CJ Information

S – AP CS A

IC – Static Keyword

HW – None

A – 2024.11.21 Thu - TEST

BASIS INTERNATIONAL™ SCHOOLS · CHINA

BASIS BILINGUAL™ SCHOOLS · CHINA

STATIC keyword

Outline :

- Why to use STATIC keyword in your code
- Use…not abuse of this keyword

BASIS INTERNATIONAL™ SCHOOLS · CHINA

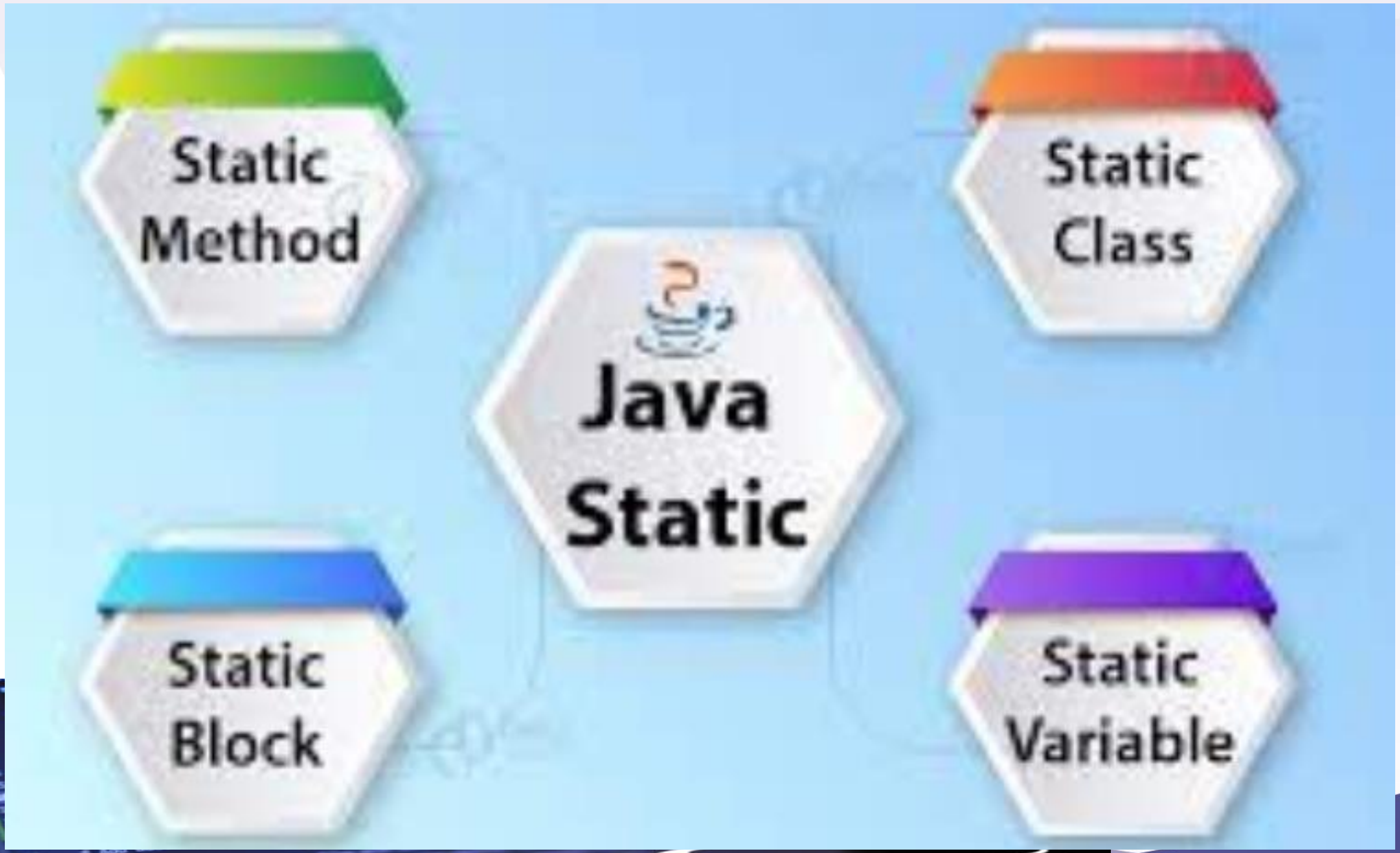
BASIS BILINGUAL™ SCHOOLS · CHINA

What we learn today:

