ABHISHEK DHAR
23201
S-10

ASSIGNMENT-10 (OOP)

FACTORY DESIGN PATTERN

Title→ Factory Design Pattern

Aim→ Design and implement factory design pattern for the given context. Consider car building process, which requires many steps from allocating accessories to final makeup. These steps should be written as methods and should be called while creating an instance of a specific or type. Sedan, SUV ~~called~~ be the subclass of car class. Car class and its subclasses, carfactory, TestFactory pattern. should be implemented.

Objectives→ To learn the concept of design pattern

Theory→

1) Design Pattern→

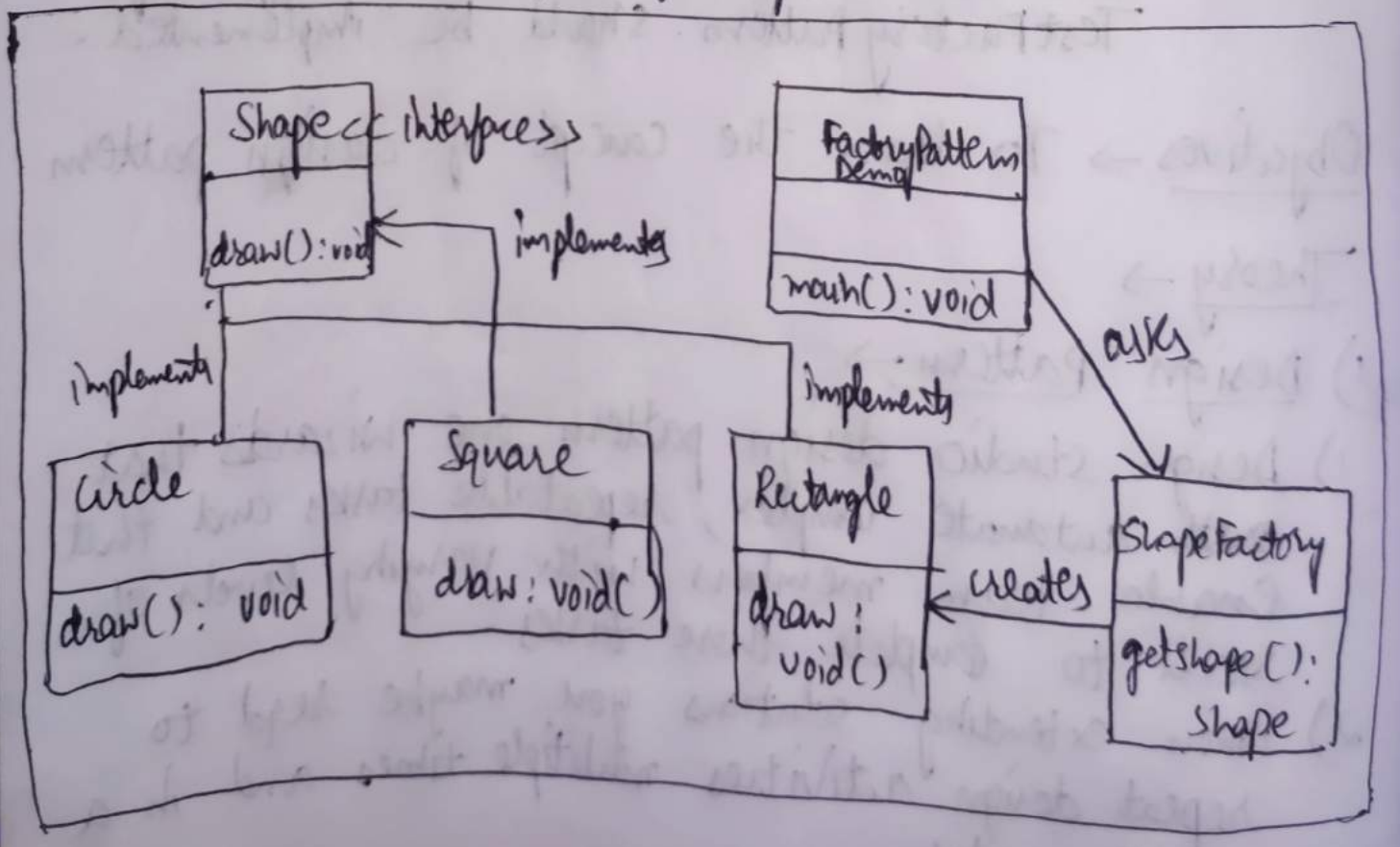
1) Design studio design pattern one wizards that ~~auto~~ automate complex, repeatable tasks and that enable team members with varying levels of skill to complete those tasks.
2) When extending solutions you maybe regd to repeat design activities multiple times and in a specific order.

3) Design patterns enable you to define a generic pattern that when executed automates the creation of model objects and their relationships in a user's workspace.

4) Your teams can design patterns to reduce errors, simplify modeling and increase productivity.

2) Factory design pattern with diagram & example-

Consider a shape interface and concrete classes implementing the shape interface - We will have a ShapeFactory as Factory class and FactoryPatternDemo which will use ShapeFactory to get the type of object it needs



1) Shape.java

```
public interface Shape {
    void draw();
}
```

2) Rectangle.java

```
public class Rectangle implement Shape {
    @override
    public void draw() { System.out.println("Draw inside rectangle"); }
}
```

Square & circle.java will be similar to Rectangle.java except print statement will be replaced by square & circle.

3) ShapeFactory.java →

```
public class ShapeFactory {
    //getshape method to get object of reqd. type
    public Shape getShape(String ShapeType)
    {
        if(ShapeType = null) {return null;}
        if(ShapeType = "circle") { return new Circle(); }
        if(ShapeType = "Square") { return new Square(); }
        if(ShapeType = "Rectangle") {return new Rectangle();}
        return null; } }
```

4) FactoryPatternDemo.java

```
Public class FactoryPatternDemo {
  Public static void main(String [] args) {
  ShapeFactory shapeFactory = new ShapeFactory();
  Shape shape1 = shapeFactory.getShape("Circle");
  Shape shape2 = shapeFactory.getShape("square");
  Shape shape3 = shapeFactory.getShape("Rectangle");

  Shape1.draw();
  Shape2.draw();
  Shape3.draw(); };
```

Output →

```
Draw inside circle
Draw inside square
Draw inside Rectangle
```

3) Advantages of Factory Design Pattern →

1) The object that we create can be used without duplication of code.
2) If we a factory method instead of constructor the factory method can have different and descriptive names also.

3) It removes the instantiation of the implementation classes.

4) Makes code more robust, less coupled and easy to expand.

5) Through inheritance, it provides abstraction b/w implementation & the client classes.

4) Usage and the application where factory design patterns can be applied →

A) Factory Method pattern can be used in following cases

i) A class cannot anticipate the type of objects it needs to create beforehand.

ii) A class requires its subclasses to specify the objects it creates.

iii) You want to localize the logic to instantiate into complex object.

iv) When parent class chooses the creation of objects of its subclass.

2B201

Class Diagram →