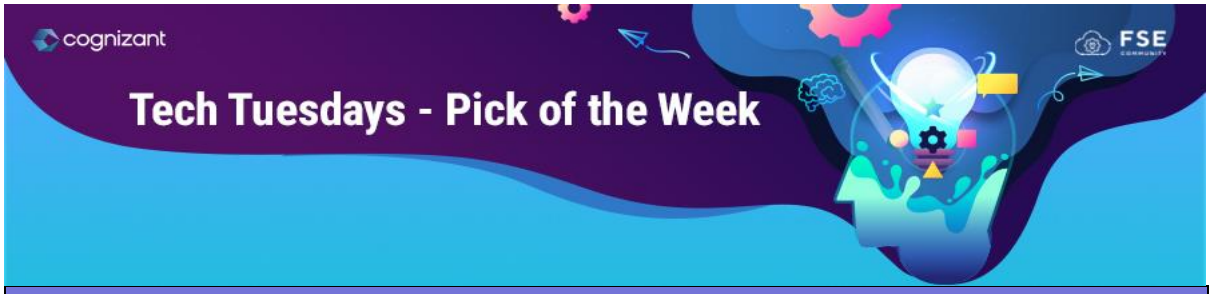

Static Methods in an Interface - Dharini Dorairaj

A new feature, added to **interface,** Java 8 onwards, is the ability to define one or more **static** methods.
Like **static** methods at class level, a **static** method in an interface can be called independently without the instantiation of any object.

As illustrated below, a static method is called by specifying the interface name, followed by the method name separated by a period.

InterfaceName.staticMethodName

The following shows an example of a static method in an interface.
The static method is getDefaultNumber(). It returns Zero.

```
public interface MyIF {
    //This is a normal interface method declaration
    int getNumber();

    //This is default method. Notice that it provides a default implementation.
    default String getString() {
        return "Default String";
    }

    //This is static interface method.
    static int getDefaultNumber() {
        return 0;
    }
}
```

In any of the implementation classes to the interface MyIF, the getDefaultNumber() method can be called by:

int defNum = MyIF.getDefaultNumber();

As mentioned, **no implementation or instance of MyIF is required to call getDefaultNumber() because it is static.**

The **static method in interface is similar to default method; we need not implement them in the implementation classes**. We can safely add them to an interface without changing the code in the implementation classes. static interface methods are not overridden by either an implementing class or a subinterface. The idea behind static interface method is to provide a simple mechanism that allows us to **increase the degree of cohesion of a design by putting together related methods in one single place without having to create an object.**

However, we can declare a static method with the same name in the implementation class and carry out execution through it based on requirements.
In the past, if we had an interface and wanted to group interface-related utils or factory methods, we had to create a separate Util class and store everything there.

Those classes would not have anything in common other than the name, and, additionally, this utils class would need to be made final and have private constructors to forbid unwanted usage. Now, with the interface static methods, we can keep everything in one place without creating additional classes.