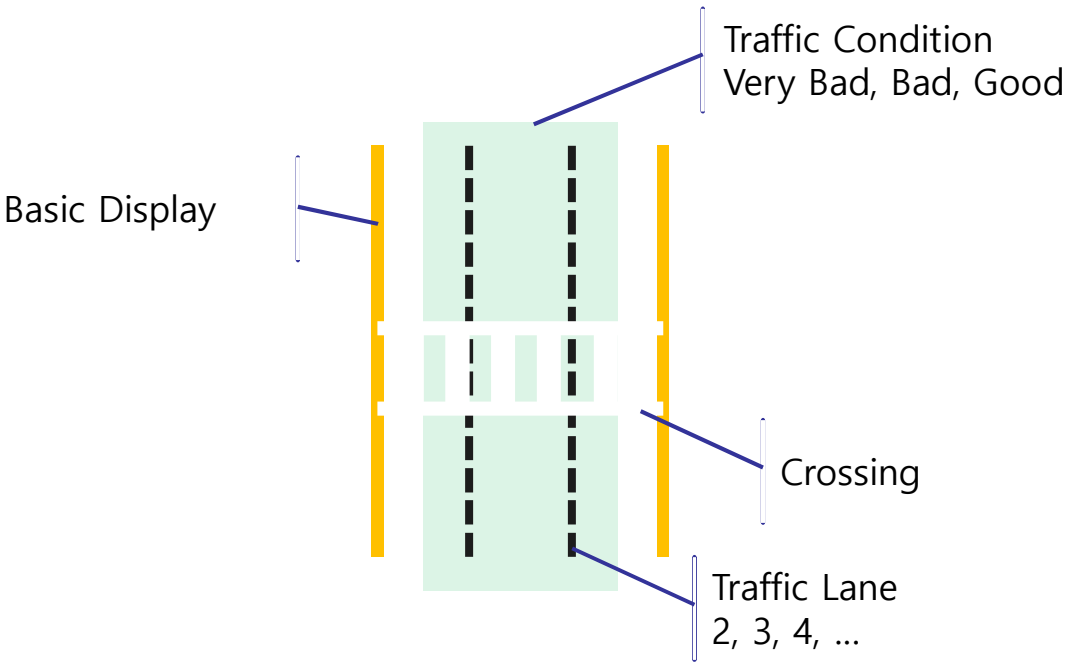
Decorator pattern

PRACTICE – ROAD DISPLAY

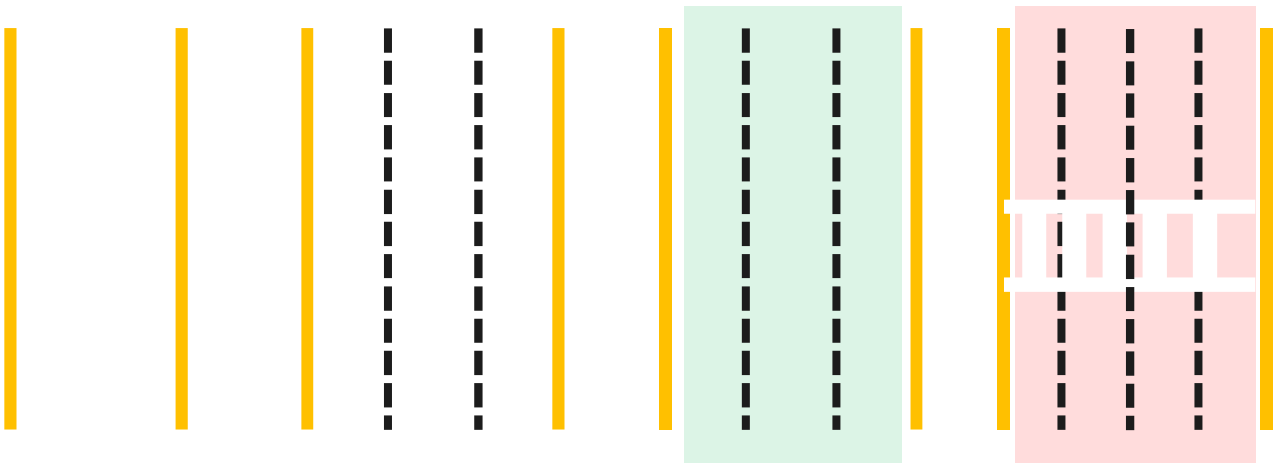
Road Display

- Road consists of several components

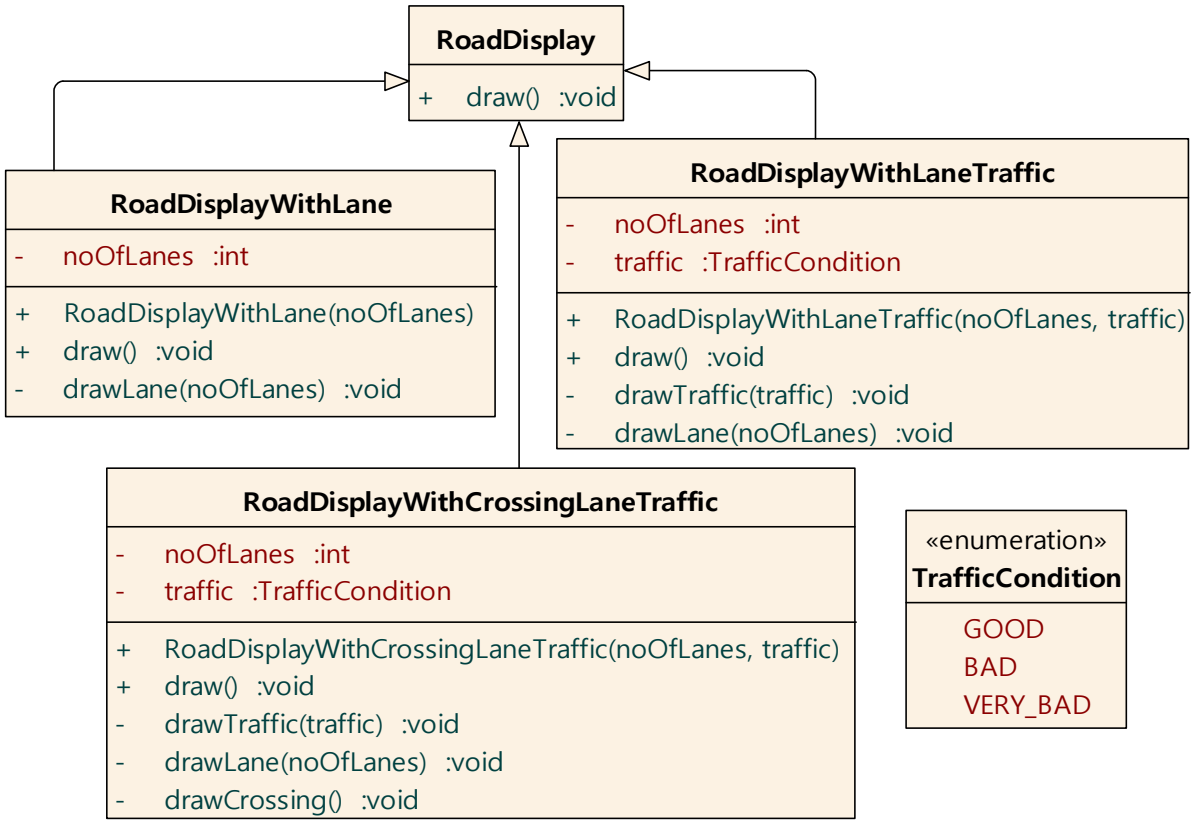


Road Display

- User can select various display strategies



Subclassing Approach



Subclassing Approach – Source Codes

```
public class Client {
  @Test
  public void test() {
    RoadDisplay road = new RoadDisplay() ;
    road.draw() ;

    RoadDisplay roadWithLane
      new RoadDisplayWithLane(3) ;
    roadWithLane.draw() ;

    RoadDisplay roadWithLaneAndTraffic =
      new RoadDisplayWithLaneTraffic(3, TrafficCondition.GOOD) ;
    roadWithLaneAndTraffic.draw() ;

    RoadDisplay roadWithCrossingAndTrafficAndLane =
      new RoadDisplayWithCrossingLaneTraffic(4,
        TrafficCondition.VERY_BAD) ;
    roadWithCrossingAndTrafficAndLane.draw() ;
  }
}
```

```
도로 기본 표시
도로 기본 표시
  차선 수 3
도로 기본 표시
  교통량: GOOD
  차선 수 3
도로 기본 표시
  교통량: VERY_BAD
  차선 수 4
  교차로 표시
```

Subclassing Approach – Source Codes

```
public class RoadDisplay {
  public void draw() {
    System.out.println("도로 기본 표시") ;
  }
}
```

```
public class RoadDisplayWithLane extends RoadDisplay {
  private int noOfLanes ;
  public RoadDisplayWithLane(int noOfLanes) {
    this.noOfLanes = noOfLanes;
  }
  public void draw() {
    super.draw();
    drawLane(noOfLanes);
  }
  private void drawLane(int noOfLanes) {
    System.out.println("₩t차선 수 " + noOfLanes) ;
  }
}
```


Subclassing Approach – Source Codes

```
public class RoadDisplayWithLaneTraffic extends RoadDisplay {
  private int noOfLanes ;
