

아키텍팅, 원점에서 다시 보기

1. 출발 - 대상의 본질

시작은 가볍게...

구구단(*TimesTable*)을 출력하세요. →

2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18

3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27

4 x 1 = 4



아주 간단하군요 !!

```
public void showTable() {  
    //  
    System.out.println(" Times table.");  
    System.out.println(" -----");  
  
    for (int i = 2; i<=9; i++) {  
        for (int j = 1; j <=9; j++) {  
            System.out.format(" %2d x %d = %2d%n", i, j, (i*j));  
        }  
  
        System.out.println(" -----");  
    }  
}
```

배려가 있는 코딩 !!

```
public void showTable() {  
    //  
    System.out.println("\n Times table.");  
    System.out.println(" -----");  
  
    for (int leftNumber = 2; leftNumber<=9; leftNumber++) {  
        for (int rightNumber = 1; rightNumber <=9; rightNumber++) {  
            System.out.format(" %2d x %d = %2d%n",  
                               leftNumber,  
                               rightNumber,  
                               (leftNumber*rightNumber));  
        }  
  
        System.out.println(" -----");  
    }  
}
```

사랑이 변하듯 요건도 변하죠...

변경해 주세요. →

$2 \times 1 = 2$

$2 \times 2 = 4$

$2 \times 3 = 6$

$2 \times 4 = 8$

$2 \times 5 = 10$

$2 \times 6 = 12$

$2 \times 7 = 14$

$2 \times 8 = 16$

$2 \times 9 = 18$

$3 \times 1 = 3$

$3 \times 2 = 6$

$3 \times 3 = 9$

$3 \times 4 = 12$

$3 \times 5 = 15$

$3 \times 6 = 18$

$3 \times 7 = 21$

$3 \times 8 = 24$

$3 \times 9 = 27$

$4 \times 1 = 4$

$4 \times 2 = 8$

$4 \times 3 = 12$

$4 \times 4 = 16$

$4 \times 5 = 20$

$4 \times 6 = 24$

$4 \times 7 = 28$

$4 \times 8 = 32$

$4 \times 9 = 36$

$5 \times 1 = 5$

$5 \times 2 = 10$

$5 \times 3 = 15$

$5 \times 4 = 20$

$5 \times 5 = 25$

$5 \times 6 = 30$

$5 \times 7 = 35$

$5 \times 8 = 40$

$5 \times 9 = 45$

$6 \times 1 = 6$

$6 \times 2 = 12$

$6 \times 3 = 18$

$6 \times 4 = 24$

$6 \times 5 = 30$

$6 \times 6 = 36$

$6 \times 7 = 42$

$6 \times 8 = 48$

$6 \times 9 = 54$

$7 \times 1 = 7$

$7 \times 2 = 14$

$7 \times 3 = 21$

$7 \times 4 = 28$

$7 \times 5 = 35$

$7 \times 6 = 42$

$7 \times 7 = 49$

$7 \times 8 = 56$

$7 \times 9 = 63$

$8 \times 1 = 8$

$8 \times 2 = 16$

$8 \times 3 = 24$

$8 \times 4 = 32$

$8 \times 5 = 40$

$8 \times 6 = 48$

$8 \times 7 = 56$

$8 \times 8 = 64$

$8 \times 9 = 72$

$9 \times 1 = 9$

$9 \times 2 = 18$

$9 \times 3 = 27$

$9 \times 4 = 36$

$9 \times 5 = 45$

$9 \times 6 = 54$

$9 \times 7 = 63$

$9 \times 8 = 72$

$9 \times 9 = 81$

이것도 간단하긴 한데...

```
private static void printTimesTable(int columnCount) {  
    //  
    int loopCount = (8 / columnCount) + 1;  
  
    int leftNumberOffset = 2;  
    for(int i=0; i<loopCount; i++) {  
        for(int rightNumber=1; rightNumber<=9; rightNumber++) {  
            for(int count = 0; count < columnCount; count++) {  
                int leftNumber = leftNumberOffset + count;  
                if (leftNumber > 9) {  
                    break;  
                }  
                printUnit(leftNumber, rightNumber);  
            }  
            System.out.println("");  
        }  
        leftNumberOffset += columnCount;  
        System.out.println("");  
    }  
}  
  
private static void printUnit(int leftNumber, int rightNumber) {  
    System.out.print("\t" + leftNumber + " x " + rightNumber + " = " + (leftNumber*rightNumber));  
}
```

잠시만, 이것도 하나 추가해 주세요.

죄송한데요... →

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

9	18	27	36	45	54	63	72	81
8	16	24	32	40	48	56	64	72
7	14	21	28	35	42	49	56	63
6	12	18	24	30	36	42	48	54
5	10	15	20	25	30	35	40	45
4	8	12	16	20	24	28	32	36
3	6	9	12	15	18	21	24	27
2	4	6	8	10	12	14	16	18
1	2	3	4	5	6	7	8	9

흠, 이건 이야기가 다른데...

```
public void showLineTable(SortOrder sortOrder) {  
    //  
    int leftNumber = 1;    /// 1 또는 2  
    int addNumber = 1;  
  
    if(SortOrder.Descending.equals(sortOrder)) {  
        leftNumber = MaxTimes;  
        addNumber = -1;  
    }  
    System.out.println();  
  
    while(true) {  
        System.out.print(" ");  
        for(int rightNumber = 1; rightNumber <= MaxTimes; rightNumber++) {  
            System.out.print(getColumn(leftNumber, rightNumber));  
        }  
        System.out.println();  
  
        leftNumber += addNumber;  
        if(leftNumber < 2 || leftNumber > MaxTimes) {  
            break;  
        }  
    }  
    System.out.println();  
}
```

Cyclometric Complexity

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
-----
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
-----
4 x 1 = 4
```

```
public void showTable() {
    //
    System.out.println("\n Times table.");
    System.out.println(" -----");

    for (int leftNumber = 2; leftNumber <= 9; leftNumber++) {
        for (int rightNumber = 1; rightNumber <= 9; rightNumber++) {
            System.out.format(" %2d x %d = %2d\n",
                              leftNumber,
                              rightNumber,
                              (leftNumber * rightNumber));
        }

        System.out.println(" -----");
    }
}
```

3

5

12

```
public void showLineTable(SortOrder sortOrder) {
    //
    int leftNumber = 1;    /// 1 또는 2
    int addNumber = 1;

    if (SortOrder.Descending.equals(sortOrder)) {
        leftNumber = MaxTimes;
        addNumber = -1;
    }

    System.out.println();

    while (true) {
        System.out.print(" ");
        for (int rightNumber = 1; rightNumber <= MaxTimes; rightNumber++) {
            System.out.print(getColumn(leftNumber, rightNumber));
        }

        System.out.println();

        leftNumber += addNumber;
        if (leftNumber < 2 || leftNumber > MaxTimes) {
            break;
        }
    }

    System.out.println();
}
```

```
private static void printTimesTable(int columnCount) {
    //
    int loopCount = (8 / columnCount) + 1;

    int leftNumberOffset = 2;
    for (int i = 0; i < loopCount; i++) {
        for (int rightNumber = 1; rightNumber <= 9; rightNumber++) {
            for (int count = 0; count < columnCount; count++) {
                int leftNumber = leftNumberOffset;
                if (leftNumber > 9) {
                    break;
                }
                printInit(leftNumber, rightNumber);
                System.out.println("");
                leftNumberOffset += columnCount;
                System.out.println("");
            }
        }
    }
}
```

2 x 1 = 2	3 x 1 = 3	4 x 1 = 4	5 x 1 = 5
2 x 2 = 4	3 x 2 = 6	4 x 2 = 8	5 x 2 = 10
2 x 3 = 6	3 x 3 = 9	4 x 3 = 12	5 x 3 = 15
2 x 4 = 8	3 x 4 = 12	4 x 4 = 16	5 x 4 = 20
2 x 5 = 10	3 x 5 = 15	4 x 5 = 20	5 x 5 = 25
2 x 6 = 12	3 x 6 = 18	4 x 6 = 24	5 x 6 = 30
2 x 7 = 14	3 x 7 = 21	4 x 7 = 28	5 x 7 = 35
2 x 8 = 16	3 x 8 = 24	4 x 8 = 32	5 x 8 = 40
2 x 9 = 18	3 x 9 = 27	4 x 9 = 36	5 x 9 = 45
6 x 1 = 6	7 x 1 = 7	8 x 1 = 8	9 x 1 = 9
6 x 2 = 12	7 x 2 = 14	8 x 2 = 16	9 x 2 = 18
6 x 3 = 18	7 x 3 = 21	8 x 3 = 24	9 x 3 = 27
6 x 4 = 24	7 x 4 = 28	8 x 4 = 32	9 x 4 = 36
6 x 5 = 30	7 x 5 = 35	8 x 5 = 40	9 x 5 = 45
6 x 6 = 36	7 x 6 = 42	8 x 6 = 48	9 x 6 = 54
6 x 7 = 42	7 x 7 = 49	8 x 7 = 56	9 x 7 = 63
6 x 8 = 48	7 x 8 = 56	8 x 8 = 64	9 x 8 = 72
6 x 9 = 54	7 x 9 = 63	8 x 9 = 72	9 x 9 = 81

구구단(Times table)은 어디에...

```
private static void printTimesTable(int columnCount) {  
    //  
    int loopCount = (8 / columnCount) + 1;  
  
    int leftNumberOffset = 2;  
    for(int i=0; i<loopCount; i++) {  
        for(int rightNumber=1; rightNumber<=9; rightNumber++) {  
            for(int count = 0; count < columnCount; count++) {  
                int leftNumber = leftNumberOffset + count;  
                if (leftNumber > 9) {  
                    break;  
                }  
                printUnit(leftNumber, rightNumber);  
            }  
            System.out.println("");  
        }  
        leftNumberOffset += columnCount;  
        System.out.println("");  
    }  
}  
  
private static void printUnit(int leftNumber, int rightNumber) {  
    System.out.print("\t" + leftNumber + " x " + rightNumber + " = " + (leftNumber*rightNumber));  
}
```

구구단(Times table)을 구성하는 개념들

$2 \times 1 = 2$
 $2 \times 2 = 4$
 $2 \times 3 = 6$
 $2 \times 4 = 8$
 $2 \times 5 = 10$
 $2 \times 6 = 12$
 $2 \times 7 = 14$
 $2 \times 8 = 16$
 $2 \times 9 = 18$

$3 \times 1 = 3$
 $3 \times 2 = 6$
 $3 \times 3 = 9$
 $3 \times 4 = 12$
 $3 \times 5 = 15$
 $3 \times 6 = 18$
 $3 \times 7 = 21$
 $3 \times 8 = 24$
 $3 \times 9 = 27$

$4 \times 1 = 4$
 $4 \times 2 = 8$
 $4 \times 3 = 12$
 $4 \times 4 = 16$
 $4 \times 5 = 20$
 $4 \times 6 = 24$
 $4 \times 7 = 28$
 $4 \times 8 = 32$
 $4 \times 9 = 36$

$5 \times 1 = 5$
 $5 \times 2 = 10$
 $5 \times 3 = 15$
 $5 \times 4 = 20$
 $5 \times 5 = 25$
 $5 \times 6 = 30$
 $5 \times 7 = 35$
 $5 \times 8 = 40$
 $5 \times 9 = 45$

5 단
(5 times table)

$6 \times 1 = 6$
 $6 \times 2 = 12$
 $6 \times 3 = 18$
 $6 \times 4 = 24$
 $6 \times 5 = 30$
 $6 \times 6 = 36$
 $6 \times 7 = 42$
 $6 \times 8 = 48$
 $6 \times 9 = 54$

$7 \times 1 = 7$
 $7 \times 2 = 14$
 $7 \times 3 = 21$
 $7 \times 4 = 28$
 $7 \times 5 = 35$
 $7 \times 6 = 42$
 $7 \times 7 = 49$
 $7 \times 8 = 56$
 $7 \times 9 = 63$

$8 \times 1 = 8$
 $8 \times 2 = 16$
 $8 \times 3 = 24$
 $8 \times 4 = 32$
 $8 \times 5 = 40$
 $8 \times 6 = 48$
 $8 \times 7 = 56$
 $8 \times 8 = 64$
 $8 \times 9 = 72$

$9 \times 1 = 9$
 $9 \times 2 = 18$
 $9 \times 3 = 27$
 $9 \times 4 = 36$
 $9 \times 5 = 45$
 $9 \times 6 = 54$
 $9 \times 7 = 63$
 $9 \times 8 = 72$
 $9 \times 9 = 81$

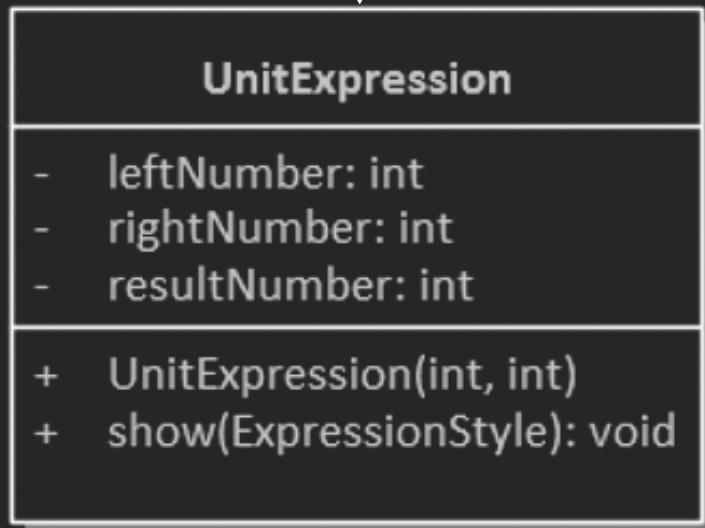
구구단
(Times table)

산술식
(Arithmetic Expression)

서식
(Style)

도메인 모델링 - 개념 1 : 산술식(Arithmetic Expression)

산술식 : $5 \times 4 = 20$



```
public class UnitExpression {  
    //  
    private int leftNumber;  
    private int rightNumber;  
    private int resultNumber;  
  
    public UnitExpression(int leftNumber, int rightNumber) {  
        //  
        this.leftNumber = leftNumber;  
        this.rightNumber = rightNumber;  
        this.resultNumber = leftNumber * rightNumber;  
    }  
  
    public String toString() {  
        //  
        return leftNumber + " * " + rightNumber + " = " + resultNumber;  
    }  
  
    public void show(ExpressionStyle style) {  
        //  
        System.out.format(style.formatStrLn(), leftNumber, rightNumber, resultNumber);  
    }  
  
    public static UnitExpression getSample() {  
        return new UnitExpression(5, 4);  
    }  
}
```


도메인 모델링 - 개념 2 : 각 단(Each Multiplication Table)

5단

5 x 1 = 5 5 x 6 = 30
5 x 2 = 10 5 x 7 = 35
5 x 3 = 15 5 x 8 = 40
5 x 4 = 20 5 x 9 = 45
5 x 5 = 25



UnitTable

- leftNumber: int
- style: ExpressionStyle
- expressions: List<UnitExpression>

- + UnitTable(int, ExpressionStyle)
- + show(): void
- + getFormattedExpression(int): String

```
public class UnitTable {  
    //  
    private int leftNumber;  
    private ExpressionStyle style;  
    private List<UnitExpression> expressions;
```

```
    public UnitTable(int leftNumber) {
```

```
        //
```

```
        this(leftNumber, ExpressionStyle.InEnglish);
```

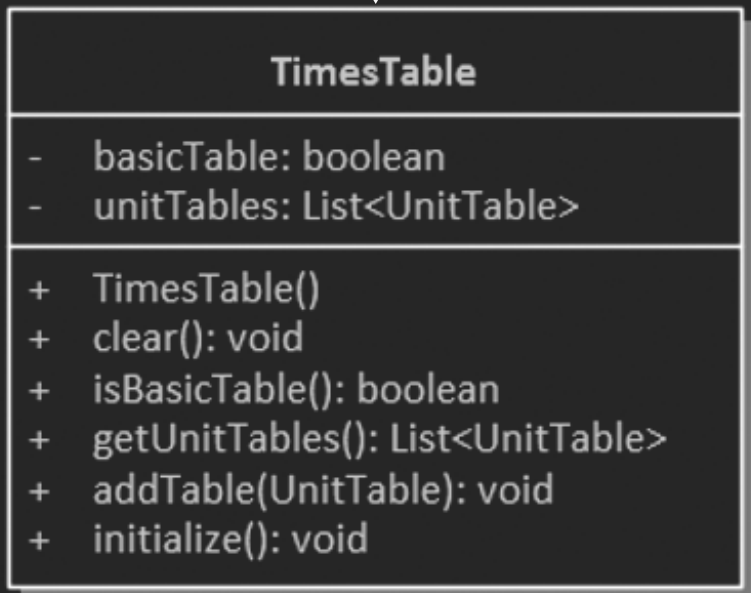
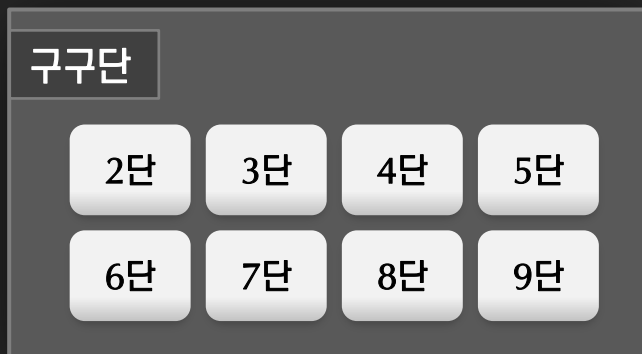
```
    }
```

