Factory Abstract Pattern

Instructions for code: Heavily comment your code throughout the program.

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.

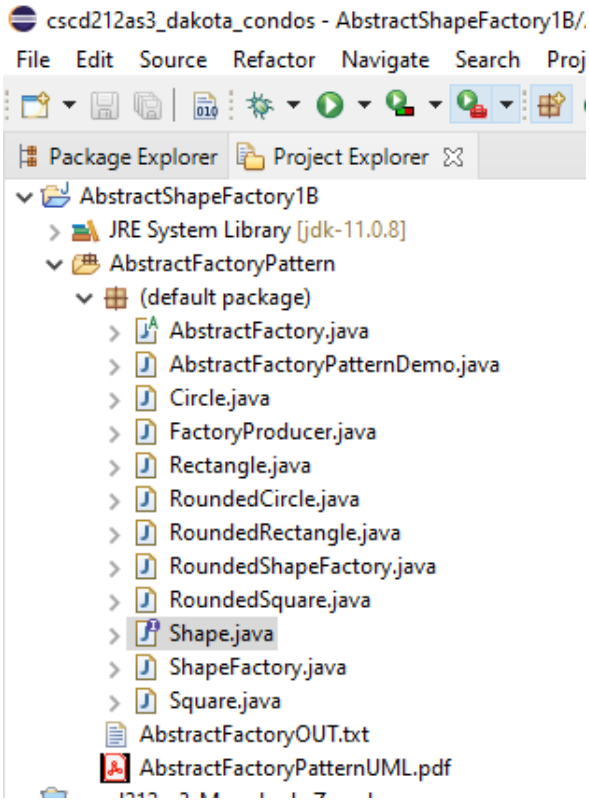
In Abstract Factory pattern an interface is responsible for creating a factory of related objects without explicitly specifying their classes. Each generated factory can give the objects as per the Factory pattern.

Implementation

We are going to create a Shape interface and a concrete class implementing it. We create an abstract factory class AbstractFactory as next step. Factory class ShapeFactory is defined, which extends AbstractFactory. A factory creator/generator class FactoryProducer is created.

AbstractFactoryPatternDemo, our demo class uses FactoryProducer to get a AbstractFactory object. It will pass information (CIRCLE / RECTANGLE / SQUARE for Shape) to AbstractFactory to get the type of object it needs.

Directory Structure in Eclipse:



Step 1

Create an interface for Shapes.

Shape.java

Step 2

Create concrete classes implementing the same interface.

RoundedSquare.java

Rectangle.java

Step 3

Create an Abstract class to get factories for Normal and Rounded Shape Objects.

AbstractFactory.java

Step 4

Create Factory classes extending AbstractFactory to generate object of concrete class based on given information.

ShapeFactory.java

RoundedShapeFactory.java

Step 5

Create a Factory generator/producer class to get factories by passing an information such as Shape

FactoryProducer.java

Step 6

Use the FactoryProducer to get AbstractFactory in order to get factories of concrete classes by passing an information such as type.

AbstractFactoryPatternDemo.java

Step 7

Verify the output.

```
Inside Rectangle::draw() method.
Inside Square::draw() method.
Inside RoundedRectangle::draw() method.
Inside RoundedSquare::draw() method.
```