**Q1. How to create Object in Java?**

Answer: An object in Java is an instance of a class. To create an object, you need to use the new operator followed by a call to the constructor of the class.

Here's an example to illustrate the creation of an object in Java:

public class CreateObjectExample1

{

void show()

{

System.out.println("Welcome to javaTpoint");

}

public static void main(String[] args)

{

//creating an object using new keyword

CreateObjectExample1 obj = new CreateObjectExample1();

//invoking method using the object

obj.show(); } }

**Q2. What is the use of a new keyword in Java?**

Answer: The new keyword in Java is used to allocate memory for an object and to call its constructor. The new keyword creates a new instance of a class and returns a reference to the object it creates.

**Q3. What are the different types of variables in Java?**

Answer: In Java, there are three types of variables:

Instance Variables: Instance variables are declared in a class, outside of any method, with the private access modifier. These variables are associated with each instance of the class, and their values are different for each object of the class. Instance variables are created when an object of the class is created and destroyed when the object is garbage collected.

Static Variables: Static variables are declared with the static keyword and are shared among all instances of the class. The value of a static variable is the same for all instances of the class, and it is stored in the class itself, not in any individual object. Static variables are created when the class is loaded into memory and destroyed when the class is unloaded.

Local Variables: Local variables are declared within a method or a block and are only accessible within that method or block. They are created when the method is called and destroyed when the method returns. Local variables do not have a default value and must be explicitly initialized before they can be used.

**Q4. What is the difference between Instance variable and Local Variable?**

Answer:

Instance variables and local variables are two different types of variables in Java that are used to store data in a program. The key difference between them is their scope and lifetime.

Instance variables are declared within a class, but outside of any method. They are associated with an instance of the class and belong to each individual instance of the class. The value of an instance variable can be accessed and modified by any method of the class, and it remains in memory as long as the instance of the class remains in memory.

Local variables, on the other hand, are declared within a method and only have meaning within the context of the method in which they are declared. They are not associated with an instance of the class and are only accessible within the method in which they are declared. The value of a local variable is destroyed when the method in which it was declared returns.

**Q5. In which area memory is allocated for instance variable and Local variable?**

Answer: Instance variables are stored in the heap memory, which is a portion of memory that is used for dynamic allocation of objects. When an object is created, memory for its instance variables is allocated in the heap, and a reference to that memory is stored in a variable.

Local variables, on the other hand, are stored in the stack memory. The stack memory is used for temporary storage of method call information and local variables. When a method is called, a new block of memory is created on the top of the stack for the method's local variables. When the method returns, the memory for its local variables is automatically freed and discarded.

**Q6. What is the method Overloading?**

Answer: Method overloading in Java is a feature that allows a class to have multiple methods with the same name, but with different parameters. This allows a class to provide multiple implementations for the same method, each with its own set of parameters and return type.

For example, consider a class Calculator that provides methods to perform addition, subtraction, and multiplication operations. If we want to provide two different implementations of the addition operation, one for adding two integers and one for adding two doubles, we can achieve this by overloading the add method:

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public double add(double a, double b) {

return a + b;

}

}

In this example, we have two methods with the same name add, but with different parameters. The first method takes two int parameters and returns an int result, while the second method takes two double parameters and returns a double result.

When the method is called, the Java compiler selects the correct implementation of the method based on the types of the parameters passed in the method call. If the parameters match the parameters of one of the overloaded methods, that method is called, otherwise, a compile-time error is generated.