OOPs Fundamentals

Assignment Questions

Q 1. How to create an object in Java?

Ans = The object is a basic building block of an OOP language. In Java, we cannot execute any program without creating an object. There are various ways to create an object in Java that we will discuss in this section, and also learn how to create an object in Java.
java provides five ways to create an object.

- Using new Keyword
- Using clone() method
- Using the newInstance() method of the Class
- Using the newInstance() method of the Constructor class
- Using Deserialization

Method 1;

Using a new keyboard:- it is the most common and general way to create obj in Java.

For example:- Test t=new Test();

Method 2;

Using clone():- clone() method is present in the obj class. It creates and returns a copy of the obj.

For example:- Test t1=new Test();
Test t2=(Test)t1.clone();

Method 3;

Using the newInstance() method of the class:- The newInstance() method of the Class class is also used to create an object. It calls the default constructor to create the object. It returns a newly created instance of the class represented by the object.

For example:- ClassName object = ClassName.class.newInstance();

Method 4;

Using the newInstance() method of the Constructor class:- It is similar to the newInstance() method of the ClassIt is known as a reflective way to create objects. The method is defined in the Constructor class which is the class of java. lang. reflect package. We can also call the parameterized constructor and private constructor by using the newInstance() method.

For example:- Constructor<Employee> constructor = Employee.class.getConstructor();

Employee emp3 = constructor.newInstance();

Method 5;

Using Deserialization:- Deserialization is a technique of reading an object from the saved state in a file.

For example:- public final Object readObject() throws IOException, ClassNotFoundException

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

Ans = The "new" keyword in Java is used to create an instance of an object. It allocates memory to an object and returns a reference to the object created. It is used with a constructor to create an object**.**

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

Ans= There are three different types of variables in Java, they are follows:

- Local Variables.
- Instance Variables.
- Static Variables.

Q 4. What is the difference between Instance variables and local variables?

Ans=

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 reserve memory for data that the class needs and that too for the lifetime of the object.	They are used to decrease 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.

Q 5. In which area memory is allocated for instance variable and a local variable?

Ans= Instance variables are allocated in the heap memory and local variables are allocated in the stack memory.

Q 6. What is the method of overloading?

Ans= In Java, method overriding occurs when a subclass (child class) has the same method as the parent class.