1.How to create an object in java ?
ANSWER:-
To create an object in java,first you need to define a class,and then use the “new” operator to instantiate an object of that class.Here is a simple example:-

```
class car{
   //fields,constructor,and methods etc
}
//creating object for the class car
car  obj_name=new car();
```

The operator creates a new instance of the class ,and the reference to this instance is sorted in the variable “obj_name”.

2.What is the new keyword in java ?
ANSWER:-
In java “new” keyword is used to create the instance of a class,which allocates the memory for the object and returns a reference to it.
Syntax:-
classname objname=new classname();
Example:-

```
class car{
   //fields,constructor,and methods etc
}
//creating object for the class car
car  obj_name=new car();
```

The ‘new’ operator creates a new instance of the class car, and allocates memory for it and returns a reference to the newly created object,which is stored in the new variable “obj_name”.

3.What are the different types of the variables in java ?
ANSWER:-
Division 1:-
Based on the type of value represented by a variable all variables are divided into two types. They are:-
1. Primitive variables
2.Reference variables

Primitive variables:-
Primitive variables are used to store the primitive values.
Example int x=5;

Reference variables:-
Reference variables can be used to refer the object.
Example:- Student s=new Student();

Division 2:-
Based on the behaviour and position of declaration all variables are divided into three types. They are:-
1.Instance variables
2.Static variables
3.Local variables

Instance variables:-
1.If the value of the variable changes from object to object such variables are known as instance variables.
2.For every object a separate copy of instance variables are created.
3.Instance variable will be created at the time of the object creation and destroyed at the time of object destruction hence the scope the instance variable is as same as the scope of the object.
4.Instance variables will store on the heap as the part of the object.
5.Instance variables must declared within a class directly but out side of any method or bock or constructor.
6.Instance variables can be accessed directly from an instance area .But cannot from a static area.
7.But by using object reference we can access instance variables from static area.
Example code:-

```
class Test
{
int i =10;
public static void main(String[] args) {
System.out.println(i);//CE: non static variable can't be referenced
Test t = new Test();
System.out.println(t.i);//valid
t.m1();
}
public void m1()
{
System.out.println(i);//valid
}
}
```

Local variables:-
1.Some time to meet temporary requirement of the programmer we declare variables inside a method or block or constructor such variables are called automatic variables or stack variables or temporary variables .
2.Local variables will store inside the stack.
3.The scope of the local variables is as same as the scope of the block in which it was declared.

Example code:-

```
class Test
{
public static void main(String[] args) {
int i =0;
for (int j =0;j<=3 ;j++ )
{
i = i<j;
}
System.out.println(j);//CE
}
}
```

Static variable:-

In java,a static variable is a class-level variable that is shared among all objects in the class. Unlike instance variables ,there is only copy of a static variable ,regardless of how many objects of the class are created .The static variable is declared using 'static' keyword and can be accessed using the class name ,without need to create an object.

Example code:-

```
class Car {
  // static variable
  static int wheels = 4;
  // instance variables
  int modelYear;
  String modelName;
  // constructor and methods
}
```

Static variables are useful when you need to maintain a single shared value across all objects of a class, for example, to keep track of the number of objects created or to represent a constant value that is common to all objects. Accessing a static variable is faster than accessing an instance variable, because you don't need to create an object to access it.

You can access a static variable in two ways:

1.Using the class name: ClassName.staticVariable
2.Using an object reference: objectName.staticVariable (but it is not recommended, because it is less readable)

4.What is the difference between instance variable and local variable ?
ANSWER:-

Instance variables and local variables are two different types of variables in Java. The main difference between them is their scope and visibility.

Instance Variables: Instance variables are declared inside a class, outside of any method, and belong to each object of the class. Each object has its own copy of instance variables, and their values can be different for different objects. Instance variables are accessible from

anywhere within the class and can be used to maintain the state of an object between method invocations.

Local Variables: Local variables are declared within a method or a constructor and are only accessible within the scope of that method or constructor. Local variables are created when the method or constructor is called and destroyed when the method or constructor returns. The values of local variables are not preserved between method invocations.

In summary, instance variables are used to maintain the state of an object across method invocations, whereas local variables are used to hold temporary values within a method or constructor and are not accessible outside the method or constructor.

5.In which area memory is allocated for instance variable and local variable ?
ANSWER:-
In Java, memory is allocated in different areas of the Java virtual machine (JVM) for instance variables and local variables.

Instance Variables: Instance variables are stored in the heap, which is a shared area of memory that holds all objects and their instance variables. The heap is a runtime data area that is created when the JVM starts and is shared by all threads running in the JVM. Each object of a class takes up a unique area of memory in the heap to store its instance variables.

Local Variables: Local variables are stored in the stack, which is a data structure that holds temporary data for each method invocation. The stack is created for each thread running in the JVM and is used to store local variables, method parameters, and intermediate results. When a method returns, the memory used by its local variables is freed.

6.What is method overloading ?
ANSWER:-
Method overloading in Java is a feature that allows multiple methods in a class to have the same name, but with different parameters. This allows you to reuse the same method name for methods that perform similar actions but with different parameters.

When a method is called, the Java compiler determines which version of the method to call based on the number and types of arguments passed in the method call. The process of determining which version of the method to call is called method resolution.

Here's an example of method overloading:

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

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

In the example, the class Calculate has two methods with the same name, add, but with different parameters. When you call the add method, the Java compiler will determine which version of the method to call based on the types of arguments passed in the method call.