Common Fork Pont

<http://bit.ly/2M2Pxns>

Avatar

<http://cyberdojo1.kgfsl.com/kata/edit/Nza28YgX0X?avatar=seal>

Vehicle Class

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=2&now_tag=3&filename=undefined>

Vehicle Extended For Bike

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=12&now_tag=13&filename=undefined>

Vehicle Extended for Bike and Car

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=13&now_tag=14&filename=undefined>

Accelerate property is Common to bike and car but implementation is different

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=14&now_tag=15&filename=undefined>

Accessing accelerate from vehicle [Not Possible]

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=15&now_tag=16&filename=undefined>

Problem fixed with common implementation of accelerate [Wrong Approach]

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=16&now_tag=17&filename=undefined>

Empty Implementation of accelerate in vehicle

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=17&now_tag=18&filename=undefined>

Creating Abstract Method [accelerate changed to abstract]:

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=20&now_tag=21&filename=undefined>

Abstract method to be placed in abstract class

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=22&now_tag=23&filename=undefined>

Abstract classes cannot be Instantiated it can be inherited

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=23&now_tag=24&filename=undefined>

Fixed by removing vehicle objects from main

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=24&now_tag=25&filename=undefined>

Vehicle Object can hold Bike or Car Instance

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=25&now_tag=26&filename=undefined>

Dynamic Binding

<http://cyberdojo1.kgfsl.com/review/show/Nza28YgX0X?avatar=seal&was_tag=27&now_tag=28&filename=undefined>

Interface Example

<http://cyberdojo1.kgfsl.com/review/show/g26MvCsWHr?avatar=squirrel&was_tag=7&now_tag=8&filename=undefined>

Interface Bike and Car Extended

<http://cyberdojo1.kgfsl.com/review/show/g26MvCsWHr?avatar=squirrel&was_tag=9&now_tag=10&filename=undefined>

Interface

* Interface is similar to classes.
* It is collection of abstract methods.
* Interfaces created for implementations by the classes.
* Interface may contain constants and static methods.
* Interface can be extended.
* If a class implements an interface it must implement all the methods otherwise the classes becomes abstract.

Rules for Interface:

* You cannot instantiate an interface.
* An interface does not contain any constructors.
* All of the methods in an interface are abstract.
* An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
* An interface is not extended by a class; it is implemented by a class.
* An interface can extend multiple interfaces.

Interface Syntax

access\_specifier interface interface\_name{

abstract method\_1

abstract method\_2

...

}

Interface Example

public interface VehicleOperations{

public void start();

public void accelerate();

public void stop();

}

Example Program

/\*

Objective: Demo of Interface

Author: Rajasekaran S

Email: proffraja@gmail.com

\*/

import java.io.\*;

interface VehicleOperations{

public void start();

public void accelerate();

public void stop();

}

class Bike implements VehicleOperations{

boolean isOn=false;

boolean isStarted=false;

public void start(){

System.out.println("Bike Started");

}

public void accelerate(){

System.out.println("Bike Accelerated");

}

public void stop(){

System.out.println("Bike Stoped");

}

}

public class Main{

public static void main(String[] args){

Bike b = new Bike();

b.start();

b.accelerate();

b.stop();

}

}

|  |  |
| --- | --- |
| **Class** | **Interface** |
| 1. It is template definitions of objects 2. It contains variables, constructors, implemented methods. 3. Class can have abstract methods. | 1. It is a collection of abstract methods. 2. It contains final static |