```
    public UnitTable(int leftNumber, ExpressionStyle style) {
```

```
        public String toString() {
```

```
            private void init() {
```

도메인 모델링 - 개념 3 : 구구단(Times table)



```
public class TimesTable {  
    //  
    private boolean basicTable;  
    private List<UnitTable> unitTables;  
  
    public TimesTable() {  
        //  
        this.basicTable = false;  
        this.unitTables = new ArrayList<>();  
    }  
  
    public void clear() {  
        this.unitTables.clear();  
    }  
  
    public boolean isBasicTable() {  
        return basicTable;  
    }  
  
    public List<UnitTable> getUnitTables() {  
        //  
        if (basicTable) {
```

도메인 모델링 - 개념 4 : 서식(Style for the expression)

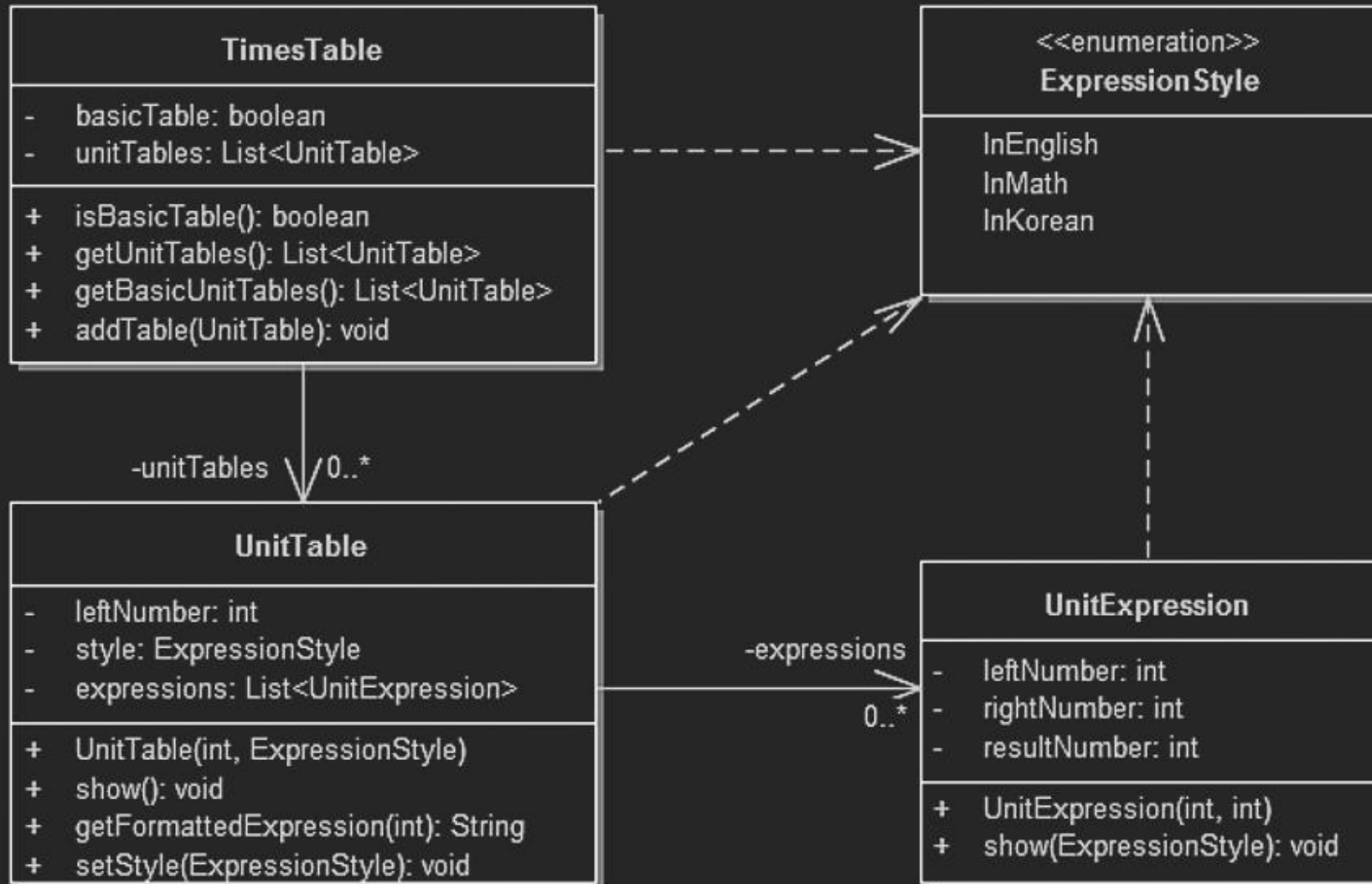
5 times 4 is 20

5 x 4 = 20

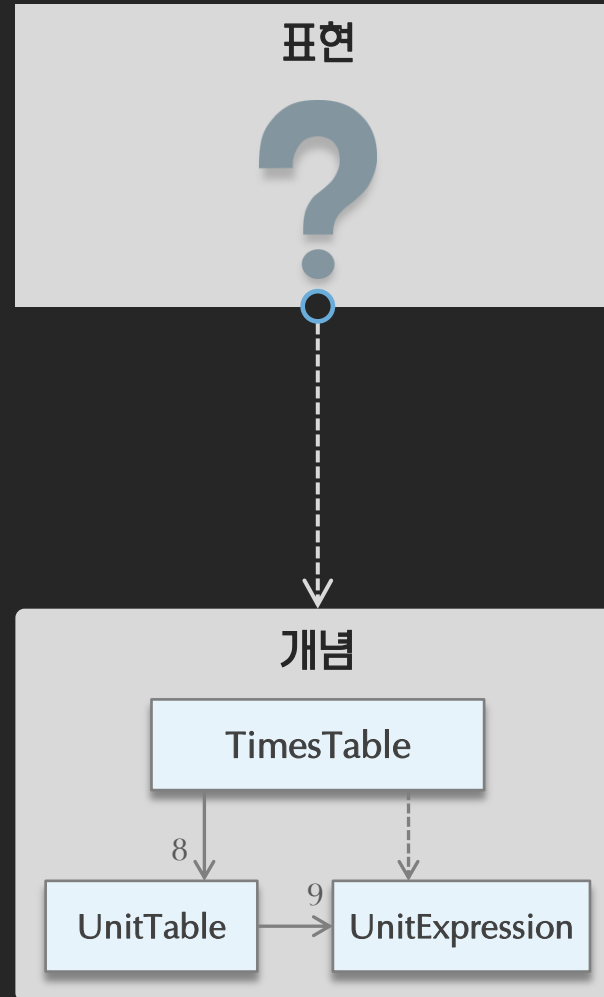
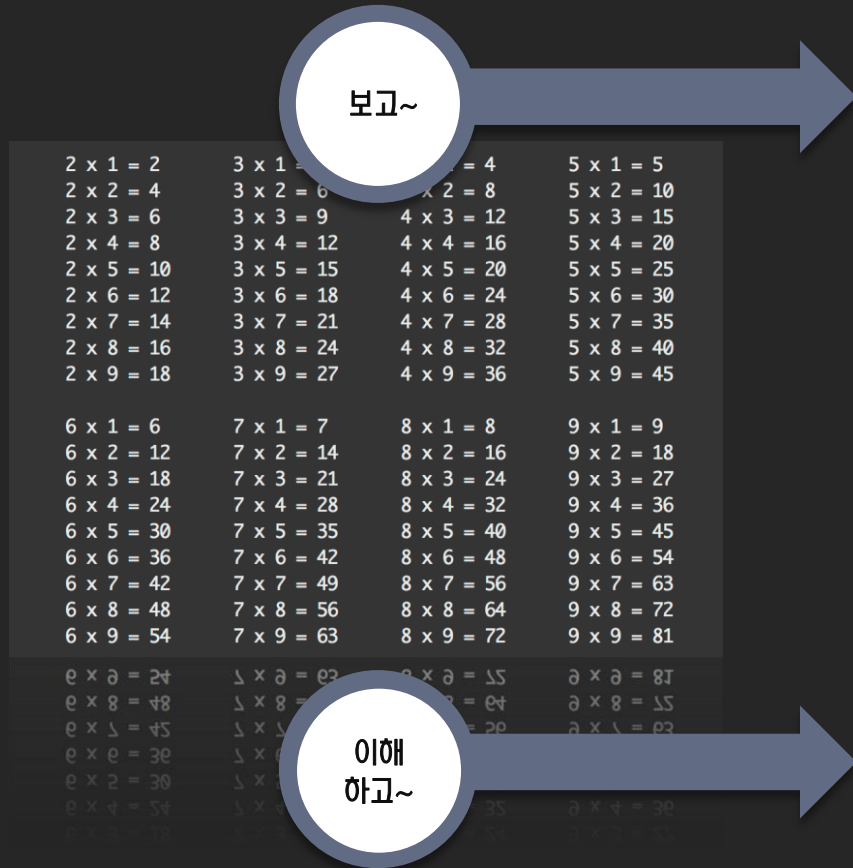
5 곱하기 4 는 20

```
public enum ExpressionStyle {  
    //  
    InEnglish(" %d times %d is %2d"),  
    InJavaCode(" %d * %d = %2d"),  
    InMath(" %d x %d = %2d"),  
    InKorean(" %d 곱하기 %d 는 %2d");  
  
    private String formatStr;  
  
    private ExpressionStyle(String style) {  
        this.formatStr = style;  
    }  
}
```


대상(TimesTable)의 본질



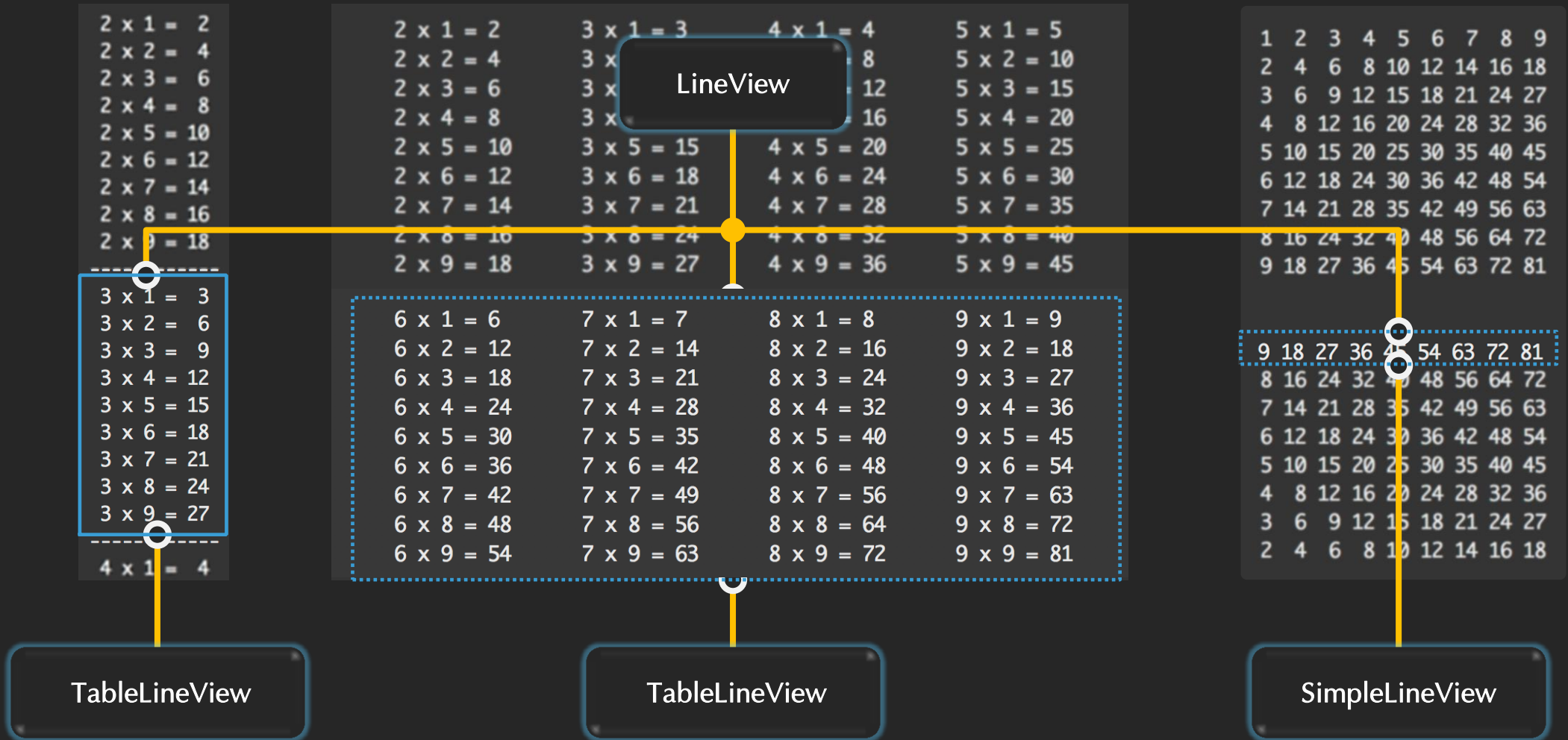
본질은 원래 추상적...



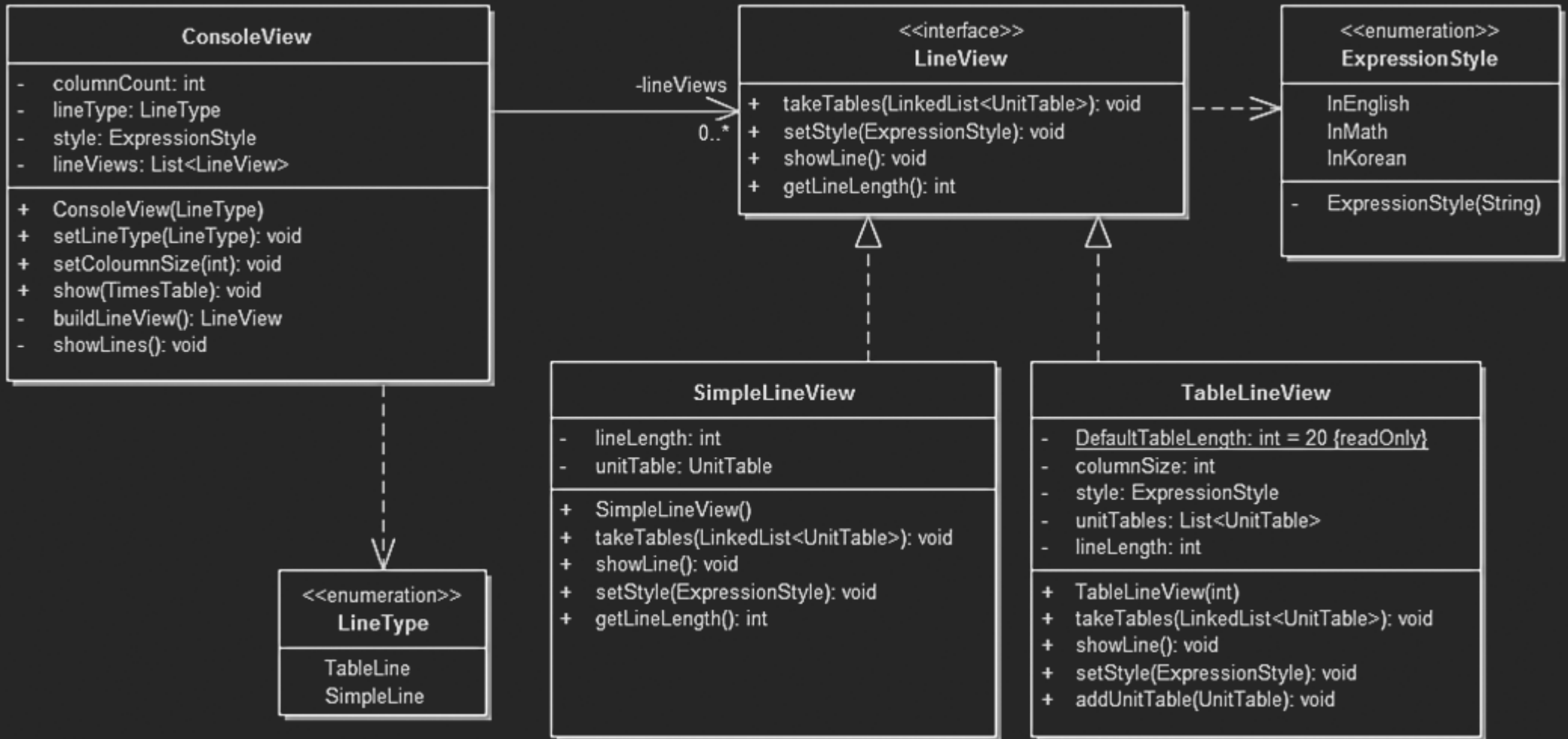
← 표현에도 개념이 있다...

← 도메인 모델

추상화(Abstraction)로 변화하는 대상에 질서를...



표현(Representation)을 위한 개념들



SOLID – SRP, OCP, LSP, DIP, except for ISP

개념은 안정적이고, 표현은 유동적이다.

