Factory Pattern : (creational design Pattern)

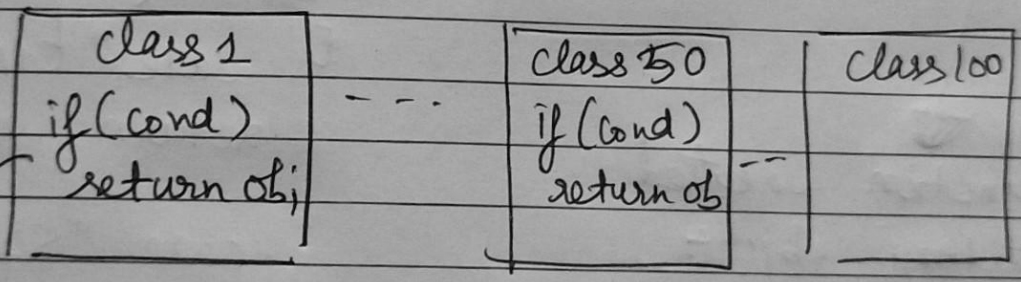
It provides an interface for creating objects in a superclass, allowing subclasses to alter the type of objects that will be created.

It encapsulate object creation logic in a seperate 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



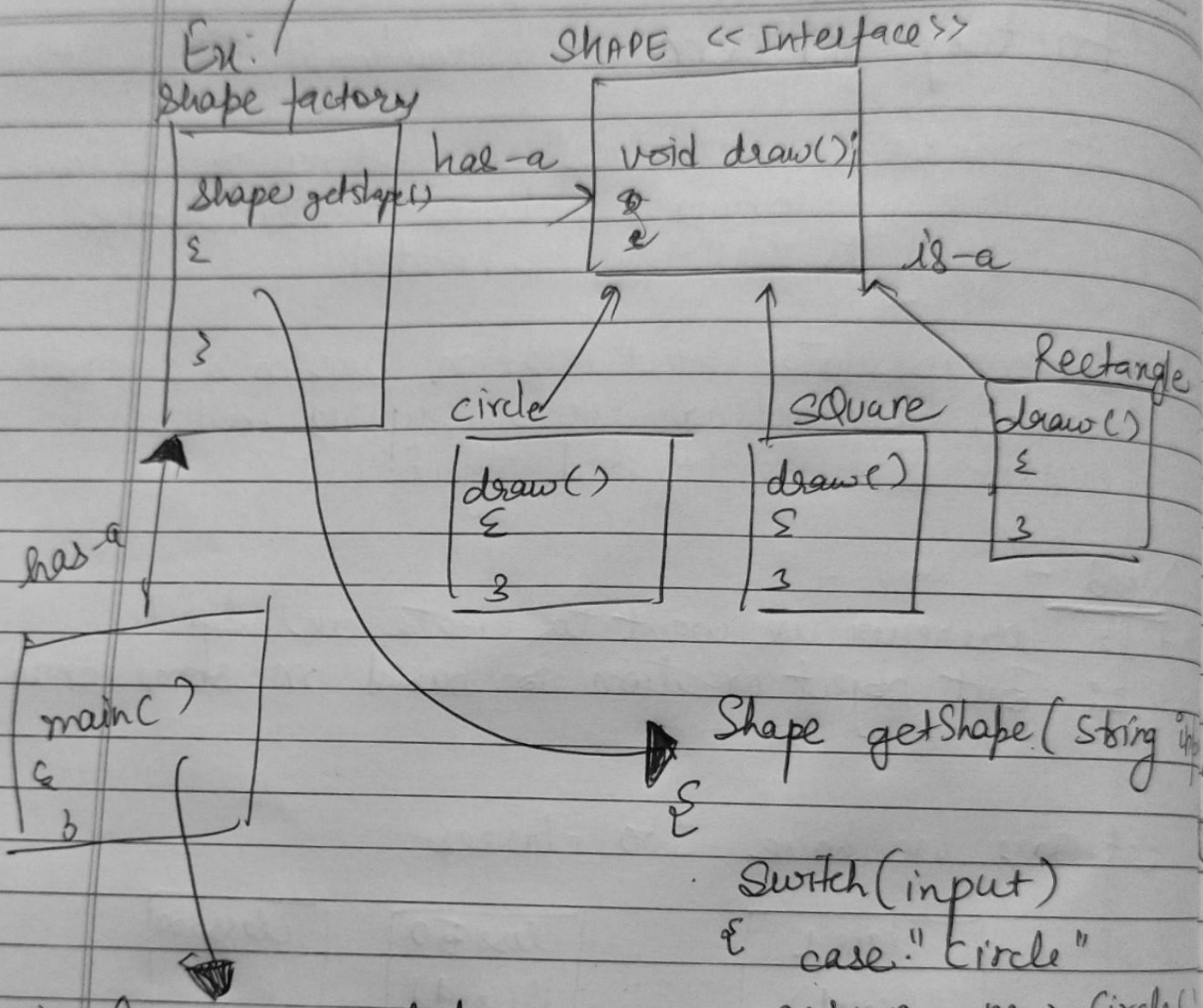
code duplicacy while returning object because we have to add this condition to return this obj again and again in every class

