interface Vehicle {

int set_num_of_wheels()

int set_num_of_passengers()

boolean has_gas()

}

1. Explain how can you use the pattern to create car and plane class?

The code here follows JAVA's basic fundamental OO construct: polymorphism. The interface is only provided here and without any more specific class definitions, it can't be deduced from just the interface what design pattern it is actually following.But it is close to Factory Design Pattern Concept.

Car and Plane class can be created by using the given pattern in following way:

Project Name: FactoryVehicle

```
import java.io.*;
interface Vehicle {
int set_num_of_wheels();
int set_num_of_passengers();
 boolean has_gas();
 }

                                                                                   class
Car implements Vehicle{
    @Override
   public int set_num_of_wheels(){
          return 4;
      }
    @Override
    public  int set_num_of_passengers(){
          return 4;
      }
     @Override
    public  boolean has_gas(){
          return true;
      }
  }
  class Plane implements Vehicle{
        @Override
       public  int set_num_of_wheels(){
          return 6;
      }
    @Override
    public  int set_num_of_passengers(){
          return 20;
      }
      @Override
     public boolean has_gas(){
          return false;
      }
  }
  public class createVehicle{
      public static void main(String[] args){
          Car newCar = new Car();

          System.out.println("Car has:"+newCar.set_num_of_wheels()+" wheels,"+
          newCar.set_num_of_passengers()+" passengers and"+ " "+
          "gas usage is :"+ newCar.has_gas());

        Plane newPlane = new Plane();
        System.out.println("Plane has:"+ newPlane.set_num_of_wheels()+" wheels,"+
        newPlane.set_num_of_passengers()+" passengers and"+ " "+
        "gas usage is :"+ newPlane.has_gas());
    }
}
```

Result

CPU Time: 0.26 sec(s), Memory: 35860 kilobyte(s) **compiled and executed in 0.985 sec(s)**

```
Car has:4 wheels,4 passengers and gas usage is :true
Plane has:6 wheels,20 passengers and gas usage is :false
```

2. Use a different design pattern for this solution

The given problem has been re-implemented using Abstract class of Java. The code can be found in “Code for vehicle design pattern” folder in the file name BuilderDemo.java.