```
public void showTableO {
    //
    System.out.println("\n Times table.");
    System.out.println("-----");
    for (int leftNumber = 2; leftNumber <= 9; leftNumber++) {
        for (int rightNumber = 2; rightNumber <= 9; rightNumber++) {
            System.out.format("%d * %d = %d\n",
                leftNumber, rightNumber,
                (leftNumber * rightNumber));
        }
        System.out.println("-----");
    }
}
```

1열

TimesTable.java

```
private static void printTimesTable(int columnCount) {
    //
    int loopCount = (5 / columnCount) + 1;
    int leftNumberOffset = 2;
    for (int i=0; i<loopCount; i++) {
        for (int rightNumber=1; rightNumber<= columnCount; rightNumber++) {
            int count = 0; count < columnCount;
            int leftNumber = leftNumberOffset;
            if (leftNumber > 9) {
                break;
            }
            printUnit(leftNumber, rightNumber);
            System.out.println("");
            leftNumberOffset += columnCount;
            System.out.println("");
        }
    }
}

private static void printUnit(int leftNumber, int rightNumber) {
    System.out.print("\t" + leftNumber + " * " + rightNumber
        + "\t");
}

private static void printTimesTable(int columnCount) {
    //
    int loopCount = (5 / columnCount) + 1;
    int leftNumberOffset = 2;
    for (int i=0; i<loopCount; i++) {
        for (int rightNumber=1; rightNumber<= columnCount; rightNumber++) {
            int count = 0; count < columnCount;
            int leftNumber = leftNumberOffset;
            if (leftNumber > 9) {
                break;
            }
            printUnit(leftNumber, rightNumber);
            System.out.println("");
            leftNumberOffset += columnCount;
            System.out.println("");
        }
    }
}

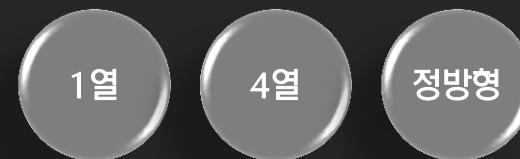
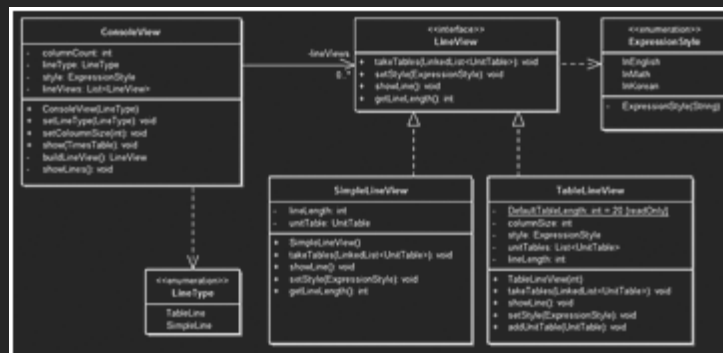
private static void printUnit(int leftNumber, int rightNumber) {
    System.out.print("\t" + leftNumber + " * " + rightNumber
        + "\t");
}
```

4열

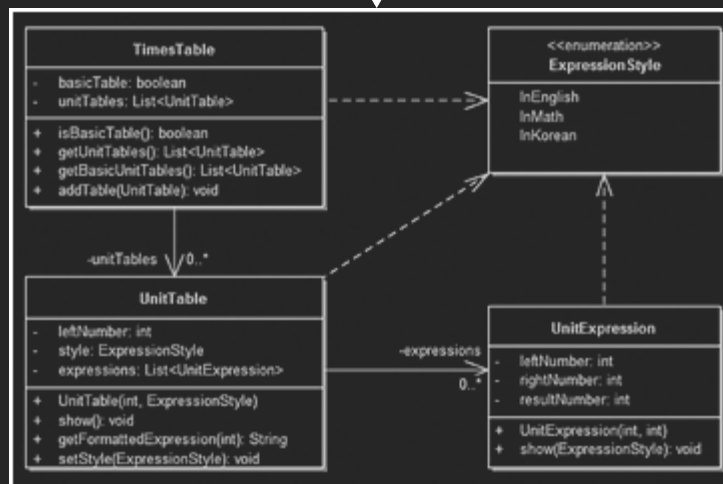
TimesTable.java

정방형

TimesTable.java



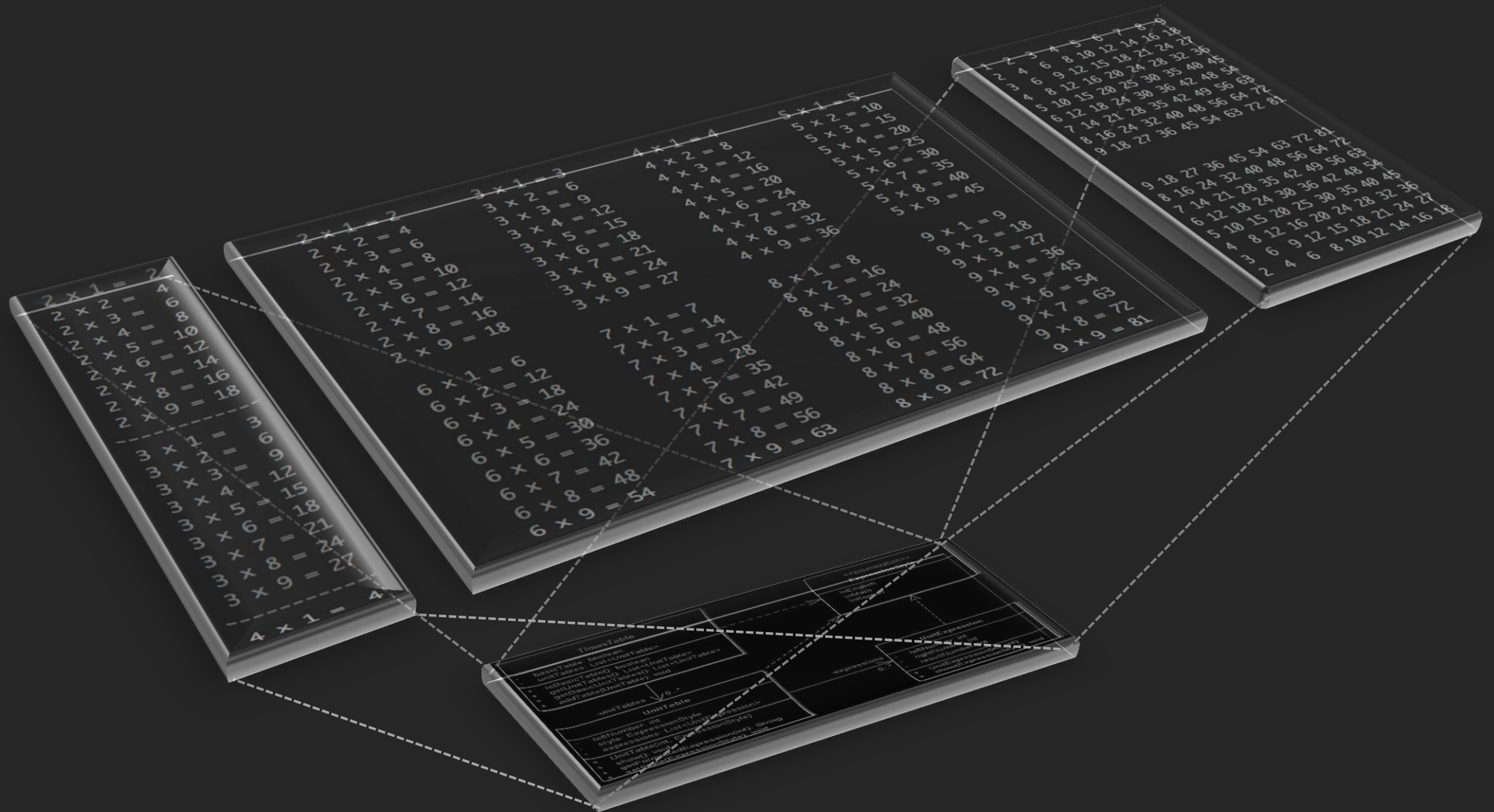
← 표현 (representation)
← Dynamic



← 도메인 (로직과 개념)
← Stable

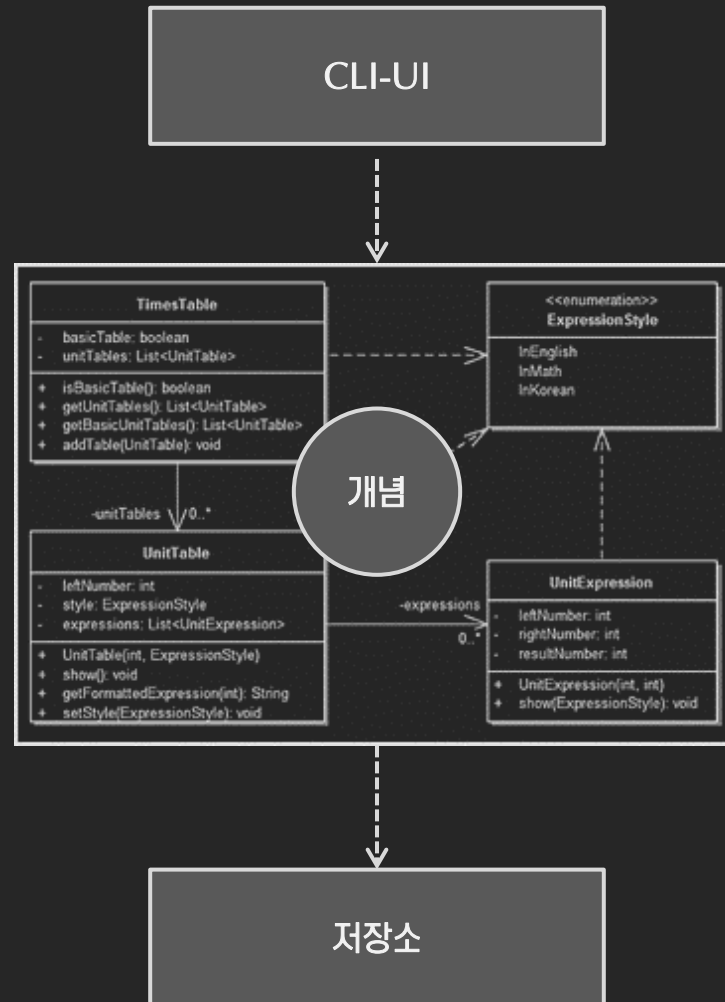
```
public TimesTableDemo() {  
    //  
    this.timesTable = new TimesTable();  
    this.timesTable.initialize();  
}  
  
public void showTableLineDemo() {  
    //  
    ConsoleView consoleView = new ConsoleView(LineType.TableLine);  
    consoleView.setColoumnSize(4);  
    consoleView.setStyle(ExpressionStyle.InMath);  
    consoleView.show(this.timesTable);  
    consoleView.showLineSeparator();  
}  
  
public void showSimpleLineDemo() {  
    //  
    ConsoleView consoleView = new ConsoleView(LineType.SimpleLine);  
    consoleView.setColoumnSize(4); // ignored  
    consoleView.show(this.timesTable);  
    consoleView.showLineSeparator();  
}  
  
