Oops Fundamentals

Assignment Questions

Assignment Questions

1. **How to Create an Object in Java ?**

   Ans :

   To create object in java we have to use new keyword.
   For example, the following code creates an object of the Calc class :

   ```
   Calc obj = new Calc();
   ```

   The new keyword allocates memory for the object and initializes the object's instance variables.

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

   Ans :

- **In Java, the new keyword is used to create an object of a class. When the new keyword is used, memory is allocated for the object and a reference to that memory location is returned.**

- **This reference can then be used to access the methods and variables of the object. For example, if you have a class Person, you can create a new object of that class using the new keyword like this:**

  ```
  Person person = new Person();
  ```

- **This creates a new instance of the Person class and assigns it to the variable person.**

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

Ans :

- **Local Variables: Local variables are declared within a method, constructor, or a block of code. They are only accessible within the scope where they are declared. Local variables must be initialized before they are used.**

- **Instance Variables: Instance variables, also known as non-static variables, are declared within a class but outside any method. Each instance of the class has its own copy of instance variables. Instance variables are initialized with default values if not explicitly assigned.**

- **Static Variables: Static variables, also known as class variables, are declared using the static keyword. They belong to the class rather than any specific instance of the class. Static variables are shared among all instances of the class. They are initialized with default values if not explicitly assigned.**

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

Ans :

- **The main difference between instance variables and local variables is that instance variables are associated with individual objects, while local variables are associated with the method or constructor in which they are declared.**

- **Instance variables are also created when an object is created, and they are destroyed when the object is destroyed.**

- **Local variables are created when a method or constructor is invoked, and they are destroyed when the method or constructor returns.**

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

Ans :

a. Instance Variables: Memory for instance variables is allocated on the heap. The heap is the runtime data area where objects and their instance variables

are stored. When an object is created using the new keyword, memory is dynamically allocated on the heap to hold the instance variables of that object.

- **Each instance of a class has its own copy of instance variables, and they exist as long as the object exists. Instance variables are associated with the object itself and can be accessed by any method or constructor within the class.**

b. Local Variables: Memory for local variables is allocated on the stack. The stack is a region of memory used for method calls and local variable storage. When a method or block of code is entered, memory is allocated on the stack to store the local variables declared within that method or block.

- **Local variables have block-level scope and are temporary in nature. They are created when the block is entered and destroyed when the block is exited. The memory allocated for local variables is automatically reclaimed by the JVM when it determines that the variable is no longer in use.**

6. What is method overloading ?

Ans :

- **Method overloading is a feature of Java that allows you to have multiple methods with the same name, but with different parameters. This can be useful for when you have methods that perform similar tasks, but with different data types or numbers of parameters.**

- **method called add() that adds two numbers. It could overload this method to also add three numbers, or to add two strings.**

```
public class Overloading {

public void add(int a, int b) {
   System.out.println(a + b);
}

public void add(int a, int b, int c) {
   System.out.println(a + b + c);
```

```
    }

    public void add(String a, String b) {
        System.out.println(a + b);
    }

    public static void main(String[] args) {
        Overloading overloading = new Overloading();
        overloading.add(3,4);
        overloading.add(3,4, 1);
        overloading.add("rahul", "varshab");
    }
}
```