**Abstract Factory Pattern**

* **Introduction –**
* 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.
* I 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 ate going to create a Shape and Color *interfaces* and concrete classes implementing these *interfaces*. We create an abstract factory class Abstract Factory as next step. Factory classes Shape Factory and Color Factory are defined where each factory extends Abstract Factory. A factory creator/generator class Factory Producer is created.
* Main, our main class, uses factory Producer to get an Abstract Factory object. It will pass information (CIRCLE / RECTANGLE / SQUARE) to Abstract Factory to get the type of object it needs. It also passes information (RED / GREEN / BLUE for Color) to Abstract Factory to get the type of object it needs.
* **Required Steps for Implementation –**
* Step-1: Create an *interface*: **Shape.java**
* Step-2: Create concreate classes implementing the same *interface*: **Rectangle.java**, **Square.java**, **Circle.java**
* Step-3: Create an *interface* for Colors: **Color.java**
* Step-4: Create an Abstract class to ger factories for Color and Shape Objects: **AbstractFactory.java**
* Step-6: Create Factory classes extending Abstract Factory to generate object of concrete class based on given information: **ShapeFactory.java**, **ColorFactory.java**
* Step-7: Create a Factory generator/producer class to get factories by passing an information such as Shape or Color: **FactoryProducer.java**