public static void main(String[] args) {  
    //  
    TimesTableDemo demo = new TimesTableDemo();  
    demo.showTableLineDemo();  
    demo.showSimpleLineDemo();  
}
```


본질은 다양한 형상으로 그 모습을 드러낸다.

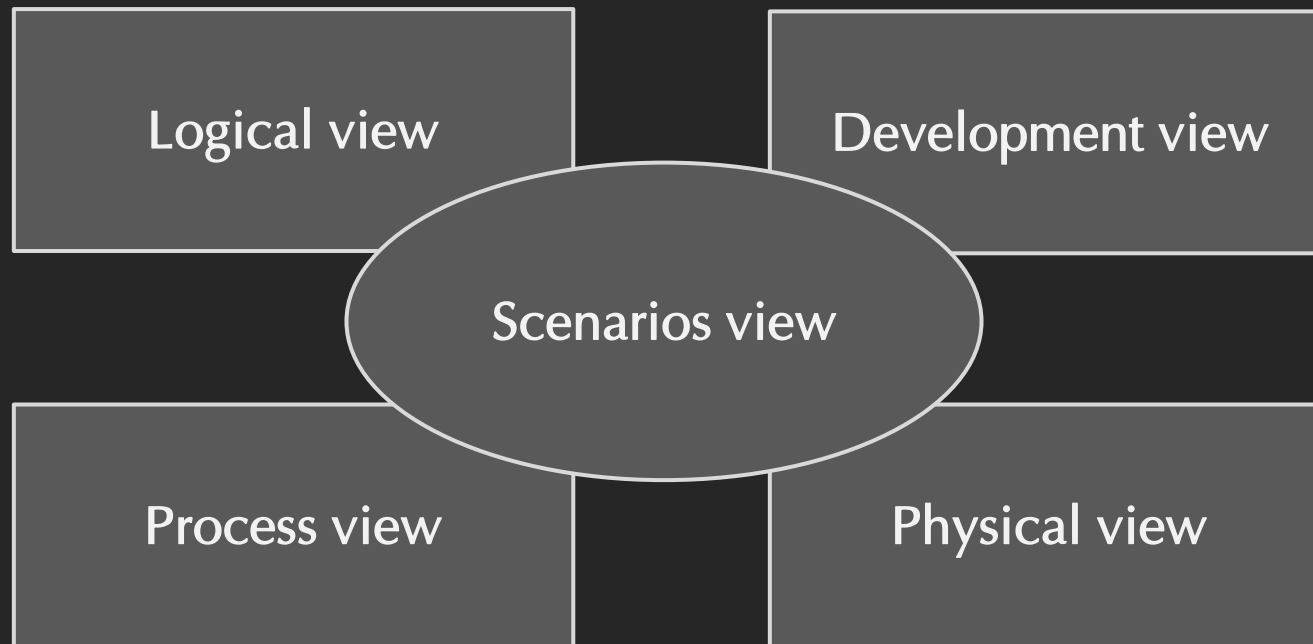


2. 확장 - 구조와 뷰

시스템으로써 틀을 갖추려면...



4 + 1 Views (as an architecture framework)



1995, Philippe Kruchten

3 Views (from CMU SEI)

Module view

C&C view

Allocation view

Deployment style

Implementation style

Work assignment style

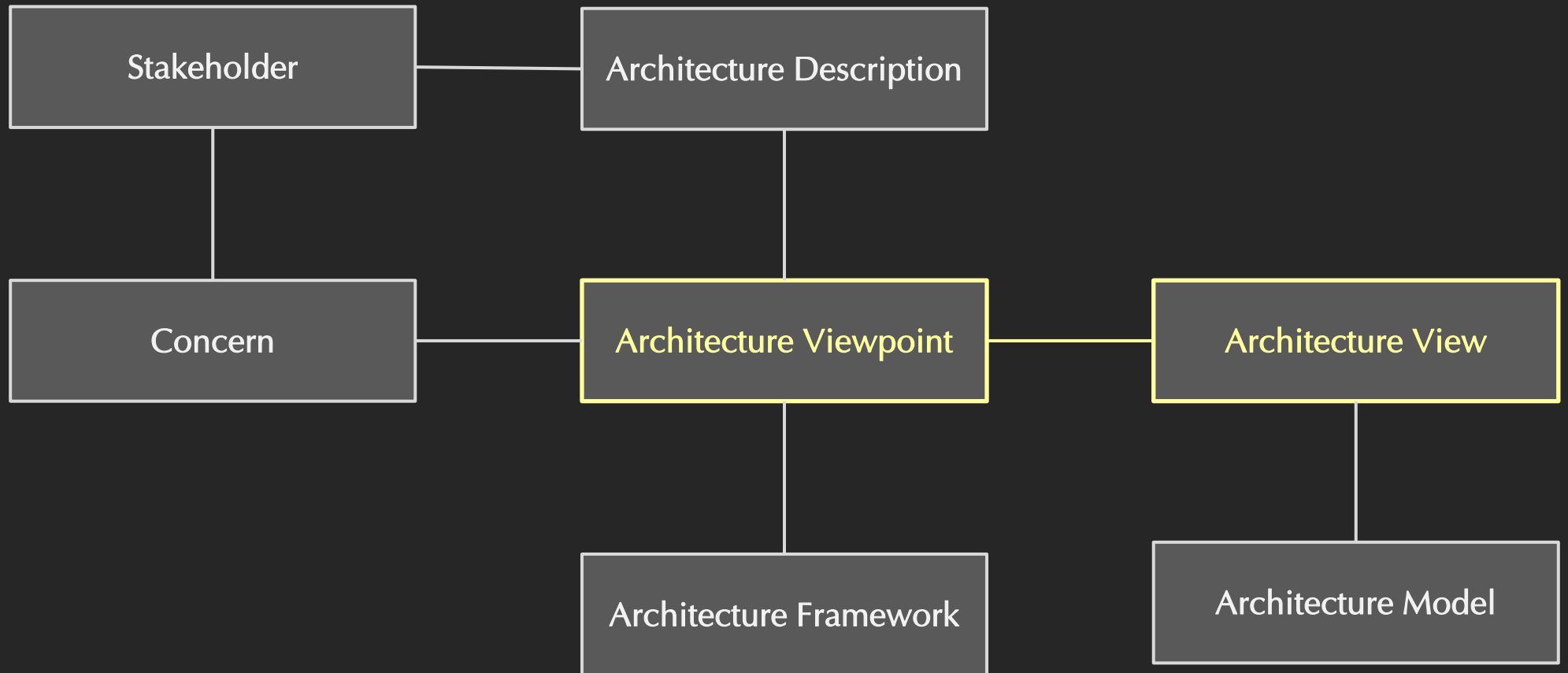
3 Perspectives (from Λ CDM by Lattanze)

Static perspective

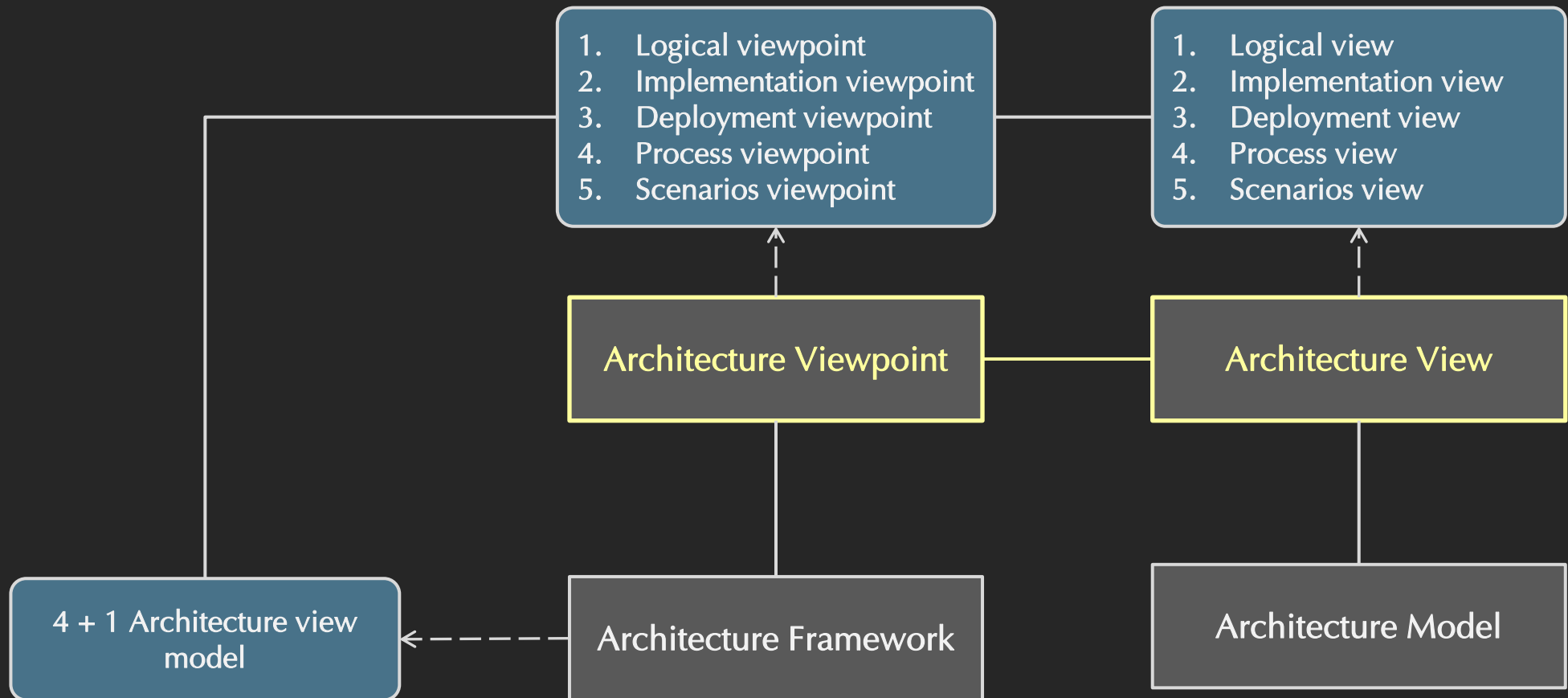
Dynamic perspective

Physical perspective

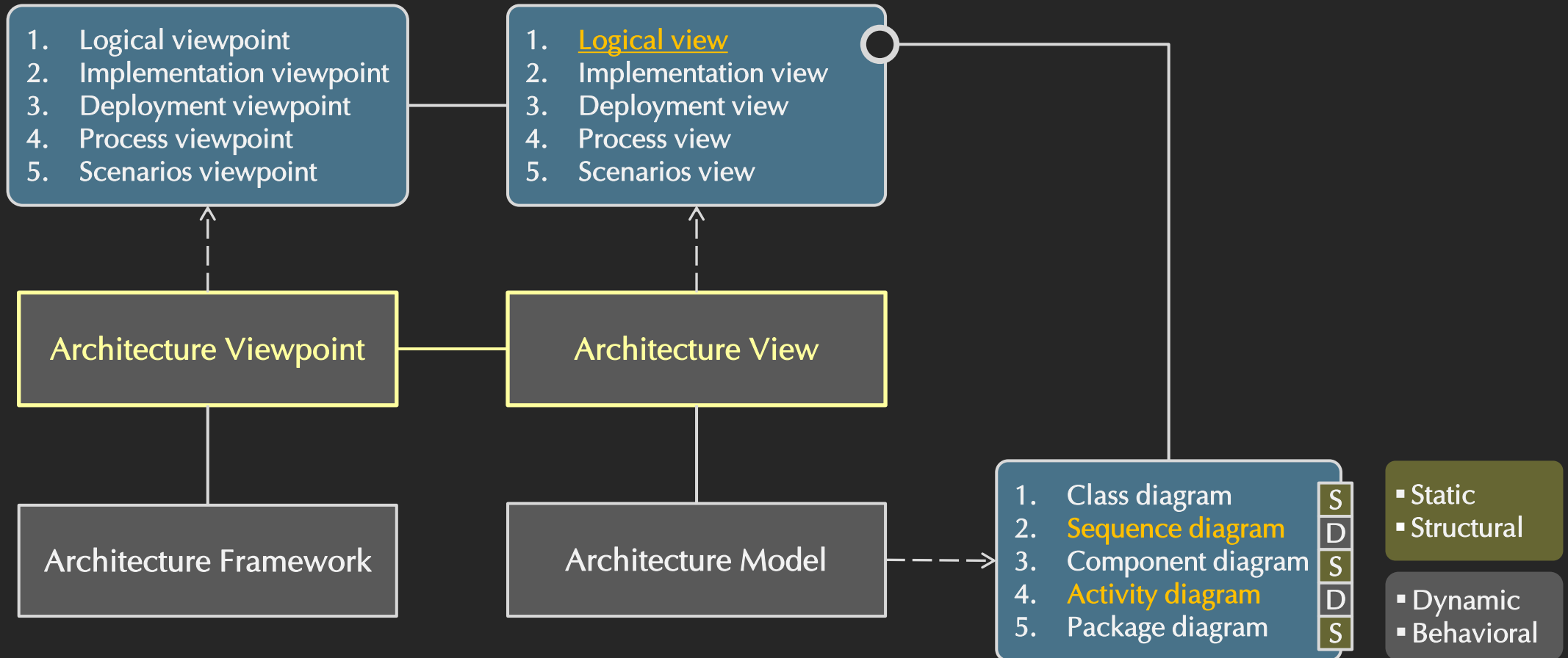
ISO/IEC/IEEE 42010:2011



Architecture framework – Viewpoint : View



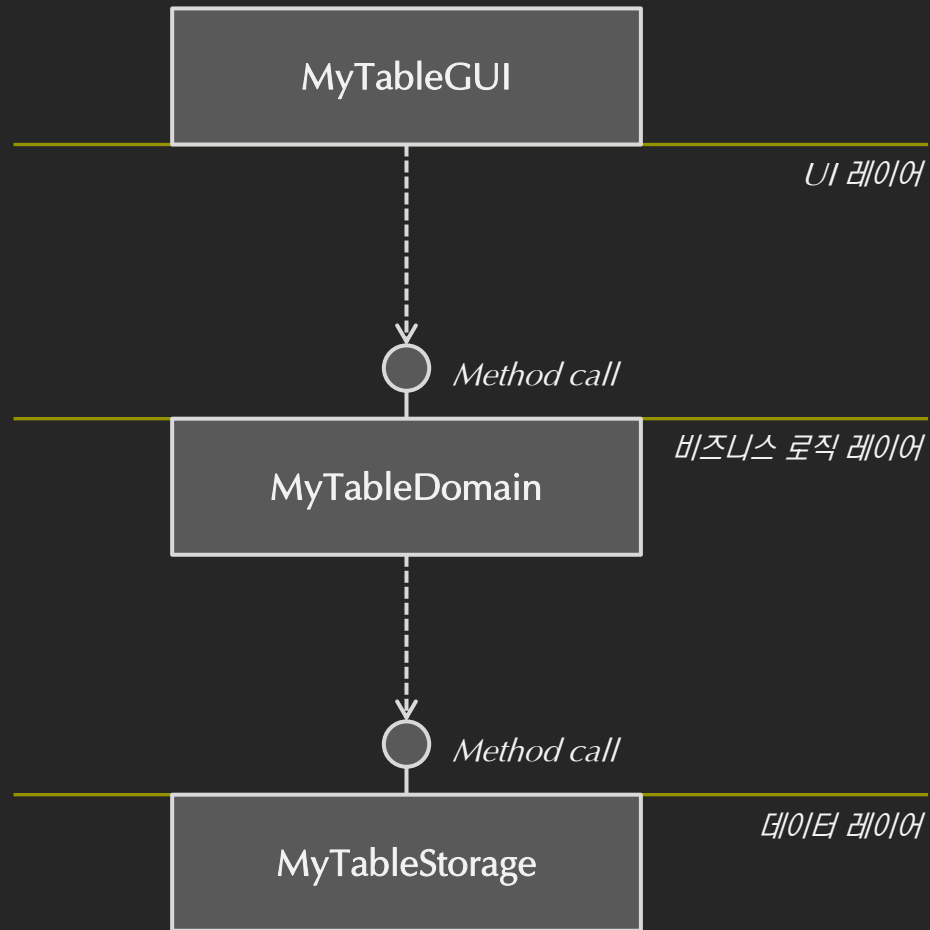
View : Model(static, dynamic)



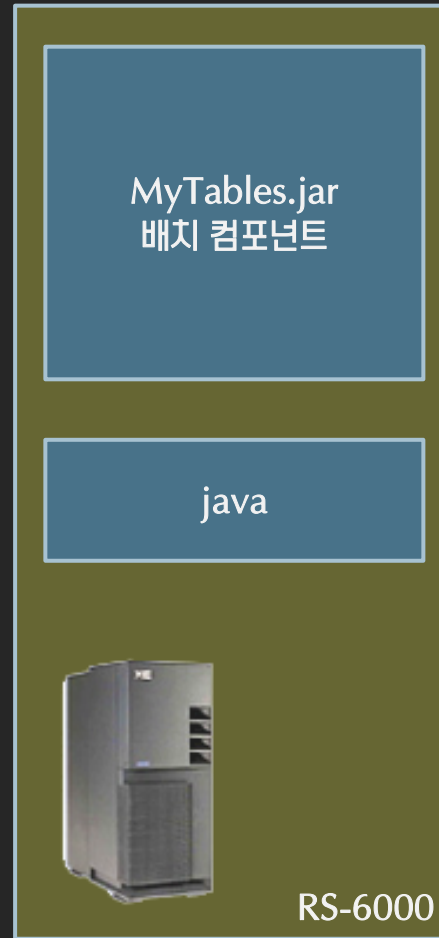
SDLC와 Views

		Requirement	Analysis	Design	Implements	Deployment	Runtime
Functional perspective		기능 요건	도메인 모델/UI 모델/데이터 모델 <div>분석 수준 모델</div> <div>설계 수준 모델</div>		비즈니스 컴포넌트 프로그램 또는 프로그램	배포 비즈니스 컴포넌트, 라이브러리	프로그램 실행
