**Static Keyword**

The static keyword is used to indicate that a particular member (variable, method, or nested class) **belongs to the class itself rather than to instances of the class**. Here are key points to understand static:

1. **Static Variables**: These are shared among all instances of a class. If one instance changes the value of a static variable, all other instances see the change.

class MyClass {

static int count = 0;

}

count is a static variable, shared by all instances of MyClass.

1. **Static Methods**: These can be called without creating an instance of the class. They can only access static variables and other static methods.

class MyClass {

static void display() {

System.out.println("Static method");

} }

MyClass.display(); // No need to create an instance

1. **Static Blocks**: These are used for static initializations of a class. This block runs once when the class is loaded.

class MyClass {

static {

System.out.println("Static block");

} }

1. **Static Classes**: Inner classes can be static, meaning they do not need an instance of the outer class to be instantiated.

class OuterClass {

static class StaticInnerClass {

} }

In summary, static members belong to the class itself rather than any specific instance, making them useful for shared properties, utility methods, or constants.

**Q- Then what is the difference bw Static and Singleton?? IMPORTANT**

* A non-static class can be extended. Polymorphism can save a lot of repetition.
* A non-static class can implement an interface, which can come in handy when you want to separate implementation from API.
* Singleton can be extended. Static can't be.
* Singleton creation may not be threadsafe if it isn't implemented properly. Static is threadsafe.
* Singleton can be passed around as an object. Static can't be.
* Singleton can be garbage collected. Static can't be.
* Singleton is better than static class!