Static

1. Static Method
2. Static Variable
3. Static Block
4. Static Class

Static Variable

- Variable that are made static belongs to class, not to a specific object.
- Same for all the objects.
- Accessed using class name (ClassName.staticVariable).
-

Static Method

- Allows accessing a method without using a object,
- Accessed using just the function name.
- Static methods cannot access non-static variables, unless access with an object to the class or using class name.

Static Block

- Executed only once, when we load the class.
- Executed before the constructor.
- Follows sequence with other static blocks in execution.
- Used to initialize the static varible.
- Class loads inmoery only once, and whenever we create an object the constructor is called.

Static Class

- No need to create an object
- Methods can be accessed using the name of outer class

Inner Class

- A class can have: member variable, member method and member class
- The class inside a class is called inner class
- For using the inner class we use outer class prefixed with incer class.
- For creating an object of inner class, we need to use the object of outer class.
- Name of gernerated Class: OuterName$InnerName.class
- Types of Inner Class:
  - Member Class (Normal Inner Class)
  - Static Inner Class
  - Anonumous Class

```
// InnerClass Example
class Outer{
    class Inner{
        public void display(){
            println("Inner Class");
        }
    }
}


public class main{
    public static void main(String args[]){

        Outer out = new Outer();   // Creating outer Class Object

        Outer.Inner in;            // Referencing Inner Class

        in= obj.new Inner();       // Creating the object of Inner Class
    }
}
```

varargs

- variable arguments
- In function defination we add three dots after data type, we can receive any numerb of parameter.
- This parameter behaves as an array and accessbile similar to array

```
public void numbers(int ...a){
    for(int i:a){
        println(i);
    }
}
```

Anonymous Inner Class

- It does not have a name
- Implemented while creating an object
- One time Use
- During object creating we provide the implementation just before closing semicolon
- Can be used for implementing an interface

```
A obj = new A(){
            public void show(){
                System.out.println("From Anon Class");
            }
        };

obj.show();         // Overrides the Base class show() method
```