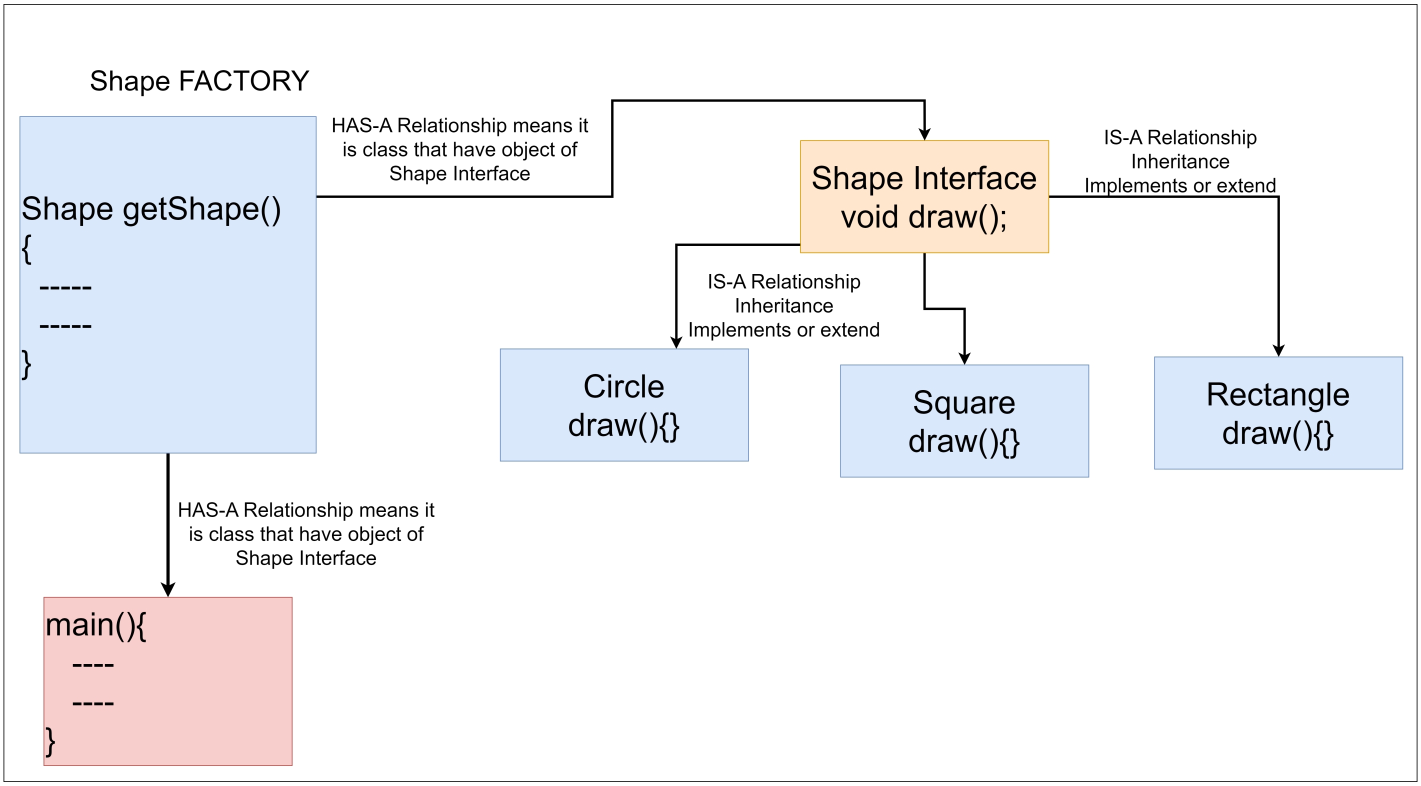
Factory Design Pattern [LLD].

1. **FACTORY vs ABSTRACT FACTORY PATTERN.**

- When we want to create object on conditional basis.
- IS-A Relationship **INHERITANCE.**



```
package System_Design.DesignPatterns.factory_design_pattern.shape_example;

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

```
package System_Design.DesignPatterns.factory_design_pattern.shape_example;