Non-functional perspective		비기능 요건	아키텍처 모델 <div>패턴/스타일</div> <div>반복적인 분해 모델</div>		<div>아키텍처 요소 프로그램</div> <div>개발 모델</div>	배포 아키텍처 요소 배포 모델	프로그램 실행 런타임 모델
아키텍처 프레임워크	4 + 1 뷰	Use Case view (or Scenarios view)	Logical view	Development view (or Implementation view)		Physical view (or Deployment view)	Process view (or Runtime view)
	CMU SEI	N/A	Module view	Allocation view		Allocation view	C&C view
	ACDM	N/A	Static perspective			Physical perspective	Dynamic perspective

독립형(stand-alone) 시스템



Logical View (모듈/설계 컴포넌트)




Deployment View

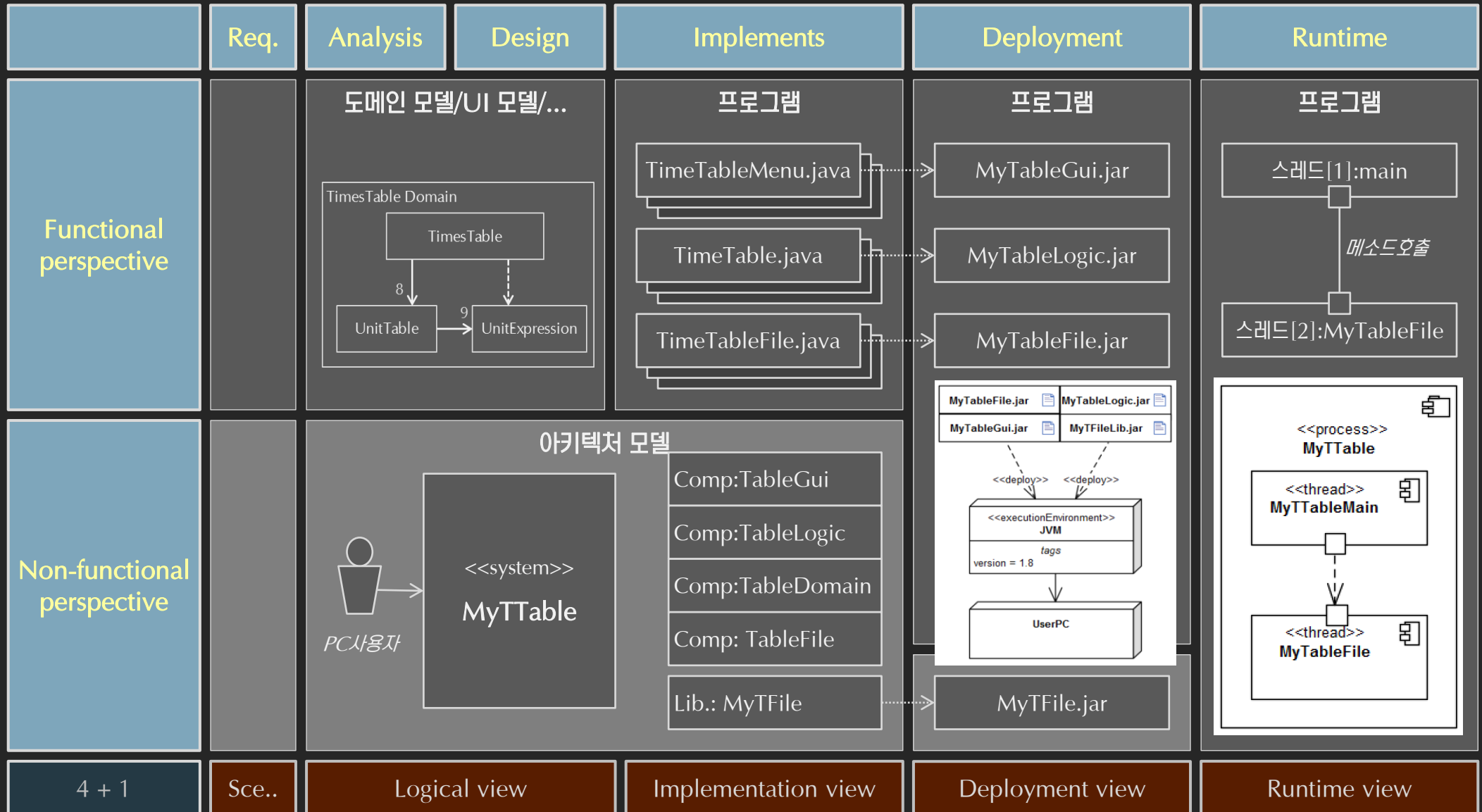


Process(Runtime)View

독립형(stand-alone) 시스템

	Requirement	Analysis	Design	Implementation
Functional perspective	<ul style="list-style-type: none">✓ 곱셈 조회✓ 단 별 조회✓ 구구단 조회✓ 유형 별 출력✓ 서식 지정	<p>도메인 모델/UI 모델/데이터 모델</p> <div><p>TimesTable Domain</p><div><div>TimesTable</div><div>UnitTable</div><div>UnitExpression</div><p>8</p><p>9</p></div></div> <div><p>도메인 모델링</p><p>UI 모델링/디자인</p><p>데이터:파일 레이아웃</p></div>		<p>프로그램</p> <div><div>TimeTableMenu.java</div><div>TimeTable.java</div><div>TimeTableFile.java</div></div>
