S.No: 16	Exp. Name: ***Write a Java program to implement Interface***	Date:2023-12-12

ID: 224G1A0591

Aim:

Write a Java program that implements an **interface**.

Create an interface called `Car` with two abstract methods `String getName()` and `int getMaxSpeed()`. Also declare one **default** method `void applyBreak()` which has the code snippet

```
System.out.println("Applying break on " + getName());
```

In the same interface include a **static** method `Car getFastestCar(Car car1, Car car2)`, which returns **car1** if the **maxSpeed** of **car1** is greater than or equal to that of **car2**, else should return **car2**.

Create a class called `BMW` which implements the interface `Car` and provides the implementation for the abstract methods **getName()** and **getMaxSpeed()** (make sure to declare the appropriate fields to store **name** and **maxSpeed** and also the constructor to initialize them).

Similarly, create a class called `Audi` which implements the interface `Car` and provides the implementation for the abstract methods **getName()** and **getMaxSpeed()** (make sure to declare the appropriate fields to store **name** and **maxSpeed** and also the constructor to initialize them).

Create a **public** class called `MainApp` with the **main()** method.
Take the input from the command line arguments. Create objects for the classes `BMW` and `Audi` then print the fastest car.

Note:
Java 8 introduced a new feature called `default` methods or `defender` methods, which allow developers to add new methods to the interfaces without breaking the existing implementation of these interface. These **default** methods can also be overridden in the implementing classes or made abstract in the extending interfaces. If they are not overridden, their implementation will be shared by all the implementing classes or sub interfaces.

Below is the syntax for declaring a `default` method in an **interface** :

```
public default void methodName() {
    System.out.println("This is a default method in interface");
}
```

Similarly, **Java 8** also introduced `static` methods inside interfaces, which act as regular static methods in classes. These allow developers group the utility functions along with the interfaces instead of defining them in a separate helper class.

Below is the syntax for declaring a `static` method in an **interface** :

```
public static void methodName() {
    System.out.println("This is a static method in interface");
}
```

Note: Please don't change the package name.

Source Code:

q11284/MainApp.java

```
package q11284;
interface Car {
   String getName();
   int getMaxSpeed();
   default void applyBreak(){
      System.out.println("Applying break on "+getName());
   }
   static Car getFastestCar(Car car1,Car car2){
      if(car1.getMaxSpeed()>=car2.getMaxSpeed())
         return car1;
      else
         return car2;
   }
}
class BMW implements Car {
   String name;
   int maxspeed;
   public BMW(String name,int maxspeed){
      this.name=name;
      this.maxspeed=maxspeed;
   }
   public String getName(){
      return name;
   }
   public int getMaxSpeed(){
      return maxspeed;
   }
}
class Audi implements Car {
   String name;
   int maxspeed;
   public Audi(String name,int maxspeed){
      this.name=name;
      this.maxspeed=maxspeed;
   }
   public String getName(){
      return name;
   }
   public int getMaxSpeed(){
      return maxspeed;
   }
}
public class MainApp {
   public static void main(String args[]) {
      Car b=new BMW(args[0],Integer.parseInt(args[1]));
      Car a=new Audi(args[2],Integer.parseInt(args[3]));
      System.out.println("Fastest car is : "+Car.getFastestCar(b,a).getName());
   }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output

Fastest car is : BMW

Test Case - 2
User Output
Fastest car is : Maruthi