- When and how to use STATIC keyword
- When to use the STATIC keyword
- Examples

STATIC keyword

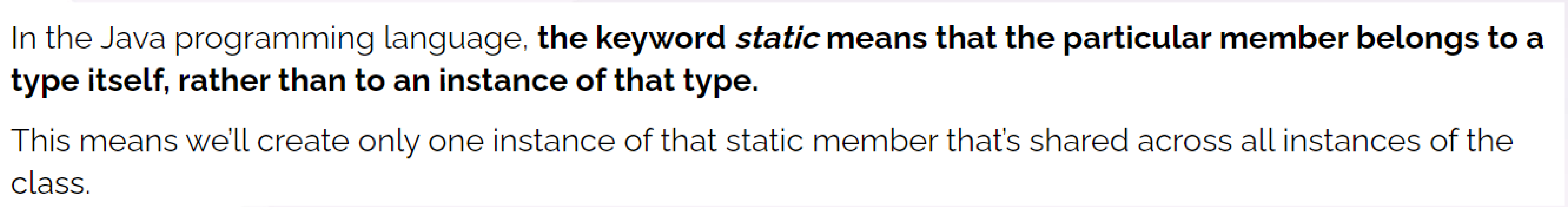
Static can be applied to the following elements in a Java program:



STATIC keyword

The *main concept* represented by Static keyword is :

“when something is declared ***Static***, that something belongs to the class and not to the object”


In the Java programming language, **the keyword *static* means that the particular member belongs to a type itself, rather than to an instance of that type.**

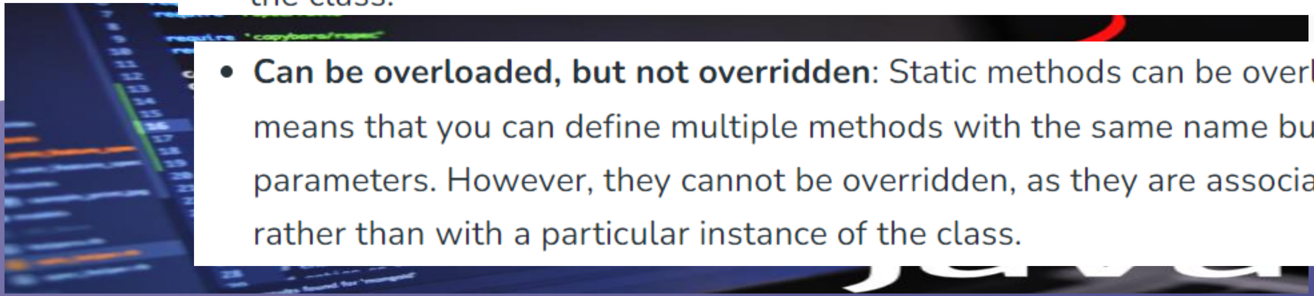
This means we'll create only one instance of that static member that's shared across all instances of the class.

STATIC keyword

Properties for STATIC elements:

- **Shared memory allocation**: Static variables and methods are allocated memory space only once during the execution of the program. This memory space is shared among all instances of the class, which makes static members useful for maintaining global state or shared functionality.
- **Accessible without object instantiation:** Static members can be accessed without the need to create an instance of the class. This makes them useful for providing utility functions and constants that can be used across the entire program.

- **Associated with class, not objects:** Static members are associated with the class, not with individual objects. This means that changes to a static member are reflected in all instances of the class, and that you can access static members using the class name rather than an object reference.
- **Cannot access non-static members:** Static methods and variables cannot access non-static members of a class, as they are not associated with any particular instance of the class.



- **Can be overloaded, but not overridden**: Static methods can be overloaded, which means that you can define multiple methods with the same name but different parameters. However, they cannot be overridden, as they are associated with the class rather than with a particular instance of the class.

