A factory design pattern is used in the example.

```
interface Vehicle {

      int set_num_of_wheels()
      int set_num_of_passengers()
      boolean has_gas()
}
```

The pattern can be used to create car and planes using the following.
Planes and Cars classes are created:

```
public class Planes implements Vehicle {

   @Override
   public void set_num_of_wheels() {

   }
   public void set_num_of_passengers() {

   }
   public void has_gas() {

   }
}
```

```
public class Cars implements Vehicle{

   @Override
   public void set_num_of_wheels() {

   }
   public void set_num_of_passengers() {

   }
   public void has_gas() {

   }
}
```

Factory class is designed:

```
public class VehicleFactory {
      public Vehicle getVehicle(String vehicleType){
      if(vehicleType == null){
         return null;
      }
      if(vehicleType.equalsIgnoreCase("CAR")){
         return new Cars();

      } else if(vehicleType.equalsIgnoreCase("PLANE")){
         return new Planes();

      }
      return null;
   }
}
```

Finally, a concrete class is created

```
public class FactoryPatternDemo {

   public static void main(String[] args) {
      VehicleFactory vehicleFactory = new VehicleFactory();

      Vehicle vehicle1 = vehicleFactory.getVehicle("CAR");

      Vehicle vehicle2 = vehicleFactory.getVehicle("Plane");
      // methods of respective vehicles can be drawn here

   }
}
```