

Factory Pattern : (creational design Pattern)

Page No.

Date / /

It provides an interface for creating objects in a superclass, allowing subclasses to alter the type of objects that will be created.

It encapsulate object creation logic in a separate method, promoting loose coupling between the creator and the Created object!

Need:-

cond 1: Whenever we want to create an object

cond 2: But object creation is based on some condition

Let say we have 100 classes

class 1	...	class 50	...	class 100
if (cond) return obj;		if (cond) return obj;		

code duplicacy while returning object
because we have to add this condition to
return this obj again and again in every
class

Solution Factory Pattern is needed to avoid duplicacy of code when object is generated with condition.

Shape factory class main shape
interface ka object hai

Page No.

Date / /

Ex:

SHAPE << Interface >>

Shape factory

Shape getShape()

has-a

void draw();

is-a

circle

square

Rectangle

draw()

draw()

draw()

has-a

main()

Shape getShape(String input)

Switch(input)

case "Circle"

return new Circle()

case "Square"

return new Square()

case "Rectangle"

return new Rectangle()

main function created
Shapefactory object

Shapefactory obj = new Shapefactory();
Shape obj1 = obj.getShape("circle");
obj1.draw();

Now this
draw function will be
class circle

used when we have products that can be grouped into two or more forms as independent factories.

Date / /

● Abstract Factory Pattern. (Factory of Factory)

[Lux → Luxury
ord - ordinary]

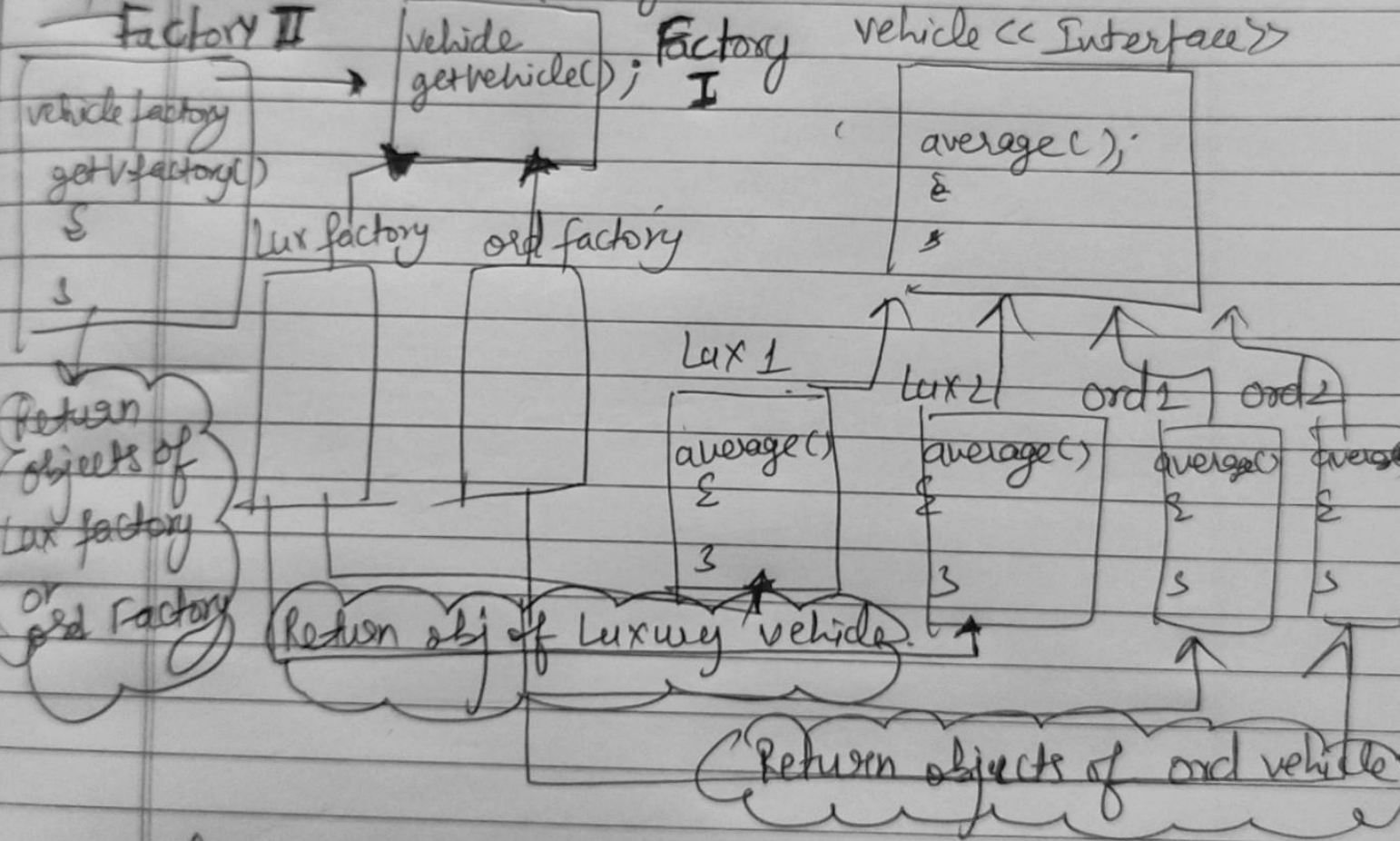
Ex:

Factory II

vehicle factory

Factory I

vehicle << Interface >>



* Abstract Factory Pattern is used when there is need of more than one Factory.

factory I: Returns the object of actual vehicle based on two different group (Lux, ord)

factory II: Returns the objects of vehicle factory i.e either lux factory or ord factory.