**Design Pattern**

* **Design pattern**
  + A design patterns are **well-proved solution** for solving the specific problem/task.
* **Factory Design Pattern**
  + - In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface

**Shape.java**

public interface Shape {

void draw();

}

**Rectangle.java**

public class Rectangle implements Shape {

@Override

public void draw() {

System.out.println("Inside Rectangle::draw() method.");

}

}

**Square.java**

public class Square implements Shape {

@Override

public void draw() {

System.out.println("Inside Square::draw() method.");

}

}

**Circle.java**

public class Circle implements Shape {

@Override

public void draw() {

System.out.println("Inside Circle::draw() method.");

}

}

**Step 3**

Create a Factory to generate object of concrete class based on given information.

**ShapeFactory.java**

public class ShapeFactory {

//use getShape method to get object of type shape

public Shape getShape(String shapeType){

if(shapeType == null){

return null;

}

if(shapeType.equalsIgnoreCase("CIRCLE")){

return new Circle();

} else if(shapeType.equalsIgnoreCase("RECTANGLE")){

return new Rectangle();

} else if(shapeType.equalsIgnoreCase("SQUARE")){

return new Square();

}

return null;

}

}

**Singleton Design Pattern**

This pattern involves a single class which **is responsible to create an object while making sure that only single object** gets created.

public class SingleObject {

//create an object of SingleObject

private static SingleObject instance = new SingleObject();

//make the constructor private so that this class cannot be

//instantiated

private SingleObject(){}

//Get the only object available

public static SingleObject getInstance(){

return instance;

}

public void showMessage(){

System.out.println("Hello World!");

}

}

**Proto Type Design Pattern**

* Prototype pattern refers to **creating duplicate object** while keeping performance in mind.
* This pattern is used when creation of object directly is costly.
  + For example, an object is to be created after a costly database operation.