BASIS INTERNATIONAL™ SCHOOLS · CHINA

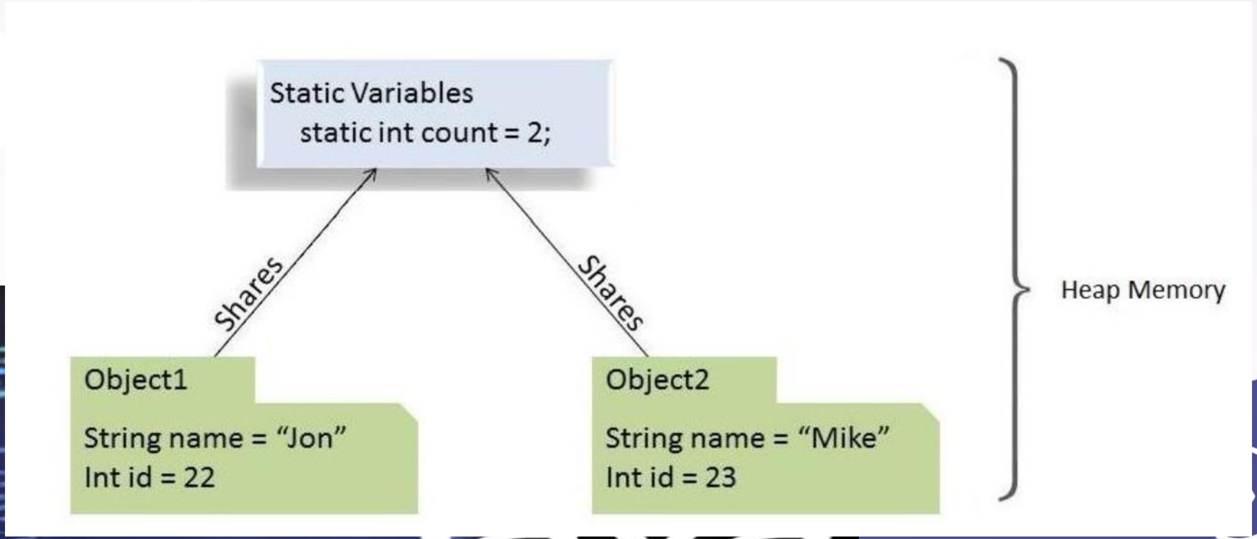
BASIS BILINGUAL™ SCHOOLS · CHINA

STATIC keyword

Static Field (or Class Variable)

In Java, **when we declare a field *static*, exactly a single copy of that field is created and shared among all instances of that class.**

It doesn't matter how many times we instantiate a class. There will always be only one copy of *static* field belonging to it. The value of this *static* field is shared across all objects of either the same class.



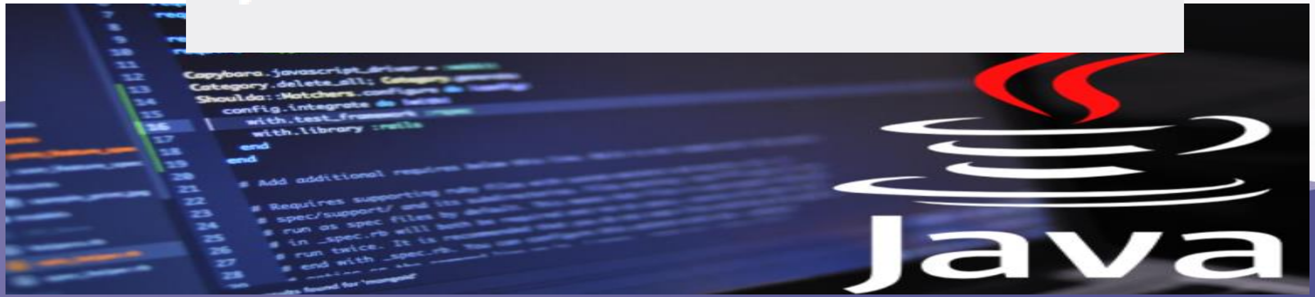
STATIC keyword

Static Field (or Class Variable)

Example

```
public class Counter {
  public static int COUNT = 0;
  Counter() {
    COUNT++;
  }
}
```

```
public class MyClass {
  public static void main(String[] args) {
    Counter c1 = new Counter();
    Counter c2 = new Counter();
    System.out.println(Counter.COUNT);
  }
}
// Outputs "2"
```



STATIC keyword

Static Method

A static method belongs to the class rather than instances. Thus, it can be called without creating instance of class. It is used for altering static contents of the class. There are some restrictions of static methods :

1. Static method can not use non-static members (variables or functions) of the class.
2. Static method can not use `this` or `super` keywords.