Non-functional perspective	<ul style="list-style-type: none">✓ 개인 사용자✓ 개인용 PC✓ 5회/1일✓ 조회 1초 이내	<p>아키텍처 모델</p> <div><div><p>PC사용자</p></div><div><div><<system>></div><div>MyTTable</div></div><div><div>UI 레이어</div><div>도메인 레이어</div><div>데이터 레이어</div></div><div><div>Comp:TableGui</div><div>Comp:TableLogic</div><div>Comp: TableFile</div><div>Lib.: MyTFile</div></div><div><p>프로토타입/파일라이브러리</p><div><div>결정: JavaFX/RCP</div><div>결정: 레이어기반호출</div><div>결정: 파일 저장소</div><div>결정: 도메인 외부제공</div><div>결정: 파일처리스레드</div></div><div>아키텍처 프로토타입</div></div></div>		
4 + 1	Scenarios view	Logical view		Implementation view

독립형(stand-alone) 시스템



아키텍처 중심 또는 도메인 중심 ?

	소규모 도메인	중규모 도메인	대규모 도메인
Functional perspective	상대적으로 작은 규모 도메인	도메인	상대적으로 큰 규모 도메인
Non-functional perspective	상대적으로 큰 아키텍처	아키텍처	상대적으로 작은 아키텍처
	Architecture centric ✓ embedded 시스템 ✓ 플랫폼, 프레임워크, 프로젝트 라인 ✓ 제어 시스템 ✓ 독립형(stand-alone) 시스템		Domain centric ✓ 기업용 시스템 ✓ 인터넷 서비스 시스템 ✓ 솔루션 시스템 ✓ 오피스용 시스템

독립 시스템에서 서비스 시스템으로 확장

Scalability → 1000 만명 회원

Availability → 24 x 369, 서비스 정지 없음

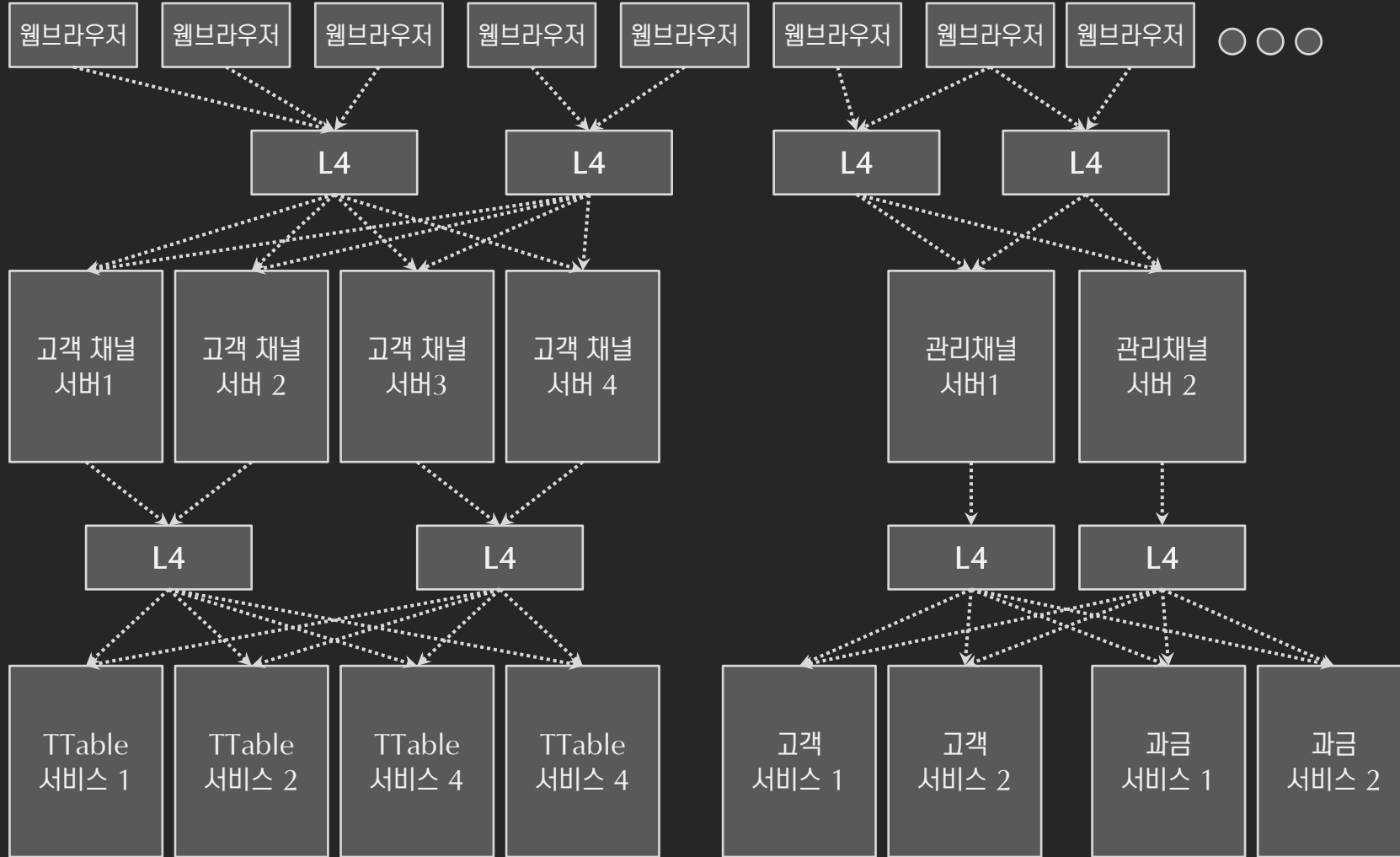
Performance → 모든 서비스 요청은 1초 이내

Easy of use → 누구나 서비스를 쉽게 사용할 수 있도록

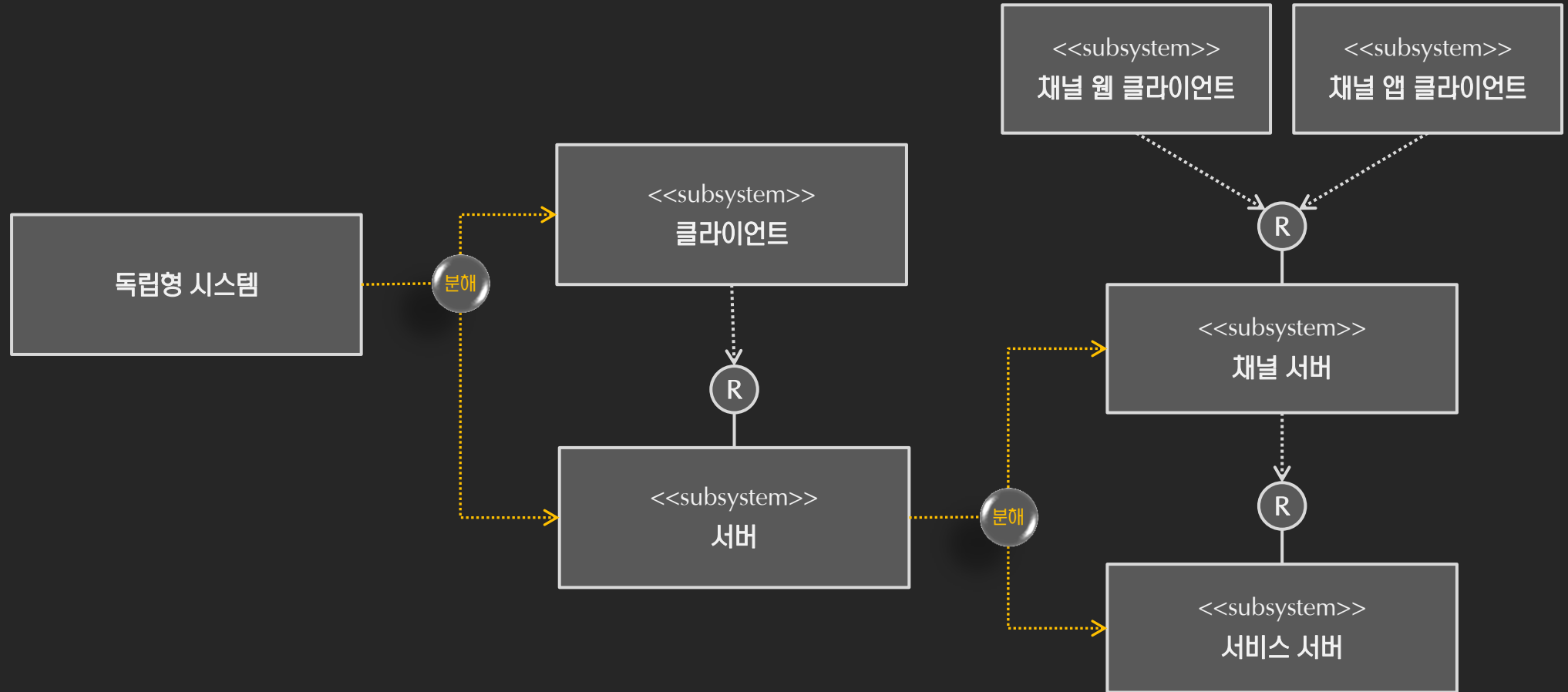
Extendability → 스마트 폰, 와치 등으로 디바이스 확장

And more...

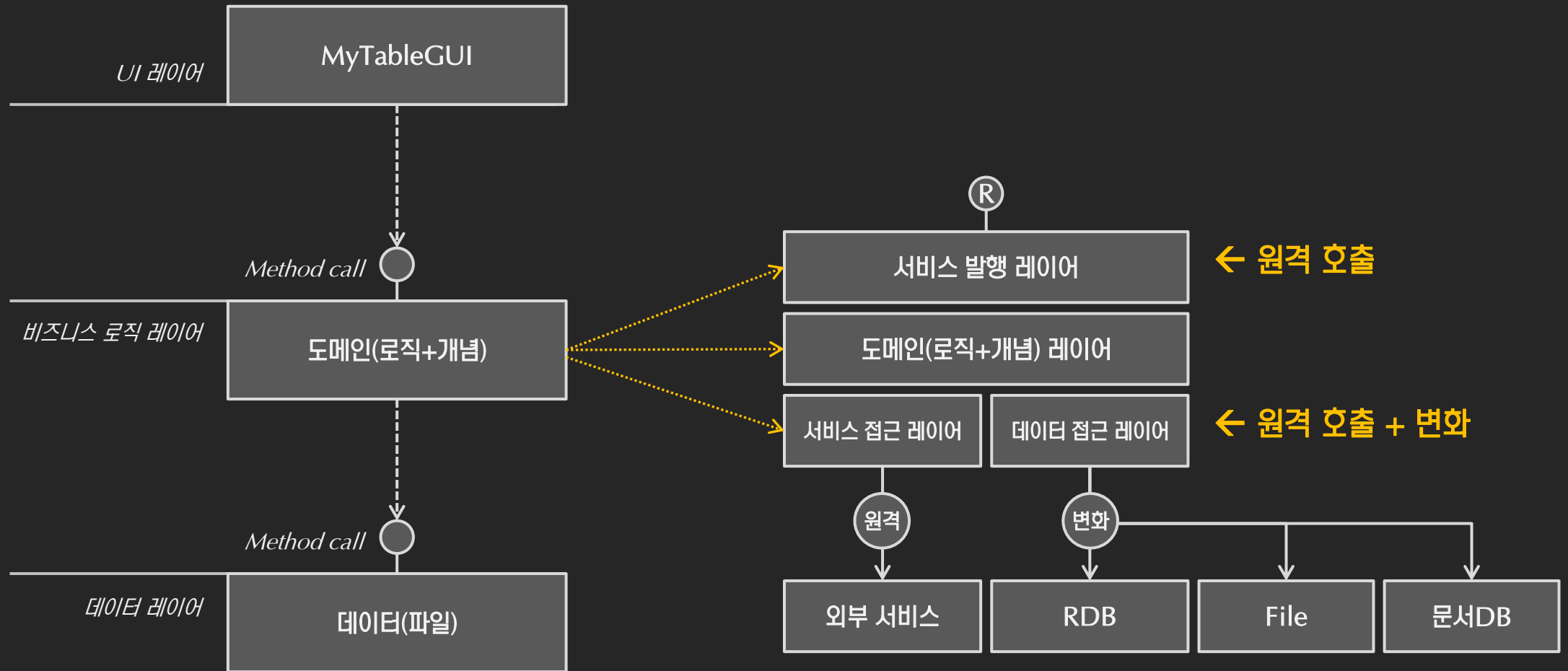