public class Circle implements Shape{
    @Override
    public void draw() {
        System.out.println("Circle");
    }
}
```

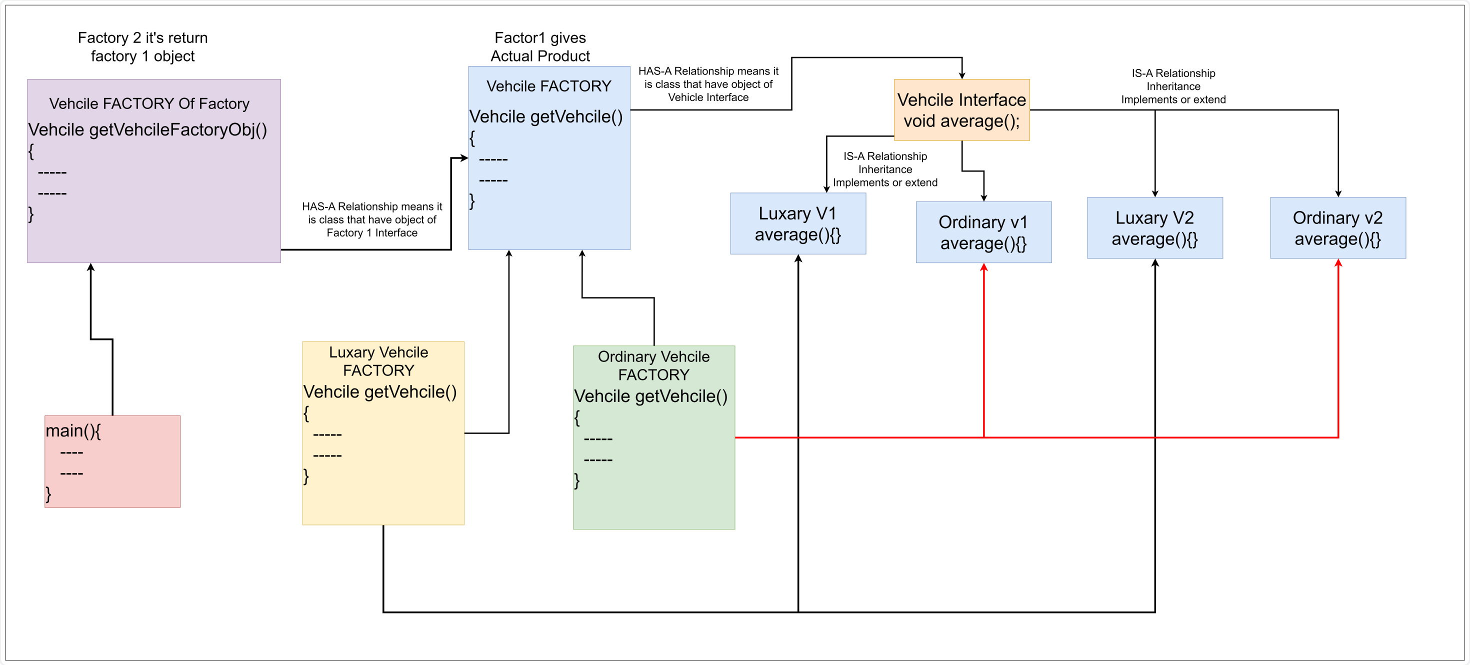
```
package System_Design.DesignPatterns.factory_design_pattern.shape_example;

public class Square implements Shape{
    @Override
    public void draw() {
        System.out.println("Square");
    }
}
```

```
package System_Design.DesignPatterns.factory_design_pattern.shape_example;

public class Rectangle implements Shape{
    @Override
    public void draw() {
        System.out.println("Rectangle");
    }
}
```

```
package System_Design.DesignPatterns.factory_design_pattern.shape_example;

public class ShapeFactory {
    public Shape getShape(String input){
        switch (input){
            case "CIRCLE":
                return new Circle();
            case "RECTANGLE":
                return new Rectangle();
            case "SQUARE":
                return new Square();
            default:
                return null;
        }
    }
}
```

```
package System_Design.DesignPatterns.factory_design_pattern.shape_example;

public class ShapeMain {
    public static void main(String[] args){
        ShapeFactory factory = new ShapeFactory();

        Shape shapeCircle = factory.getShape("CIRCLE");
        shapeCircle.draw();

        Shape shapeRectangle = factory.getShape("RECTANGLE");
        shapeRectangle.draw();

        Shape shapeSquare = factory.getShape("SQUARE");
        shapeSquare.draw();

    }
}
output:
Circle
Rectangle
Square

Process finished with exit code 0
```

ABSTRACT Factory


Factory 2 it's return
factory 1 object
Vehcile FACTORY Of Factory
Vehcile getVehcileFactoryObj()
{

}
HAS-A Relationship means it
is class that have object of
Factory 1 Interface
Factor1 gives
Actual Product
Vehcile FACTORY
Vehcile getVehcile()
{

}
HAS-A Relationship means it
is class that have object of
Vehicle Interface
Vehcile Interface
void average();
IS-A Relationship
Inheritance
Implements or extend
IS-A Relationship
Inheritance
Implements or extend
Luxary V1
average(){}
Ordinary v1
average(){}
Luxary V2
average(){}
Ordinary v2
average(){}
main(){

}
Luxary Vehcile
FACTORY
Vehcile getVehcile()
{

}
Ordinary Vehcile
FACTORY
Vehcile getVehcile()
{

}