Static Methods in interfaces

This lesson explains static methods in interfaces and why they were introduced in Java 8.

We'll cover the following

What are static methods in interfaces?

The static methods in interfaces are similar to default methods but the only difference is that you can't override them. Now, why do we need static methods in interfaces if we already have default methods?

Suppose you want to provide some implementation in your interface and you don't want this implementation to be overridden in the implementing class, then you can declare the method as static.

In the below example, we will defined a `Vehicle` interface with a static method called `cleanVehicle()`.

```
public interface Vehicle {

    static void cleanVehicle(){
        System.out.println("I am cleaning vehicle");
    }
}
```

Let us declare a class `Car`, which implements this `Vehicle` interface.

Car.java

Vehicle.java

All code files are copied to end of the page...

In the above interface, we get a compilation error in the `Car` class because a static method cannot be overridden. Also, since a static method is hidden, we can't call it

from the object of the implementing class. The below code will also not compile.

Car.java

Vehicle.java

All code files are copied to end of the page...

The below class will compile because we are calling the static method that is defined in the interface from the interface reference.

Vehicle.java

Car.java

All code files are copied to end of the page...

What is a default method?

Retake Quiz

In the next lesson, we will explore functional interfaces.

Code Files Content !!!

⤋ ⤋

```
------------------------------------------------------------------------
|  Car.java [1]
------------------------------------------------------------------------


public class Car implements Vehicle {

    @Override
    public void cleanVehicle() {
        System.out.println("Cleaning the vehicle");
    }

    public static void main(String args[]) {
        Car car = new Car();
        car.cleanVehicle();

    }
}



------------------------------------------------------------------------
|  Vehicle.java [1]
------------------------------------------------------------------------
```

```
public interface Vehicle {

    static void cleanVehicle(){
        System.out.println("I am cleaning vehicle");
    }
}
```

```
********************************************************************************
```

```
-------------------------------------------------------------------------
|  Car.java [2]
-------------------------------------------------------------------------
```

```
public class Car implements Vehicle {

    public static void main(String args[]){
        Car car = new Car();

        car.cleanVehicle();  //This will not compile.

    }
}
```

```
-------------------------------------------------------------------------
|  Vehicle.java [2]
-------------------------------------------------------------------------
```

```
public interface Vehicle {

    static void cleanVehicle(){
        System.out.println("I am cleaning vehicle");
    }
}
```

```
********************************************************************************
```

```
-------------------------------------------------------------------------
|  Vehicle.java [3]
-------------------------------------------------------------------------
```

```
public interface Vehicle {

    static void cleanVehicle(){
```

```
        System.out.println("I am cleaning vehicle");
    }
}



--------------------------------------------------------------------------
|  Car.java [3]
--------------------------------------------------------------------------


public class Car implements Vehicle {

    public static void main(String args[]){
        Car car = new Car();

        Vehicle.cleanVehicle(); //This will compile.
    }
}







*******************************************************************************
```