STATIC METHODS AND FIELDS

Fundamentals

- A **static variable** or **class variable** is any field declared with the `static` modifier; this tells the compiler that there is exactly one (1) copy of this variable in existence, regardless of how many times the class has been instantiated.
  Example:

```
public class StaticVarDemo {
   static int count = 0;
   public void increment() {
       count++;
   }
   public static void main(String args[]) {
       StaticVarDemo obj1 = new StaticVarDemo();
       StaticVarDemo obj2 = new StaticVarDemo();
       obj1.increment();
       System.out.println("Obj1: count = "+ obj1.count);
       obj2.increment();
       System.out.println("Obj1: count = "+ obj1.count);
       System.out.println("Obj2: count = "+ obj2.count);
   }
}
```

  Output: `count` is a static variable, while `increment` is a method whose task is to increase the value of the `count` variable by 1. In the main method, two (2) objects are created, named `obj1` and `obj2`. `obj1` invoked the increment method and showed the value of 1. Then, `obj2` also invoked the `increment` method, resulting in `obj1` having also the value of 2. This is because `count` is a static variable. Only a single copy of a static variable is created and shared among all the instances of the class.
  Explanation:
- A **static method** belongs to the class rather than the object of a class.
  - It can be invoked without creating an instance of a class.
  - It can access static members and change their values.
- Static methods have two (2) main purposes:
  - For utility or helper methods that don't require any object state: Since there is no need to access instance variables, having static methods eliminates the need for the caller to instantiate the object just to call the method.
  - For a state shared by all instances of a class (like a counter): All instances must share the same state. Methods that merely use that state should be static as well.
- To call a static method or variable, put the class name before the method or variable.
  Example:

```
class NumberClass {
    public static int num = 0;
    public static void displayNumber() {
        System.out.print(num);
    }
}
public class NumberClassDemo {
    public static void main(String[] args) {
        System.out.print(NumberClass.num);
        NumberClass.displayNumber();
    }
}
```

  Output: 00
- An instance of an object can be used to call a static method or update a static variable.
  Example:

```
public class NumberClass {
    public static int num = 0;
    public static void main(String[] args) {
        NumberClass.num = 4;
        NumberClass nc1 = new NumberClass();
        NumberClass nc2 = new NumberClass();
        nc1.num = 6;
        nc2.num = 5;
        System.out.print(NumberClass.num);
    }
}
```

  Output: 5
  Explanation: There is only one (1) `num` variable since it is static.

Static versus Instance

- An **instance variable or method** is any variable or method that is non-static.
    - It requires an object of its class to be created before it can be used or called.
- An **instance variable** is any variable that is declared outside any method, constructor, or block.

Type	Accessing	Legal?	How?
Static method	Another static method or variable	Yes	Using the class name
Static method	An instance method or variable	No	
Instance method	A static method or variable	Yes	Using the class name or reference variable
Instance method	Another instance method or variable	Yes	Using a reference variable

Example of a static method calling an instance method:

```
public class NumberClass {
    public static int num = 0;
    public static void main(String[] args) {
        displayNumber(); //does not compile
    }
    public void displayNumber() {
        System.out.print(num);
    }
}
```

Example of a static method trying to access an instance variable:

```
public class NumberClass {
    int num = 0;
    public static void main(String[] args) {
        displayNumber();
    }
    public static void displayNumber() {
        System.out.print(num); //does not compile
    }
}
```

Reference:

Oracle Docs (n.d.). *Citing sources.* Retrieved from https://docs.oracle.com/javase/tutorial/java/javaOO/index.html