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

Shape factory class main shape interface ka object hai

Page No.
Date / /

Ex:



main function created shapefactory object

```
Shapfactory obj = new shapefactory();
Shape obj1 = obj.getshape("circle");
obj1.draw();
```

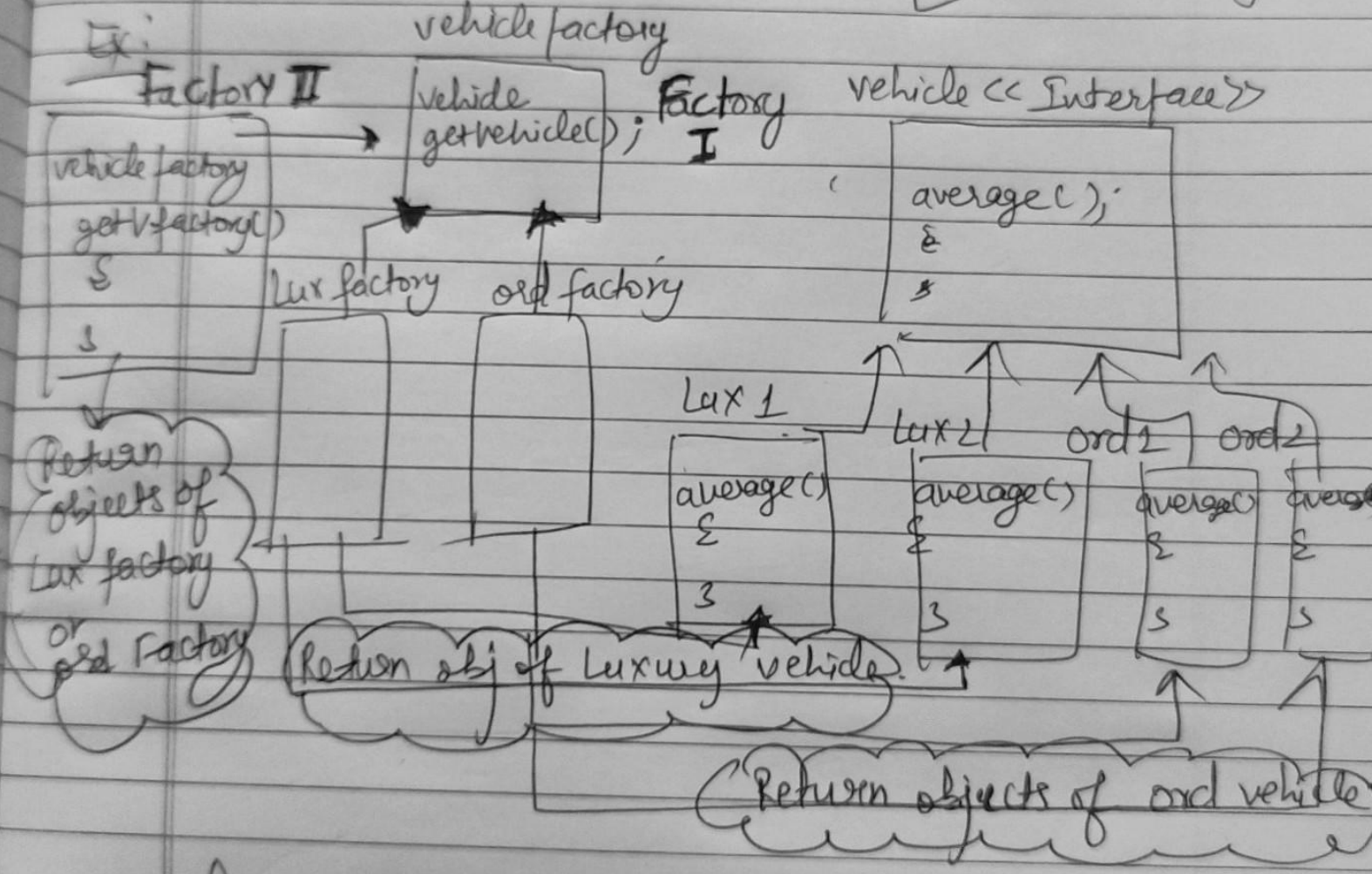
Now this draw function will be class circle

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]



* 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.