1. How to create an object in java?

Ans: - In Java, objects are instances of classes and are created using the "new" keyword.
syntax:

- ClassName objectName = new ClassName();

where "ClassName" is the name of the class and "objectName" is the name you give to the created object.

The "new" keyword is followed by the constructor of the class, which is a special method used to initialize the object.

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

Ans: - The "new" keyword in Java is used to create objects or instances of a class. It's also used to allocate memory for these objects. The "new" keyword creates an instance of a class by calling the class constructor, which initializes the object's properties.
syntax:

- ClassName objectName = new ClassName();

where "ClassName" is the name of the class and "objectName" is the reference variable to the newly created object.

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

Ans: - There are three types of variables in Java:

- Instance variables (non-static fields): - Instance variables are declared within a class but outside of any method, constructor or block. They belong to each object of the class and have different values for each object.
- Static variables (static fields): - Static variables are also declared within a class but outside of any method, constructor or block, but have the "static" keyword. They belong to the class, not objects of the class, and there is only one copy of a static variable shared among all objects of the class.
- Local variables: - Local variables are declared within a method, constructor or block and are only accessible within the scope of the method, constructor or block where they are declared.

4. What is the difference between instance variable and local variables?

Ans: - Instance variables and local variables are different in the following ways:

NAME :- RISHABH AGRAWAL
MBNO:- 9420155354
EMAIL:- agrawaltraders.rishabh@gmail.com

- Scope: Local variables are only accessible within the method, constructor, or block where they are declared and have limited scope. Instance variables are accessible from anywhere within the class and have a wider scope.
- Initialization: Local variables must be initialized before use, otherwise, the compiler will show an error. Instance variables are automatically assigned a default value by the JVM when an object is created.
- Memory allocation: Local variables are stored on the stack and are allocated memory when the method, constructor, or block they are declared in is called. Instance variables are stored in the heap and are allocated memory when an object is created.
- Lifetime: The lifetime of a local variable ends when the method, constructor, or block it was declared in ends. The lifetime of an instance variable ends when the object it belongs to is garbage collected.

In short, local variables are used for temporary storage within a method or block, while instance variables are used for permanent storage that lasts as long as the object they belong to.

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

Ans: - In Java, memory for instance variables is allocated in the heap while memory for local variables is allocated on the stack.

The heap is a portion of memory where all objects and their associated instance variables are stored. When an object is created using the "new" keyword, memory for its instance variables is allocated in the heap.

The stack is a portion of memory where all method calls, local variables, and intermediate results are stored. When a method, constructor, or block is executed, memory for its local variables is allocated on the stack. When the method, constructor, or block execution is completed, the memory for its local variables is automatically deallocated from the stack.

6. What is method overloading?

Ans: - Method overloading in Java is the practice of creating multiple methods with the same name within the same class, but with different parameters. The compiler distinguishes between these methods based on the number and/or type of parameters.

Method overloading allows you to have multiple methods with the same name, but different behaviors, based on the input. It makes your code more readable and flexible, as it eliminates the need to have different names for similar methods with only slight variations in behavior.

For example:

```
public class Example {
```

NAME :- RISHABH AGRAWAL
MBNO:- 9420155354
EMAIL:- agrawaltraders.rishabh@gmail.com

```
    public int add(int a, int b) {
        return a + b;
    }

    public double add(double a, double b) {
        return a + b;
    }
}
```

Here, we have two methods named "add" with different parameter lists. The first method accepts two integer parameters and returns the sum of them, while the second method accepts two double parameters and returns the sum of them.

NAME :- RISHABH AGRAWAL
MBNO:- 9420155354
EMAIL:- agrawaltraders.rishabh@gmail.com