OOPS Fundamentals

Assignment

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

   Ans: <u>To create an Object in Java:</u>

   An **Object** is an instance of a class which has an identity, a behavior and a state. An Object is created by the use of 'new' keyword. There are the three steps when creating an object from class:

   - **Declaration:** A variable declaration with a variable name with an object type.
   - **Instantiation:** The 'new' keyword is used to create the object.
   - **Initialization:** The 'new' keyword is followed by a call to a class/constructor. This call initializes the new object.

   Example:

```
public class Test {
  public static void main (String args[]){
    Test tb = new Test ();
  }
}
```

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

   Ans: 'new' keyword is used to create an instance of the class. It initiates by allocating memory for a new object and returning a reference to that memory.

   Syntax:

   Table obj=new Table();

   Uses of new keyword:

   - It is used to create the object.
   - It allocates the memory at runtime.
   - All objects occupy memory in the heap area.
   - It invokes the object constructor.
   - It requires a single, postfix argument to call the constructor

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

   Ans: There are 3 types of variables in java:

   1. **Local variable -**
      - A variable declared inside the body of the method is called a local **variable.**
      - We can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
      - The scope of these variables exists only within the block in which the variables are declared.
      - A local variable cannot be defined with the "static" keyword.

2. **Instance variable –**
   - A variable declared inside the class but outside the body of the method, is called an **instance variable.**
   - Initialization of an instance variable is not mandatory.
   - Its default value is 0. Instance variables can be accessed only by creating objects.
3. **Static variable –**
   - A variable that is declared as a static keyword is called a **static variable**.
   - Memory allocation for static variables happens only once when the class is loaded in the memory.
   - This variable is initialized only once, just when the program execution starts.
   - It is the variable that should be initialized first, especially before an instance variable is initialized.

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

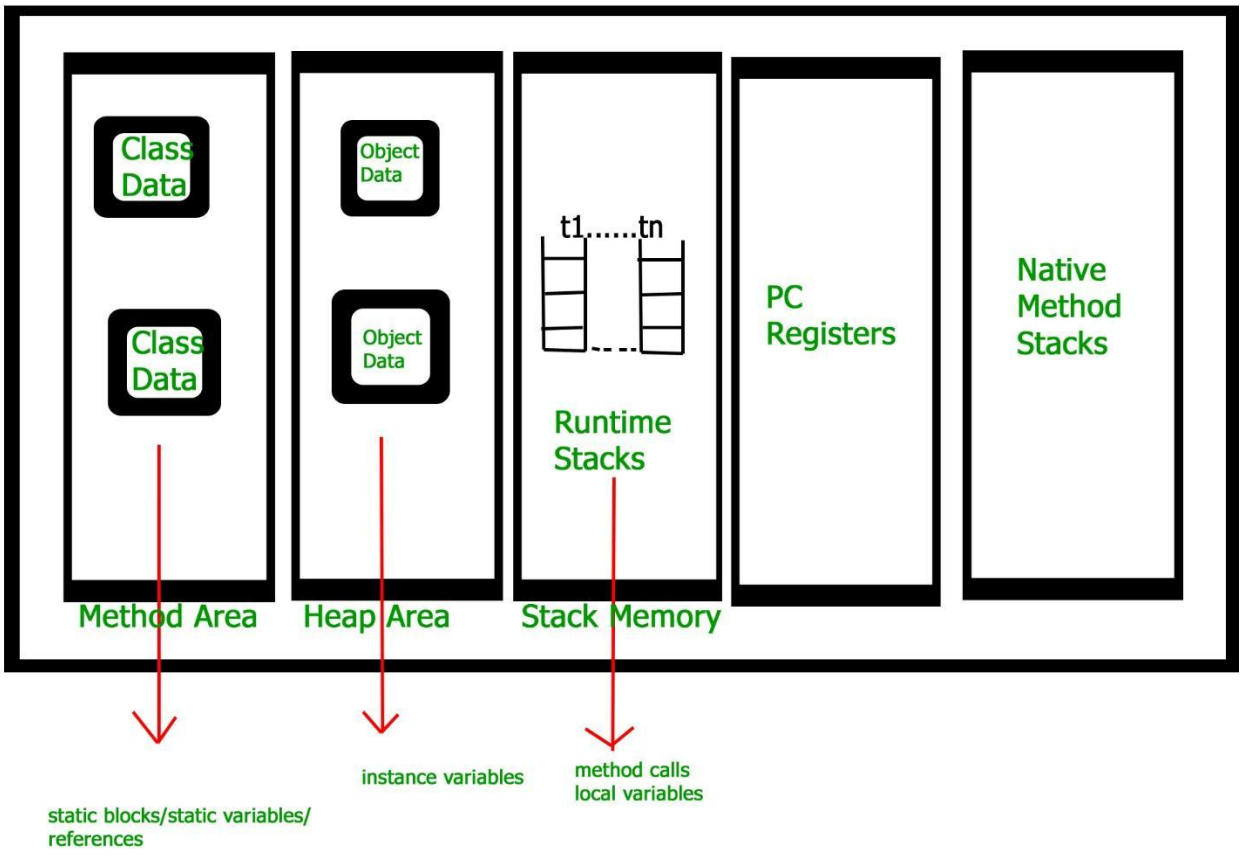
Ans:

	Instance Variables	**Local Variables**
1.	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.
2.	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.
3.	These variables are destroyed when the object is destroyed.	These variables are destroyed when the constructor or method is exited.
4.	It can be accessed throughout the class.	Its access is limited to the method in which it is declared.
5.	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.
6.	It is not compulsory to initialize instance variables before use.	It is important to initialize local variables before use.
7.	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?**

Ans:

- **Instance variables are** allocated in the heap area in JVM.
- **Local variable** is allocating in the stack memory of the JVM



6. **What is method overloading?**

Ans: Method overloading is in which multiple methods can have the same name with different parameters.

In other words, In Java, two or more methods may have the same name if they differ in parameters (different number of parameters, different types of parameters, or both). These methods are called overloaded methods and this feature is called method overloading.

Example :

```
class MethodOverloading {
    public int add (int a, int b)
    {
    int res= a+b;
    return res;
    }

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

```
    public static void main(String[] args) {
        MethodOverloading mo = new MethodOverloading();
         mo.add(1,4);
         mo.add(1,4,1);
    }
```

Output:

```
5
6
```