* Abstract Factory patterns work around a super-factory which creates other factories.
* This factory is also called as factory of factories.
* This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.
* In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes.
* Each generated factory can give the objects as per the Factory pattern.

Use-Case : If we have Vehicles , Cars and Bikes and subtypes also in them , we can have two factories one for Car / Bike and other for the subtype.

public interface Vehicle {  
  
 void getInfo() ;  
}

public abstract class AbstractVehicleFactory {  
  
 public abstract Vehicle getVehicle(String vehicleType) ;  
  
}

public class BikeFactory extends AbstractVehicleFactory{  
  
 @Override  
 public Vehicle getVehicle(String vehicleType) {  
 if(vehicleType.equals("Normal Bike")) {  
 return new NormalBike() ;  
 } else if (vehicleType.equals("Sports Bike")) {  
 return new SportsBike() ;  
 }else {  
 return null;  
 }  
 }  
}

public class CarFactory extends AbstractVehicleFactory{  
 @Override  
 public Vehicle getVehicle(String vehicleType) {  
 if(vehicleType.equals("Micro Car")) {  
 return new MicroCar() ;  
 } else if (vehicleType.equals("Mega Car")) {  
 return new MegaCar() ;  
 }else {  
 return null;  
 }  
 }  
}

public class MegaCar implements Vehicle{  
 @Override  
 public void getInfo() {  
 System.*out*.println("I am a Mega Car! ");  
 }  
}

public class MicroCar implements Vehicle{  
 @Override  
 public void getInfo() {  
 System.*out*.println("I am a Micro Car! ");  
 }  
}

public class NormalBike implements Vehicle{  
 @Override  
 public void getInfo() {  
 System.*out*.println("I am a Normal Bike! ");  
 }  
}

public class SportsBike implements Vehicle{  
  
 @Override  
 public void getInfo() {  
 System.*out*.println("I am a Sports Bike! ");  
 }  
  
}

public class VehicleFactoryProducer {  
 public static AbstractVehicleFactory getFactory(String factory ){  
 if(factory.equals("car")) {  
 return new CarFactory() ;  
 } else if (factory.equals("bike")) {  
 return new BikeFactory() ;  
 }else {  
 return null;  
 }  
 }  
}

public class AbstractFactoryDesignPattern {  
 public static void main(String[] args) {  
  
  
 Vehicle v1 = VehicleFactoryProducer.*getFactory*("car").getVehicle("Micro Car" );  
 Vehicle v2 = VehicleFactoryProducer.*getFactory*("bike").getVehicle("Sports Bike" );  
  
 v1.getInfo();  
 v2.getInfo();  
  
 }  
}

Output :

I am a Micro Car!

I am a Sports Bike!

Process finished with exit code 0