예상 Deployment view



Logical view: 서브-시스템 식별

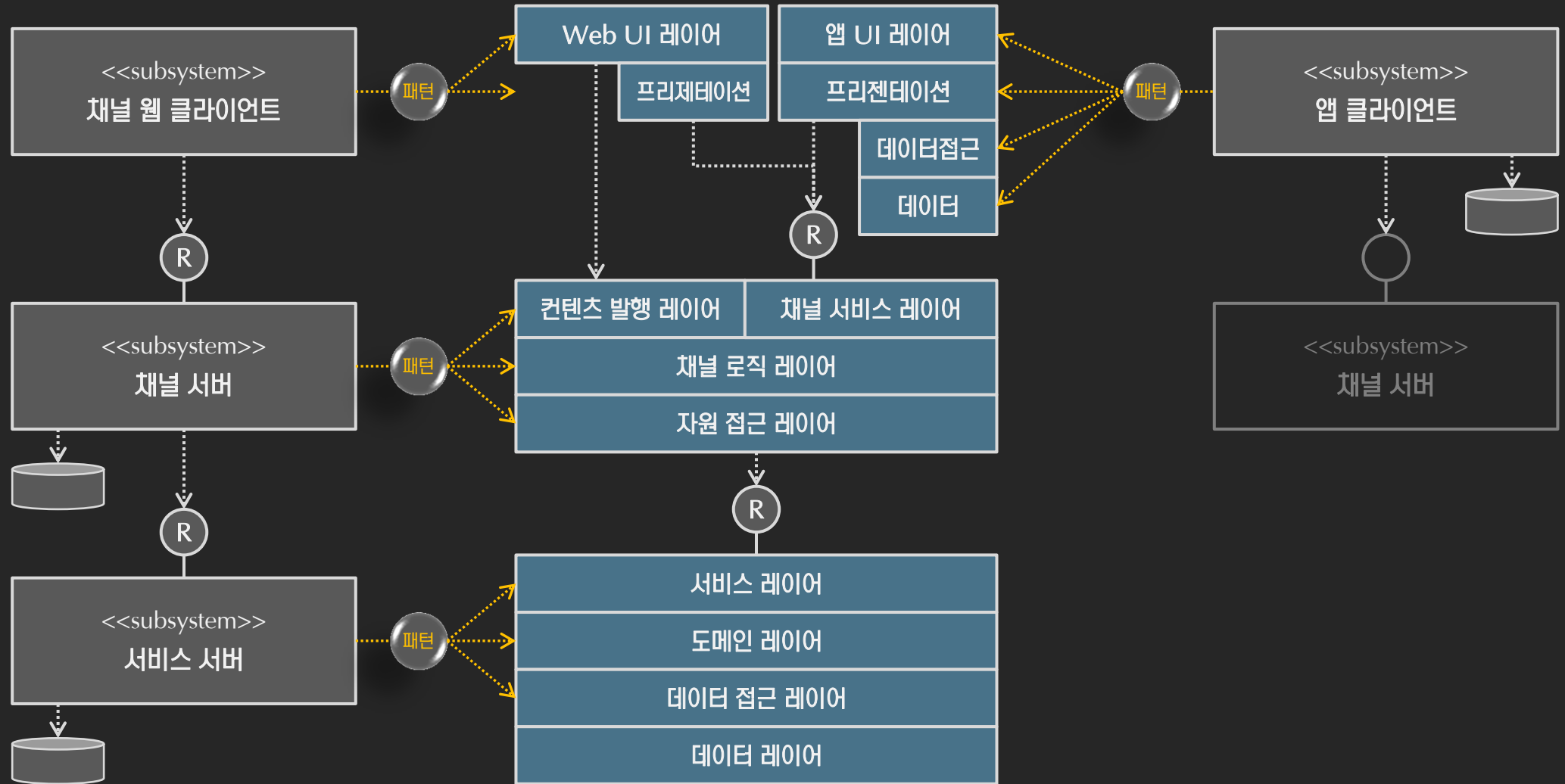


Logical view: 레이어 확장

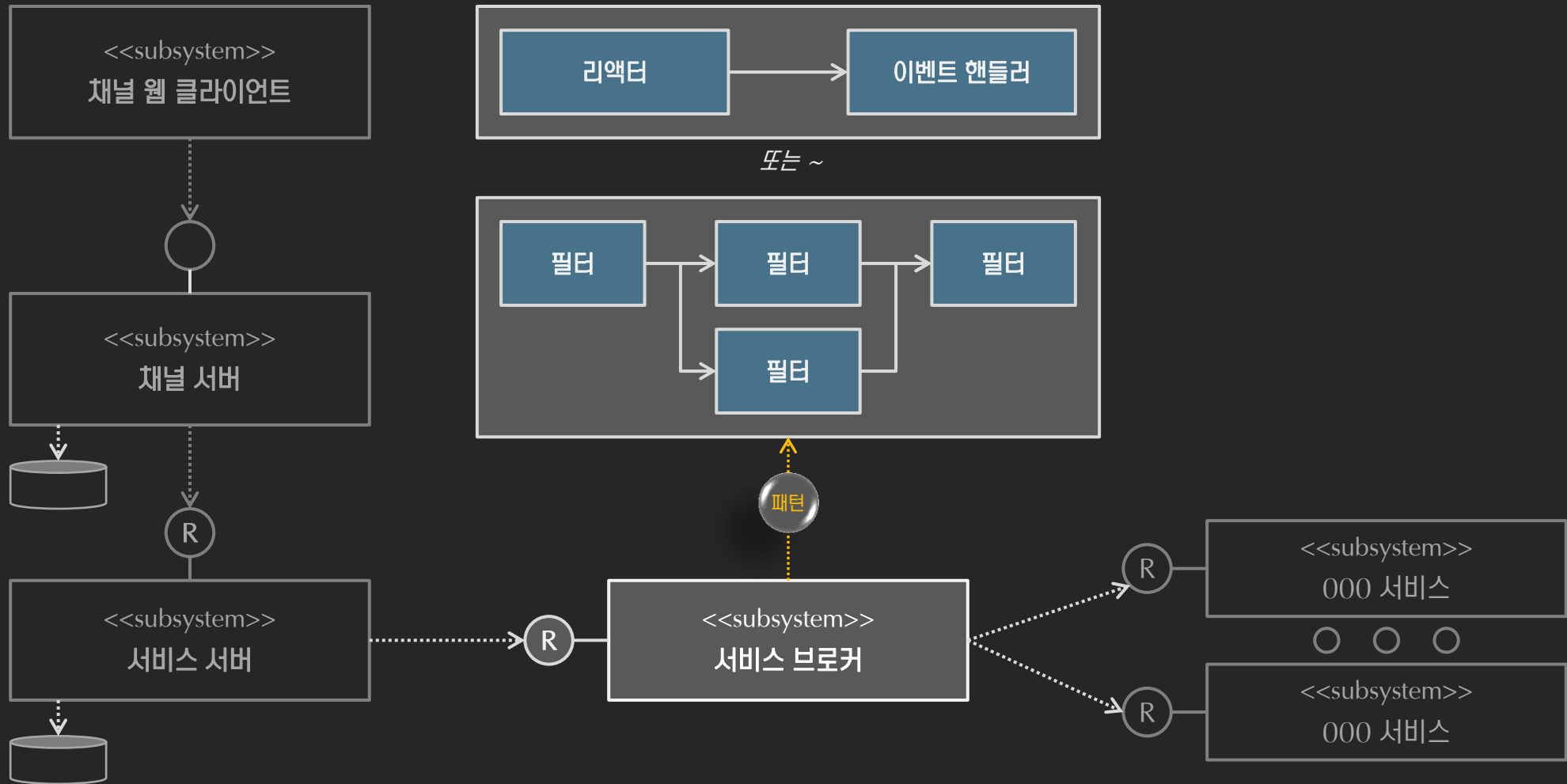


Logical View (모듈/설계 컴포넌트)

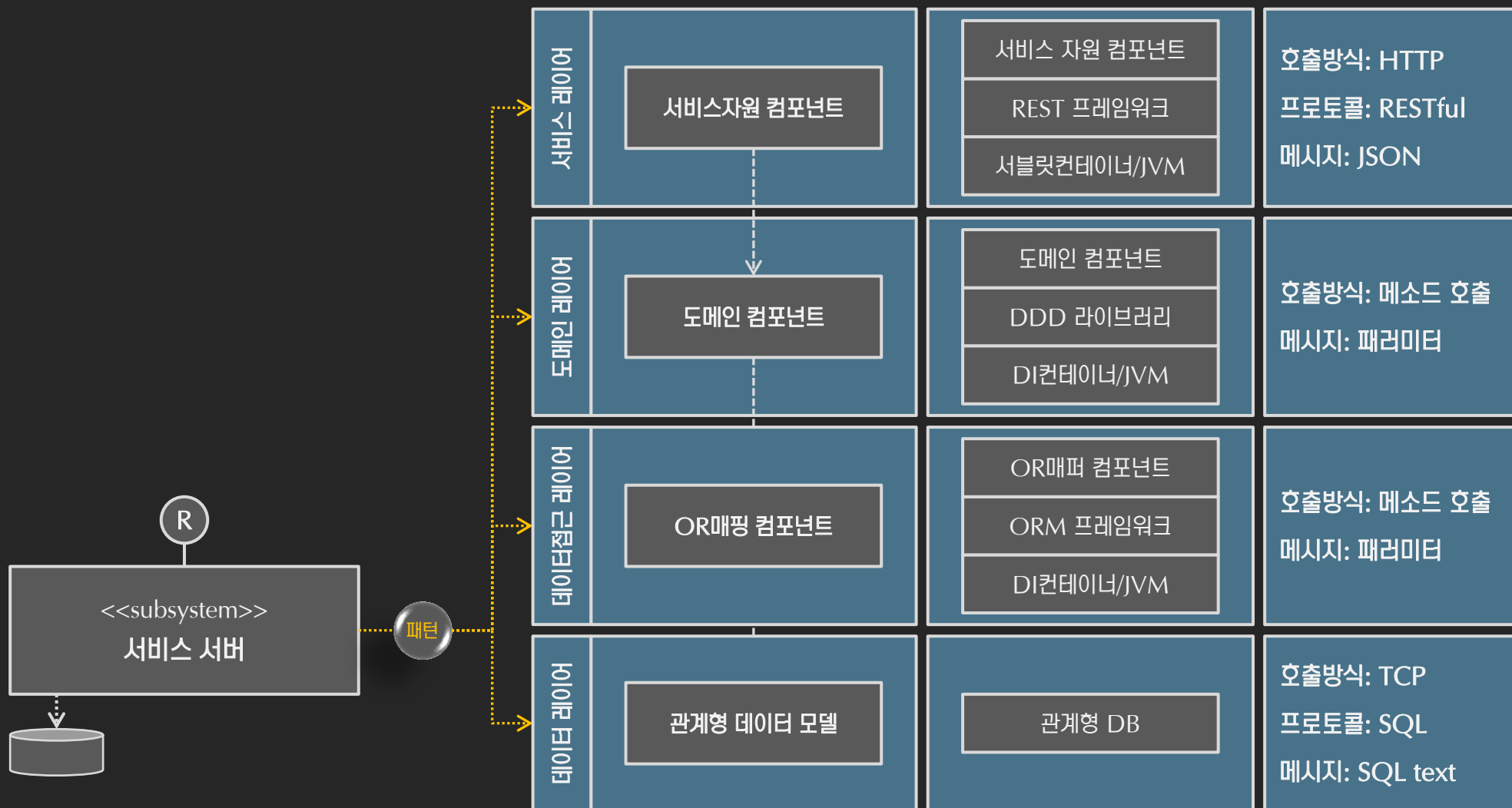
Logical view: 레이어 설계



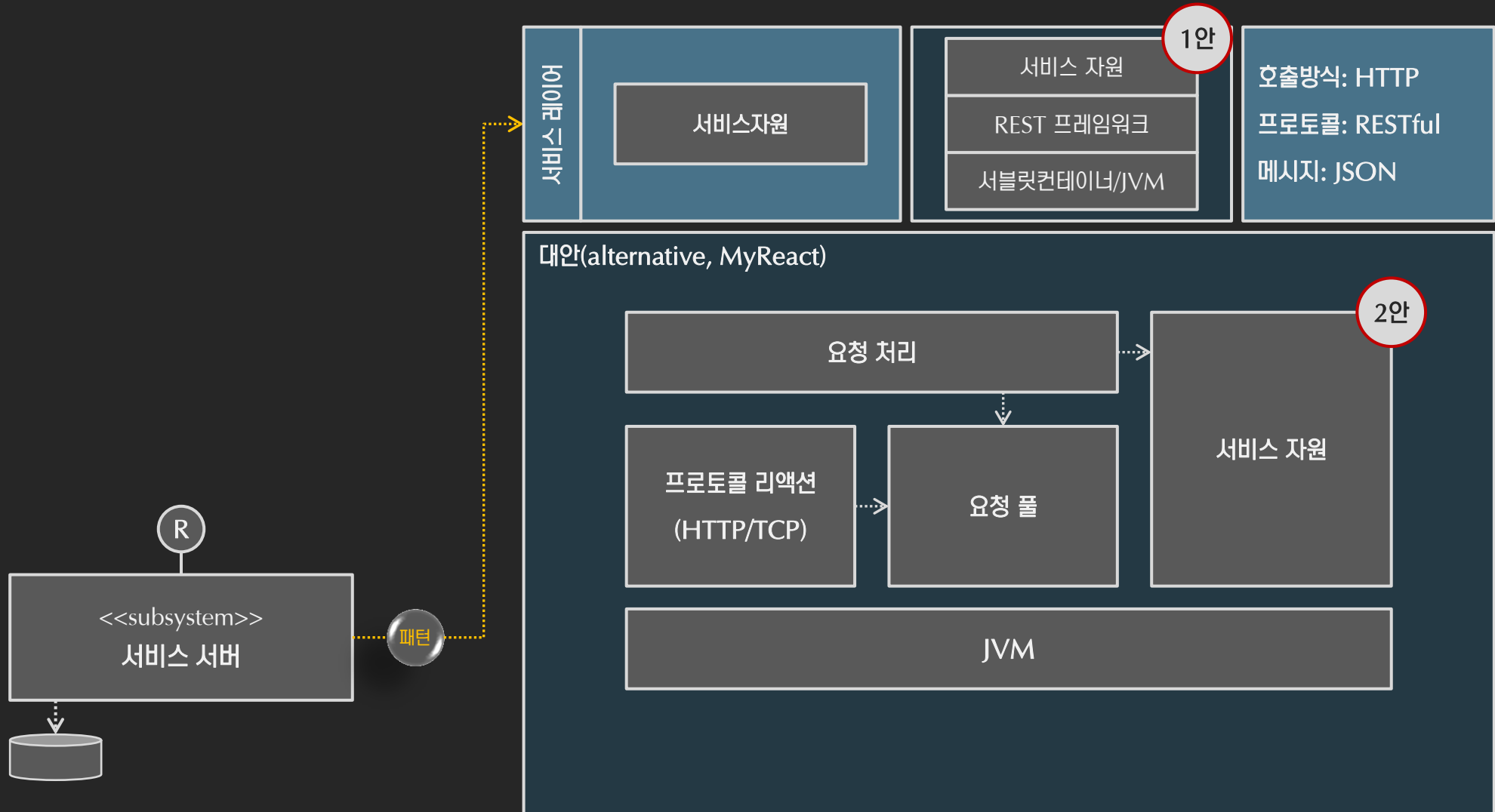
Logical view: 다른 아키텍처 패턴의 예



Logical view: 레이어 구성 모듈



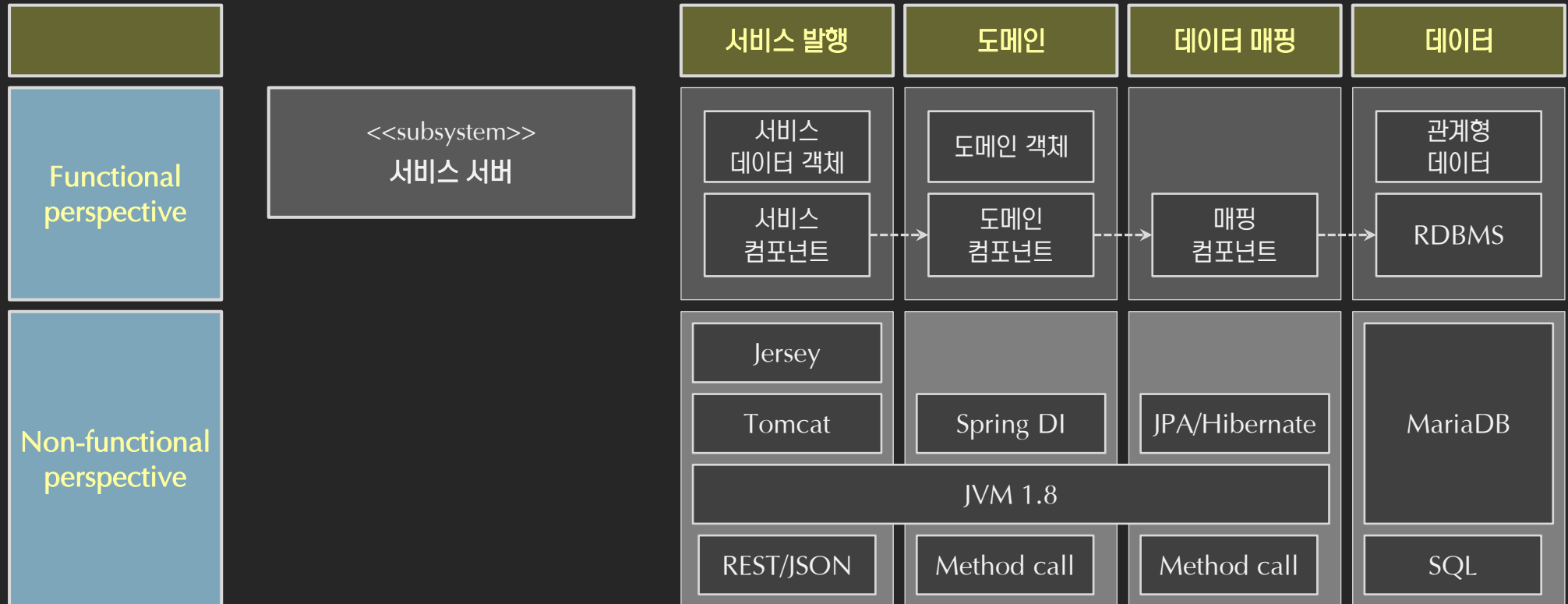
Logical view: 레이어 구성 – 설계 대안



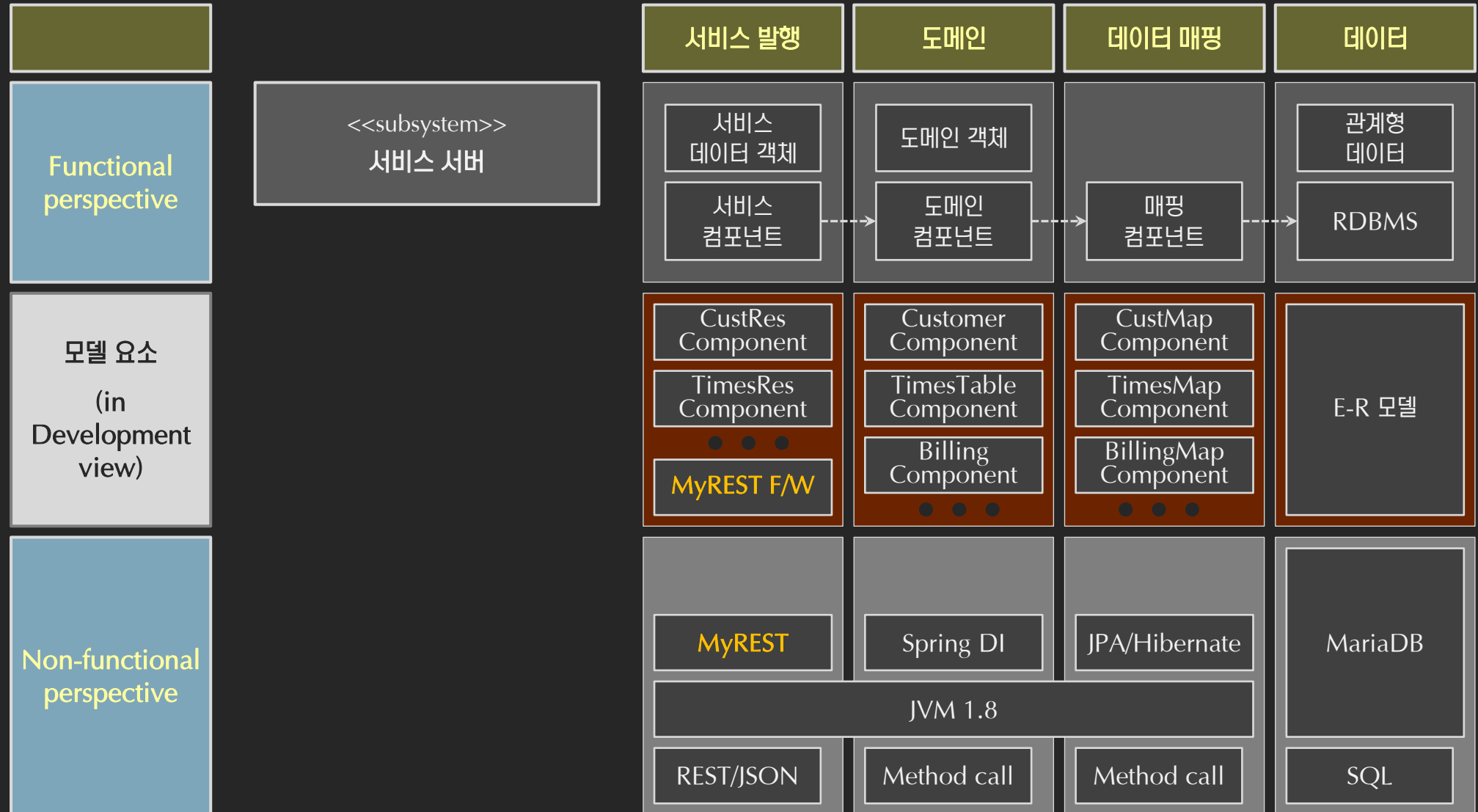
Logical view: 레이어 구성 요소 - 상세



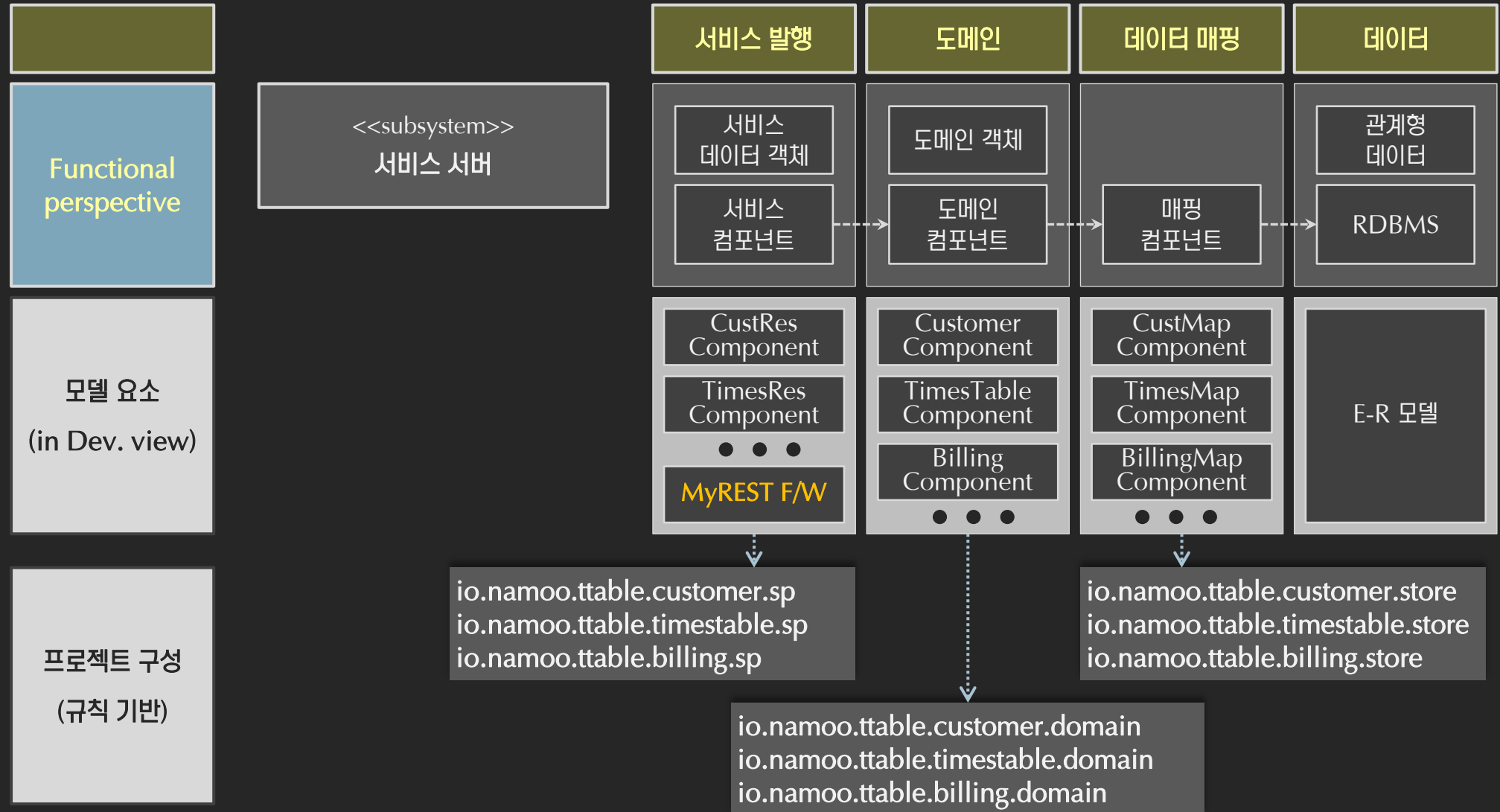
Logical view: 레이어 구성 요소 – 서비스 서버



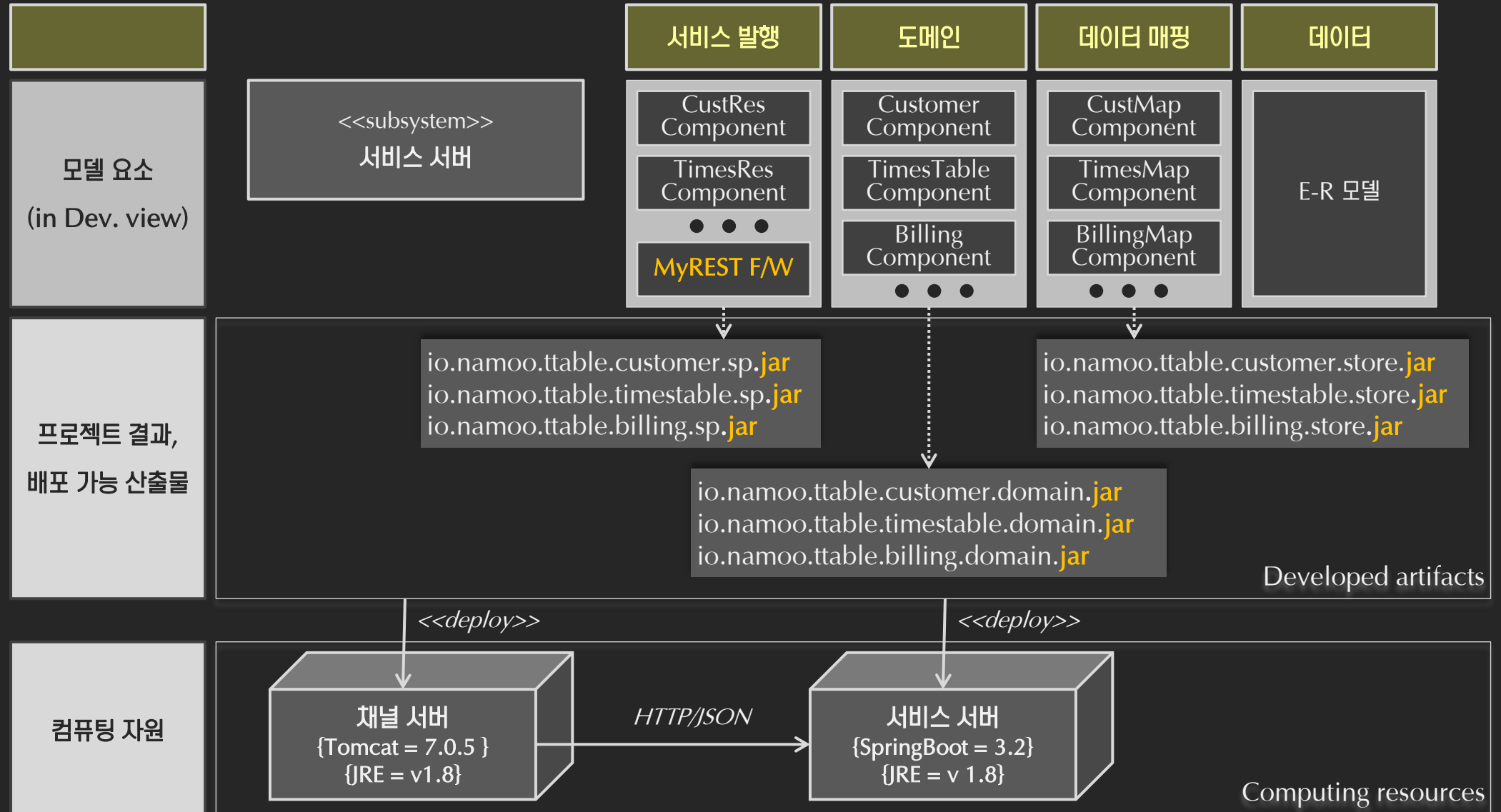
Development view



Development view



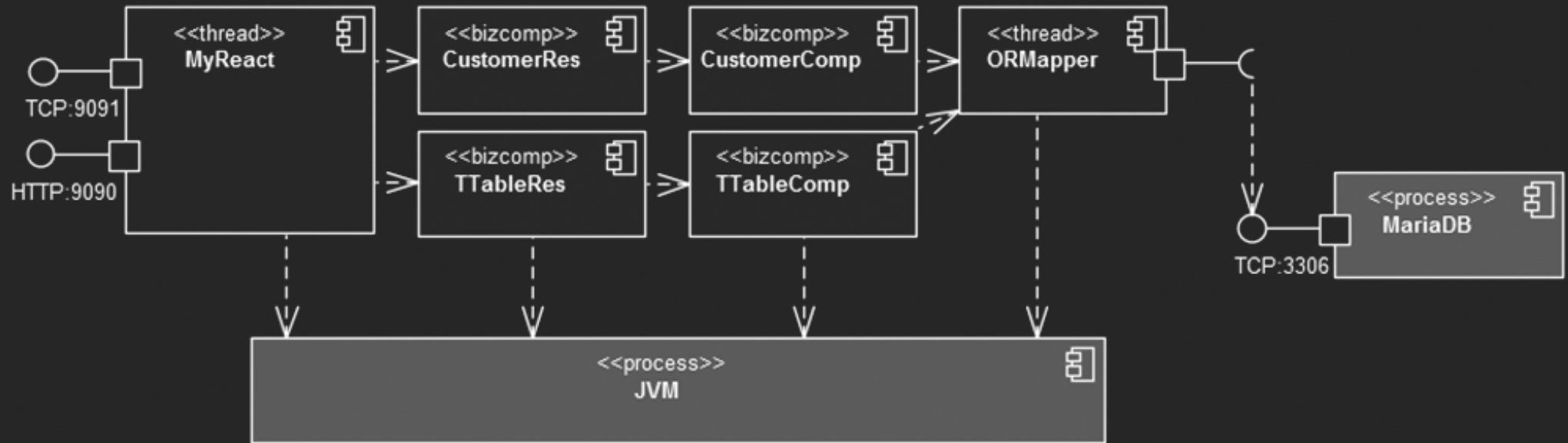
Deployment view – static only



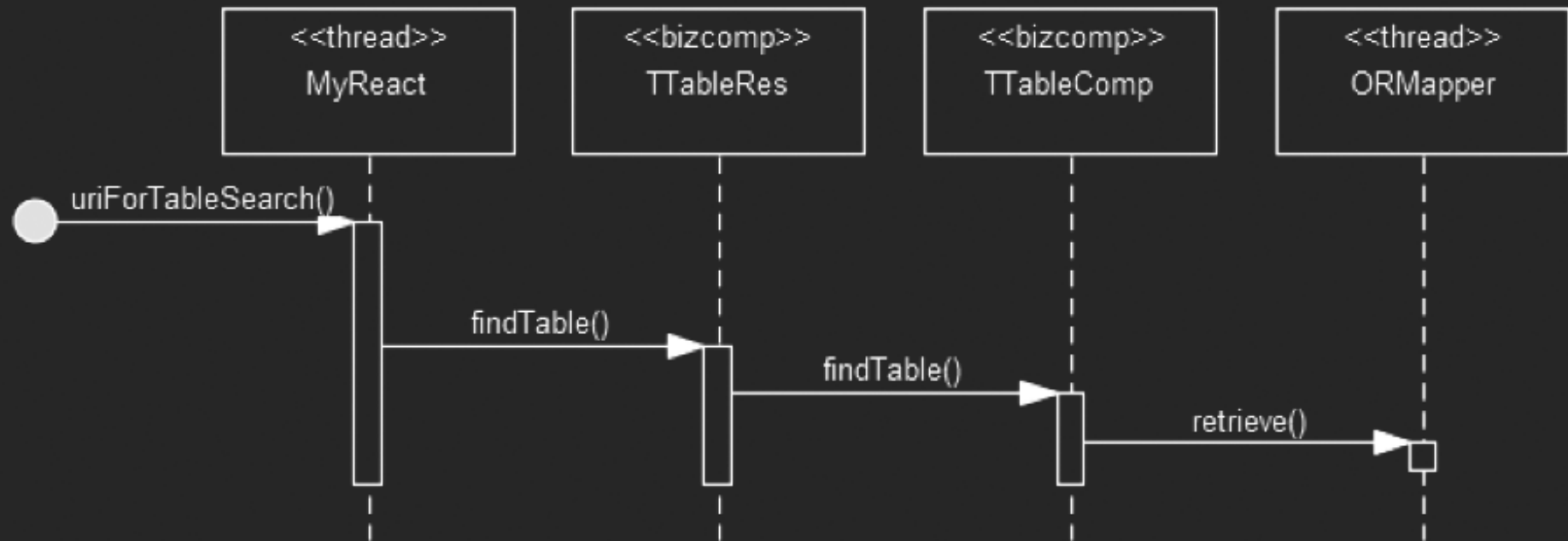
Process(Runtime) view – Level 1



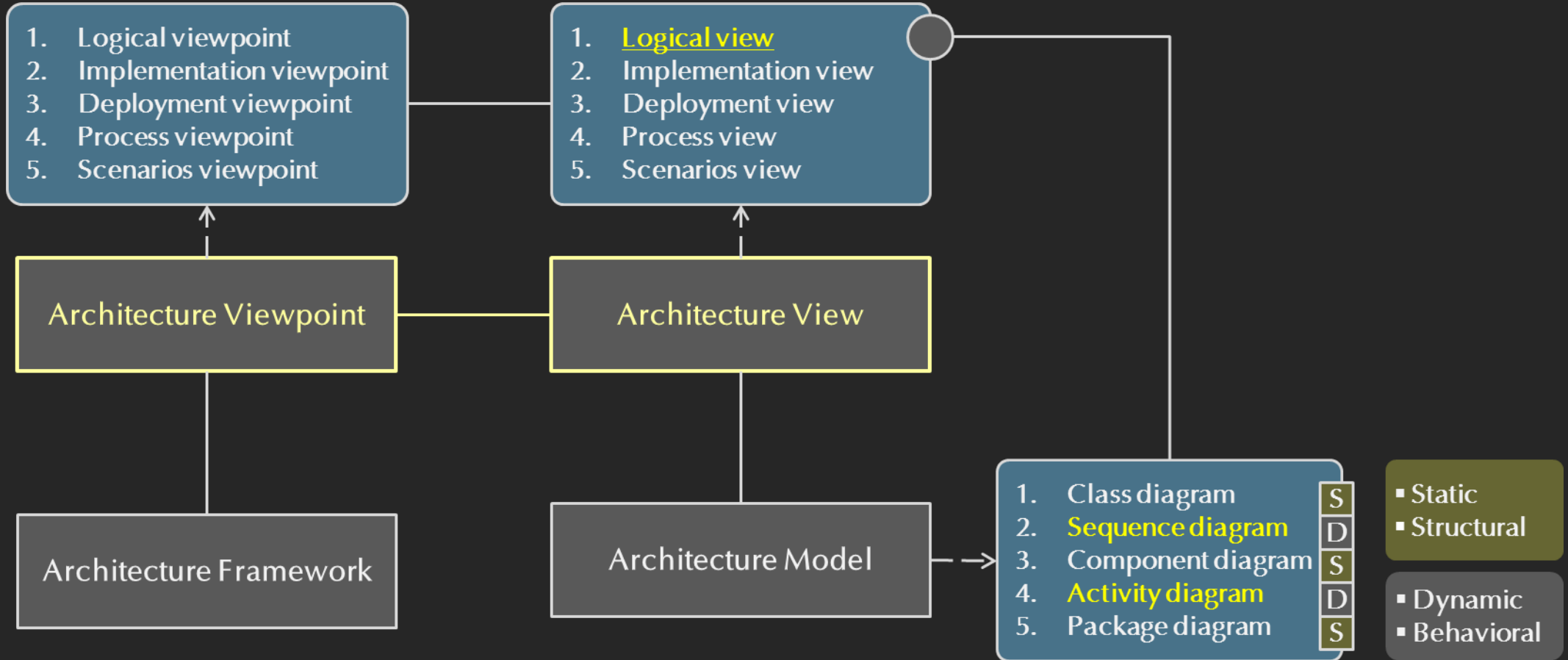
Process(Runtime) view – level 2 - static



Process(Runtime) view – level 2 - dynamic



요약 1



요약 2

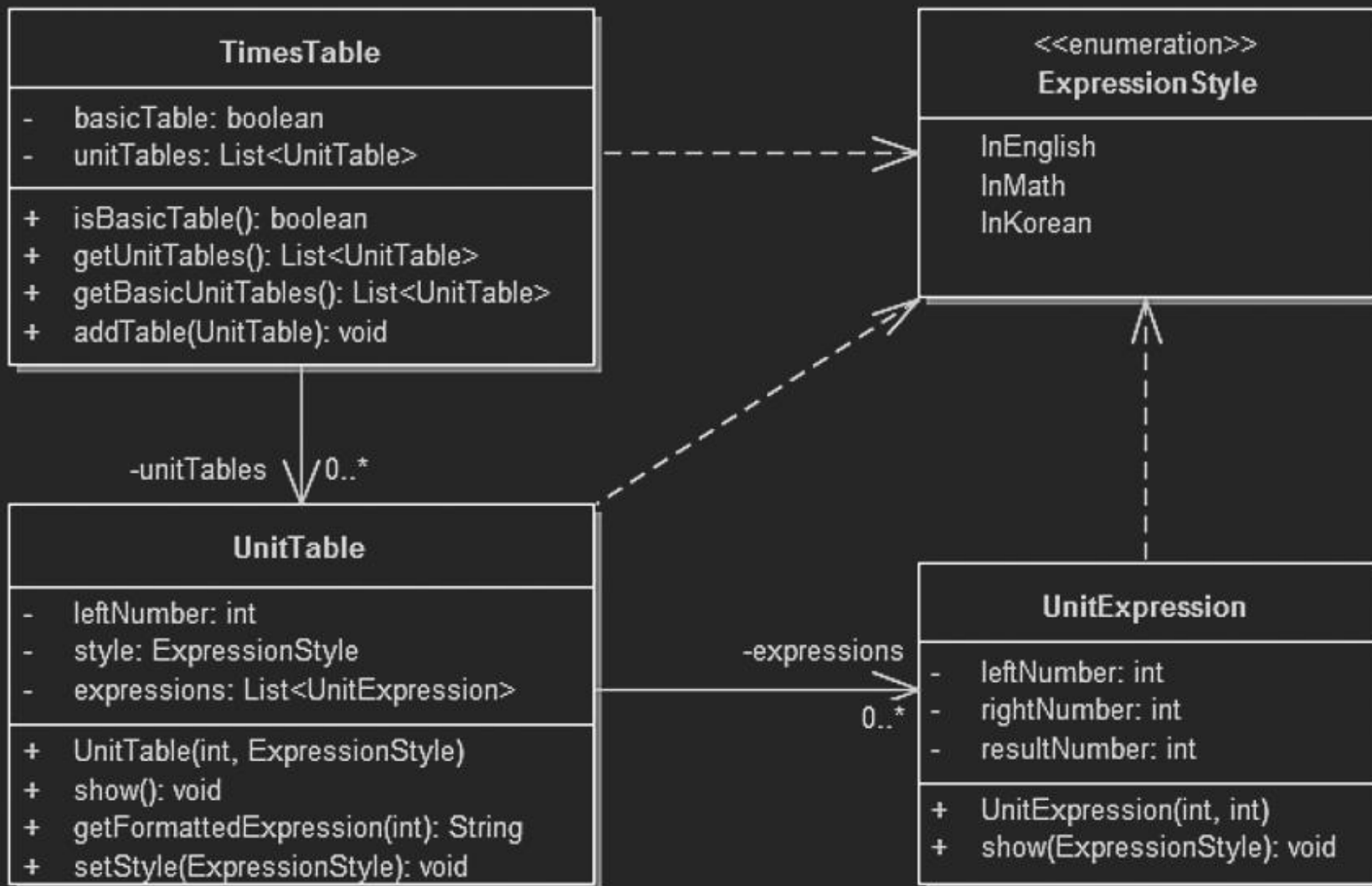
		Requirement	Analysis	Design	Implements	Deployment	Runtime
Functional perspective		기능 요건	도메인 모델/UI 모델/데이터 모델 <div>분석 수준 모델설계 수준 모델</div>		비즈니스 컴포넌트 프로그램 또는 프로그램	배포 비즈니스 컴포넌트, 라이브러리	프로그램 실행
