Oops fundamentals

1. How to Create an Object in Java?

Answer: We can create Objects in Java in various ways:

1. Using a new keyword.
2. Using the newInstance() method of the Class class.
3. Using the newInstance() method of the Constructor class.
4. Using Object Serialization and Deserialization.
5. Using the clone() method.

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

Answer: “new” keyword is used to create an object in java.

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

Answer:

Classification of variable based on type of value represented:-

1. Primitive variable: It is used to represent primitive datatypes.

Example: int x = 5;

1. Reference variable: It is used to refer objects.

Example: Student rohan = new Student();

Classification of variable based on behaviour and position of declaration:-

1. Local variable: A variable is defined within a method or block or

constructor is called local variable.

1. Instance variable: The variables that are declared inside the class but outside the scope of any method are called instance variables in Java.
2. Static variable: Static variables are declared inside a class but outside of a method starting with a keyword static.

Example:

public class ExampleClass {

    // Static variable

    static int staticVariable = 10;

    // Instance variable

    int instanceVariable = 5;

    public void exampleMethod() {

        // Local variable

        int localVariable = 3;

        System.out.println("Static variable: " + staticVariable);

        System.out.println("Instance variable: " + instanceVariable);

        System.out.println("Local variable: " + localVariable);

    }

    public static void main(String[] args) {

        ExampleClass obj = new ExampleClass();

        obj.exampleMethod();

    }

}

Top of Form

Top of Form

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

Answer:

|  |  |
| --- | --- |
| 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. |
| 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. |

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

Answer: In stack memory.

6. What is method overloading?

Answer: Method Overloading allows different methods to have the same name, but different signatures where the signature can differ by the number of input parameters or type of input parameters, or a mixture of both.

Example:

public class methodOverloading {

    // Overloaded sum(). This sum takes two int parameters

    public int sum(int x, int y) { return (x + y); }

    // Overloaded sum(). This sum takes three int parameters

    public int sum(int x, int y, int z)

    {

        return (x + y + z);

    }

    // Overloaded sum(). This sum takes two double

    // parameters

    public double sum(double x, double y)

    {

        return (x + y);

    }

    // Driver code

    public static void main(String args[])

    {

        methodOverloading s = new methodOverloading();

        System.out.println(s.sum(10, 20));

        System.out.println(s.sum(10, 20, 30));

        System.out.println(s.sum(10.5, 20.5));

    }

}