. | They are defined as a type of variable declared within programming blocks or subroutines. |
| These variables are created when an object is instantiated and are accessible to all constructors, methods, or blocks in class. | These variables are created when a block, method or constructor is started and the variable will be destroyed once it exits the block, method, or constructor. |
| These variables are destroyed when the object is destroyed. | These variables are destroyed when the constructor or method is exited. |
| It can be accessed throughout the class. | Its access is limited to the method in which it is declared. |
| They are used to reserving memory for data that the class needs and that too for the lifetime of the object. | They are used to decreasing dependencies between components I.e., the complexity of code is decreased. |
| These variables are given a default value if it is not assigned by code. | These variables do not always have some value, so there must be a value assigned by code. |
| It is not compulsory to initialize instance variables before use. | It is important to initialize local variables before use. |
| It includes access modifiers such as private, public, protected, etc. | It does not include any access modifiers such as private, public, protected, etc. |

1. In which area memory is allocated for instance variable and a local variable?

**For instance variable**

A **heap** is a memory place where the objects and its instance variable are stored.

**For local variable**

A **stack** is a memory place where the methods and the local variables are stored.

1. What is method overloading?

If a [class](https://www.javatpoint.com/object-and-class-in-java) has multiple methods having the same name but different in parameters, it is known as **Method Overloading**.

If we have to perform only one operation, having the same name of the methods increases the readability of the [program](https://www.javatpoint.com/java-programs).

There are two ways to overload the method in java

1. By changing number of arguments
2. By changing the data type

Example: 1)

class Adder{

static int add(int a,int b)

{

return a+b;

}

static int add(int a,int b,int c)

{

return a+b+c;

}

}

class TestOverloading1{

public static void main(String[] args)

{

System.out.println(Adder.add(10,11));

System.out.println(Adder.add(10,31,41));

}

}

Example 2)

class Adder{

static int add(int a, int b)

{

return a+b;

}

static double add(double a, double b)

{return a+b;

}

}

class TestOverloading2{

public static void main(String[] args)

{

System.out.println(Adder.add(11,11));

System.out.println(Adder.add(12.3,12.6));

}

}