Non-functional perspective		비기능 요건	아키텍처 모델 <div>패턴/스타일반복적인 분해 모델</div>		아키텍처 요소 프로그램 <div>개발 모델</div>	배포 아키텍처 요소 <div>배포 모델</div>	프로그램 실행 <div>런타임 모델</div>
아키텍처 프레임워크	4 + 1	Use Case view (or Scenarios view)	Logical view	Development view (or Implementation view)		Physical view (or Deployment view)	Process view (or Runtime view)
	CMU SEI	N/A	Module view	Allocation view		Allocation view	C&C view
	ACDM	N/A	Static perspective			Physical perspective	Dynamic perspective

3. 분리 – 아키텍처와 도메인 모델

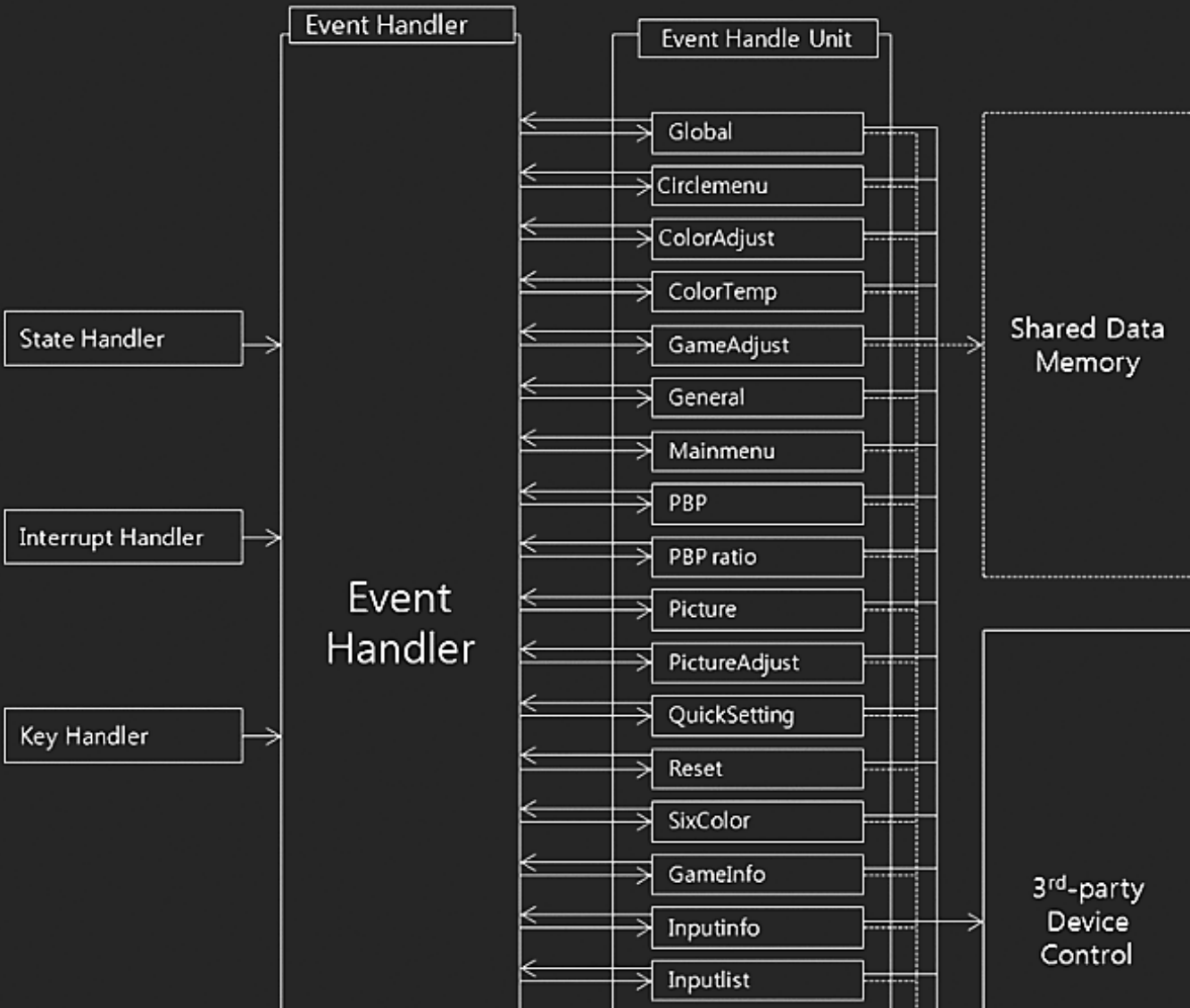
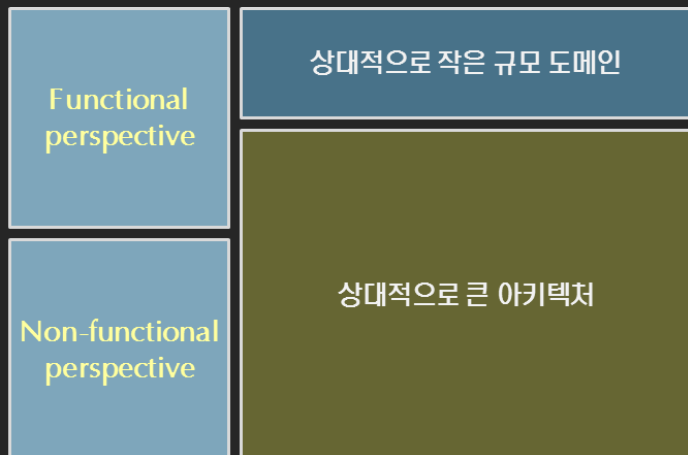
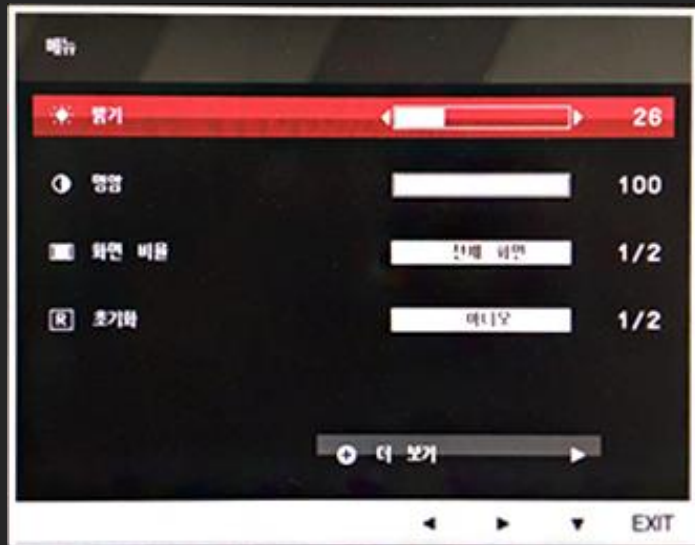
도메인 객체와 메커니즘



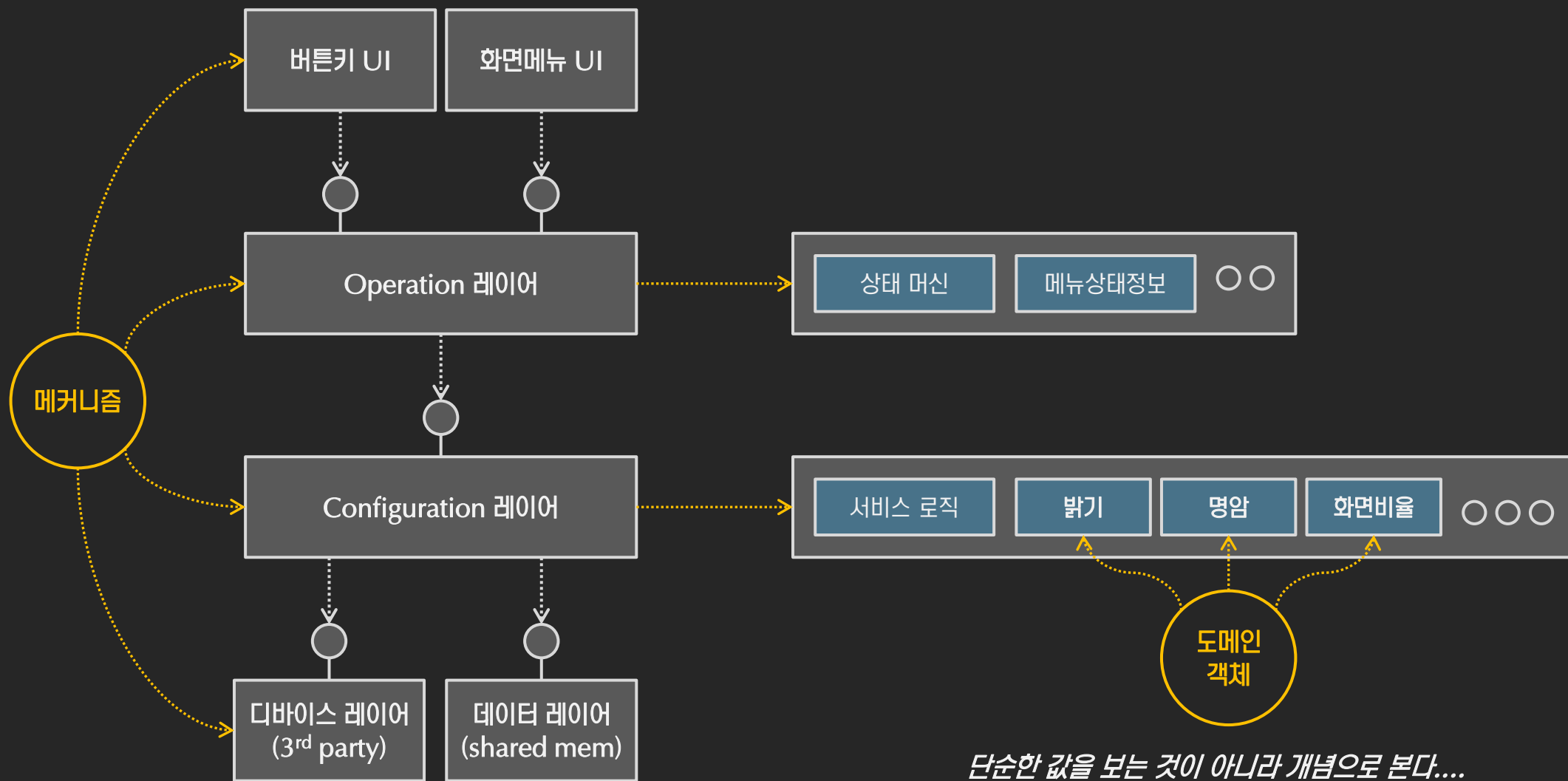
도메인 모델의 가치



도메인과 아키텍처를 섞으면...



도메인과 아키텍처를 나누면...



무엇을 분리하는가?

도메인 모델 VS 아키텍처 모델

서비스 컨텍스트 VS 사용 컨텍스트

개념 VS 영속성

개념 VS 사용자 인터페이스

관심사의 분리

관심사의 분리(Separation of concern)는 아키텍팅 활동의 Best practice이며 원칙입니다. SoC의 다른 표현은 Loose coupling, High cohesion 입니다. 이런 원칙을 지키며 설계하는 이유는 단순함을 지켜 주기 때문입니다. 결국 기본으로 돌아가서 원칙을 준수하는 설계를 하면, 좋은 소프트웨어를 개발할 수 있습니다.

요약

1. 시스템의 본질은 도메인에 담아야 합니다.
2. 아키텍처는 기준 뷰를 갖고 분해하며 설계합니다.
3. 도메인 모델과 아키텍처 모델은 분리합니다.

감사합니다. !!

Nextree Consulting, Taegook Song