  private TrafficCondition traffic;

  public RoadDisplayWithLaneTraffic(int noOfLanes, TrafficCondition traffic) {
    this.noOfLanes = noOfLanes;
    this.traffic = traffic;
  }
  public void draw() {
    super.draw();
    drawTraffic(traffic);
    drawLane(noOfLanes);
  }
  private void drawTraffic(TrafficCondition traffic) {
    System.out.println("₩t교통량: " + traffic) ;
  }
  private void drawLane(int noOfLanes) {
    System.out.println("₩t차선 수 " + noOfLanes) ;
  }
}
```

Subclassing Approach – Source Codes

```
public class RoadDisplayWithCrossingLaneTraffic extends RoadDisplay {
  private int noOfLanes ;
  private TrafficCondition traffic;
  public RoadDisplayWithCrossingLaneTraffic(int noOfLanes,
    TrafficCondition traffic) {
    this.noOfLanes = noOfLanes;
    this.traffic = traffic;
  }
  public void draw() {
    super.draw();
    drawTraffic(traffic);
    drawLane(noOfLanes);
    drawCrossing() ;
  }
  private void drawTraffic(TrafficCondition traffic) {
    System.out.println("₩t교통량: " + traffic) ;
  }
  private void drawLane(int noOfLanes) {
    System.out.println("₩t차선 수 " + noOfLanes) ;
  }
  private void drawCrossing() { System.out.println("₩t교차로 표시") ; }
}
```


Problems with Subclassing Approach

- Class explosion to support every combination

Combinations	Lane	Traffic	Crossing	Class
1				**C0**
2	√			**C1**
3		√		**C2**
4			√	**C3**
5	√	√		**C12**
6	√		√	**C13**
7		√	√	**C23**
8	√	√	√	**C123**

Solution – Decorator Pattern - Design

Source Codes: Display, RoadDisplay, DisplayDecorator

Source Codes: LaneDecorator

Source Codes: TrafficDecorator

Source Codes: CrossingDecorator

Source Codes: Client