BASIS INTERNATIONAL™ SCHOOLS · CHINA

BASIS BILINGUAL™ SCHOOLS · CHINA

STATIC keyword

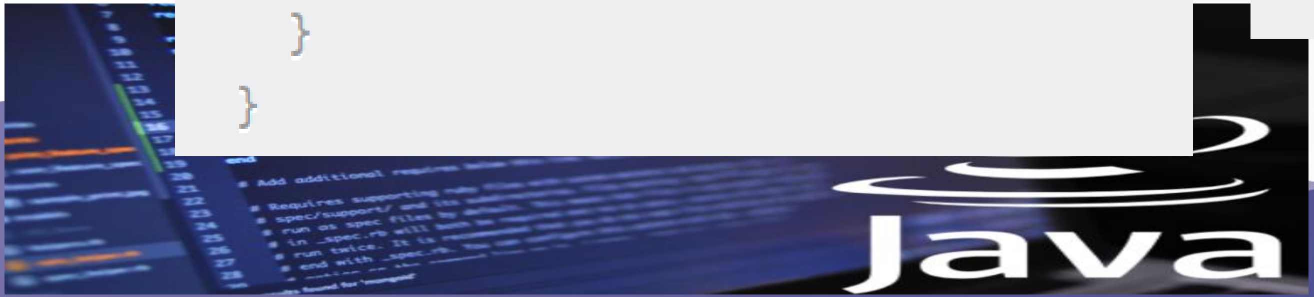
Static Method

Example

```
public class Counter {
  public static int COUNT = 0;
  Counter() {
    COUNT++;
  }

  public static void increment(){
    COUNT++;
  }
}
```

```
public class MyClass {
  public static void main(String[] args) {
    Counter.increment();
    Counter.increment();
    System.out.println(Counter.COUNT);
  }
}
// Outputs "2"
```

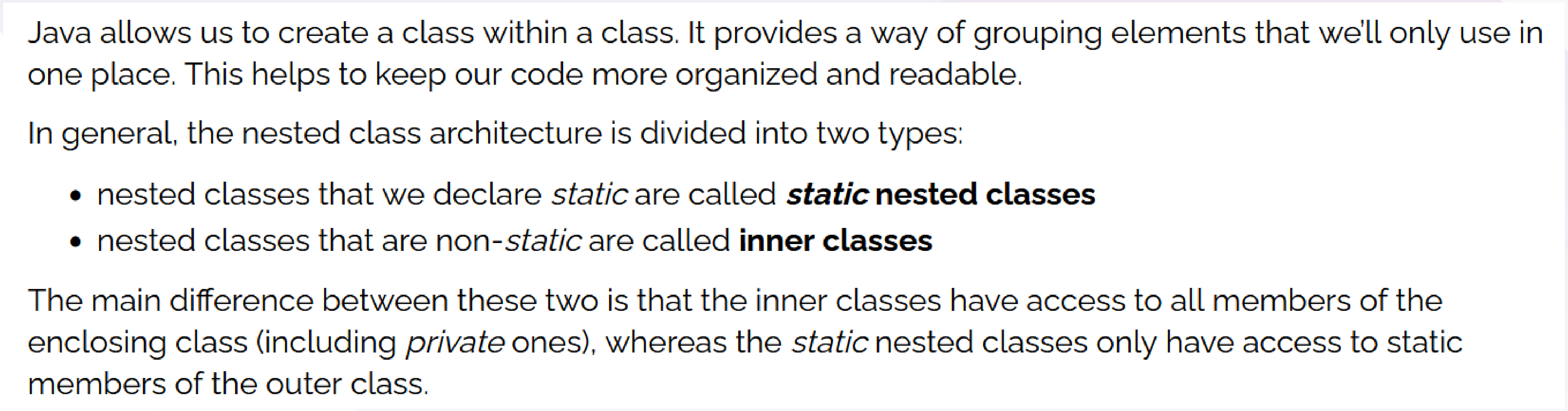


BASIS INTERNATIONAL™ SCHOOLS · CHINA

BASIS BILINGUAL™ SCHOOLS · CHINA

STATIC keyword

Static Class …..only FYI (For Your Information)



Java allows us to create a class within a class. It provides a way of grouping elements that we'll only use in one place. This helps to keep our code more organized and readable.

In general, the nested class architecture is divided into two types:

- nested classes that we declare *static* are called ***static* nested classes**
- nested classes that are non-*static* are called **inner classes**

