Description :

This project is an example of Composite Design Pattern Implementation. Composite is a structural design pattern that lets us compose objects into tree structures and then work with these structures as if they were individual objects.

Here village is a global composite object. And the village is built using: House, Tree, River, and Market. Each House, Tree, River, and Market are composed of smaller shapes. The market is built using House 2 and the river. House 1 is composed of a rectangle and triangle. House 2 is composed of two squares and a triangle. The tree is a composite of a line and a circle. A river is a composite object of aline and an oval. The basic shapes are independent of other shapes. In this way, the village is built using the composite design pattern.

Code:

First, we create an interface named shape and two methods named shapeDescription() and shapeReturn();

```
public interface Shape {
    void shapeDescription();
    String shapeReturn();

}
```

We have two classes Pattern class and composite class. Both of them implements the shape interface. The pattern is used to create single independent objects, and the composite class creates composite objects using the output of the pattern class. And both of the pattern and composite classes override the methods shapeDescription() and shapeReturn() of their parent class

We have a list named “shapes” of type shape in the composite class because a composite object can have two or more independent or composite shapes. In this class, we also have a method of adding conditions to the list named addShape().

```java
public class Pattern implements Shape {
    String name;

    public Pattern(String name) {
        super();
        this.name = name;
    }

    public void shapeDescription() {
        System.out.println(name);
    }
    public String shapeReturn() {
        return name;
    }

}
```

```java
import java.util.*;
public class Composite implements Shape {
    String name;
    List<Shape> shapes= new ArrayList<>();

    public Composite(String name) {
        super();
        this.name = name;
    }
    public void addShape(Shape shape)
    {
        shapes.add(shape);
    }
        public String shapeReturn() {
        return name;
    }

    public void shapeDescription() {
        System.out.println(name+" is built using : ");
        for(Shape a : shapes)
        {
            String name = a.shapeReturn();
            System.out.println( name +"\n");
        }
        System.out.println("\n");
    }

}
```

Finally, in the main class, we create house1, house2, tree, river, market, and subsequently the village using the following hierarchy -