*// Java code to demonstrate*

*// bridge design pattern*

*// abstraction in bridge pattern*

**abstract class Vehicle {**

    protected Workshop workShop1;

    protected Workshop workShop2;

    protected Vehicle(Workshop workShop1, Workshop workShop2)

    {

        this.workShop1 = workShop1;

        this.workShop2 = workShop2;

    }

    abstract public void manufacture();

}

*// Refine abstraction 1 in bridge pattern*

**class Car extends Vehicle {**

    public Car(Workshop workShop1, Workshop workShop2)

    {

        super(workShop1, workShop2);

    }

    @Override

    public void manufacture()

    {

        System.out.print("Car ");

        workShop1.work();

        workShop2.work();

    }

}

*// Refine abstraction 2 in bridge pattern*

**class Bike extends Vehicle {**

    public Bike(Workshop workShop1, Workshop workShop2)

    {

        super(workShop1, workShop2);

    }

    @Override

**public void manufacture()**

    {

        System.out.print("Bike ");

        workShop1.work();

        workShop2.work();

    }

}

*// Implementor for bridge pattern*

**interface Workshop**

{

    abstract public void work();

}

*// Concrete implementation 1 for bridge pattern*

**class Produce implements Workshop {**

    @Override

    public void work()

    {

        System.out.print("Produced");

    }

}

*// Concrete implementation 2 for bridge pattern*

**class Assemble implements Workshop {**

    @Override

    public void work()

    {

        System.out.print(" And");

        System.out.println(" Assembled.");

    }

}

*// Demonstration of bridge design pattern*

**class BridgePattern {**

    public static void main(String[] args)

    {

        Vehicle vehicle1 = new Car(new Produce(), new Assemble());

        vehicle1.manufacture();

        Vehicle vehicle2 = new Bike(new Produce(), new Assemble());

        vehicle2.manufacture();

    }

}

Car Produced And Assembled.

Bike Produced And Assembled.