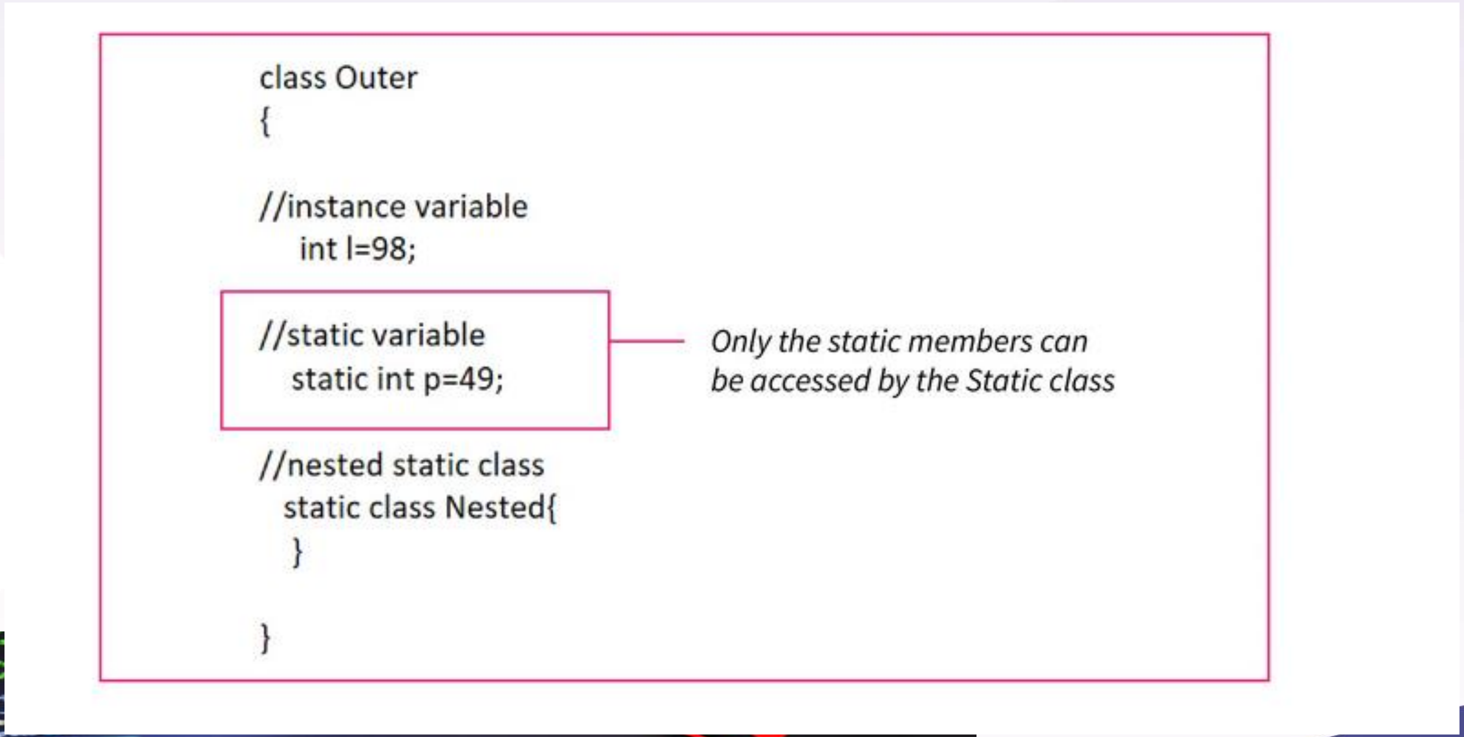
The main difference between these two is that the inner classes have access to all members of the enclosing class (including *private* ones), whereas the *static* nested classes only have access to static members of the outer class.

BASIS INTERNATIONAL™ SCHOOLS · CHINA

BASIS BILINGUAL™ SCHOOLS · CHINA

STATIC keyword

Static Class …..only FYI (For Your Information)



```
class Outer
{

//instance variable
   int I=98;

//static variable
   static int p=49;

//nested static class
  static class Nested{
  }

}
```

Only the static members can be accessed by the Static class

STATIC keyword

To Summarize:

- Understand WHEN to use static keyword
    - Every time you need something that needs to be shared among all the objects you create

- Do NOT abuse of it

BASIS INTERNATIONAL™ SCHOOLS · CHINA

BASIS BILINGUAL™ SCHOOLS · CHINA

STATIC keyword

Check Your Understanding

Read, understand and implement the Java code provided as example

Explanation:

1. **Static Variable**:
   - `static int count = 0;` is a static variable. It is shared by all instances of the class. When you call the static method `incrementCount()`, the `count` variable is modified.
2. **Static Method**:
   - `public static void incrementCount()` is a static method. It can be called directly using the class name (`StaticExample.incrementCount()`) without creating an object.
3. **Instance Method**:
   - `public void displayMessage()` is a non-static method, meaning it can only be called on an object of the class, as shown with `obj.displayMessage()`.

```java
public class StaticExample {
    // Static variable
    static int count = 0;

    // Static method
    public static void incrementCount() {
        count++;
        System.out.println("Count is: " + count);
    }

    // Instance method (non-static)
    public void displayMessage() {
        System.out.println("This is a non-static method!");
    }

    public static void main(String[] args) {
        // Calling the static method directly from the class
        StaticExample.incrementCount();
        StaticExample.incrementCount();

        // Creating an object to call the instance method
        StaticExample obj = new StaticExample();
        obj.displayMessage();
    }
}
```