소프트웨어 공학 발표

김민혁 | 이재민 | 최 혁

2024.5.27

CONTENTS

SECTION 01

프로젝트 현장

프로젝트명과 개요

개발자의 협업을 위한 워크스페이스 프로그램

Code ✚ Cooperate

Codeperate

프로젝트 목적

Teamspace Linking + Live + IDE 통합 환경

오류 발생 확률 감소

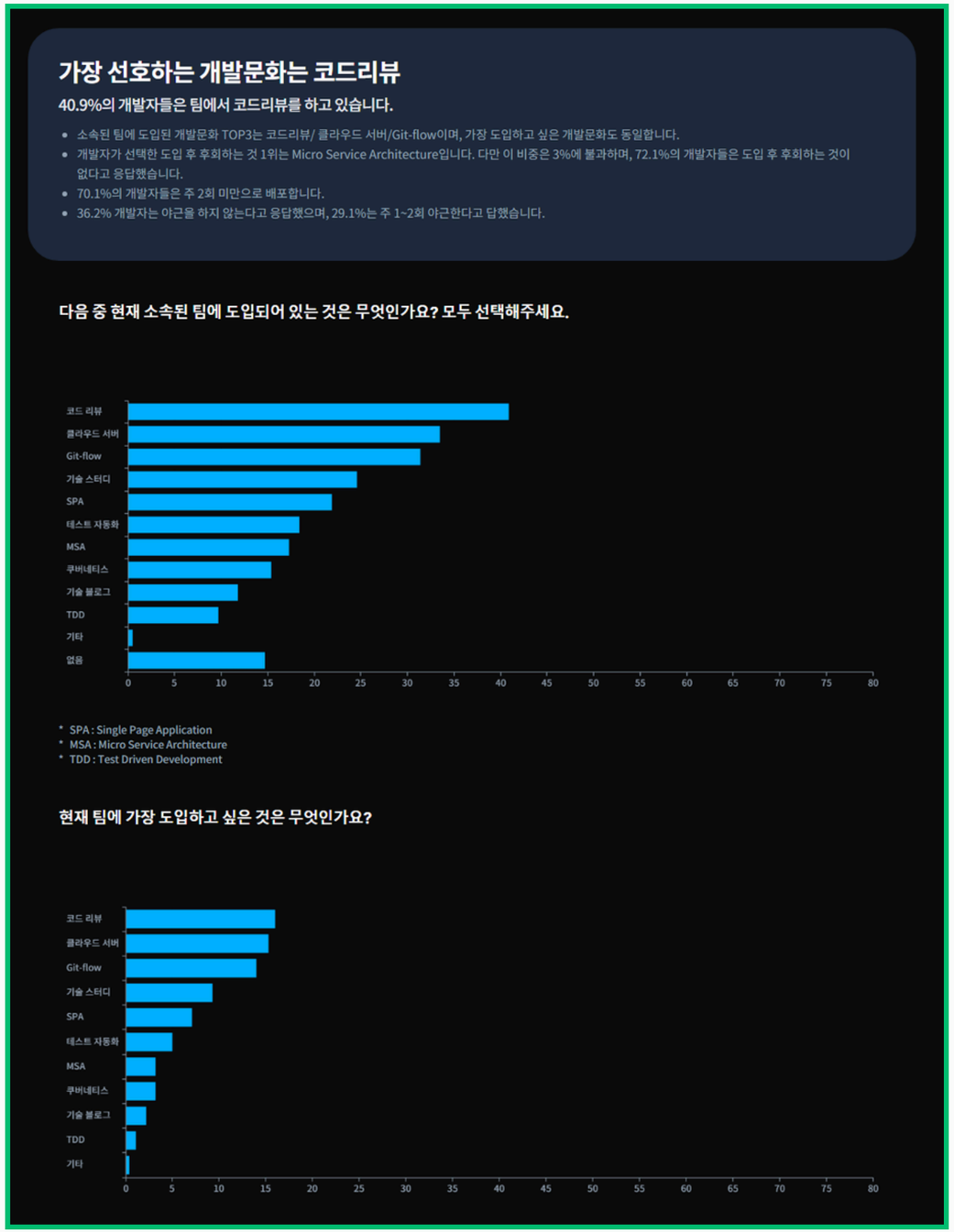
효율적인 팀 작업 가능

프로젝트 당위성

2023 프로그래머스 사이트 개발자 설문조사

개발자가 가장 선호하는 개발문화

코드 리뷰 42.3%



프로젝트 당위성

현직 개발자 인터뷰 진행

개발자의 팀 단위 프로젝트에서 중요한 요소

코드 일관성

팀 프로젝트 경험

예비 개발자 입장에서 팀 프로젝트를 진행할때 필요한 요소

팀 협업 기능

예산 예측치

기능 점수 모델

노력 분석

GFP 320.3

PCA 1.19

GFP

	기능분야	개수		복잡도		간이 계산값
			단순	보통	복잡	
1	외부 입력	16	3	4	6	4
2	외부 출력	16	4	5	7	5.2
3	외부 조회	10	3	4	6	3.9
4	내부 논리 파일	15	7	10	15	7.5
5	외부 인터페이스	4	5	7	10	5.4

위의 표를 바탕으로 GFP를 계산하였다.

외부 입력 : $16 \times 4 = 64$

외부 출력 : $16 \times 5.2 = 83.2$

외부 조회 : $10 \times 3.9 = 39$

내부 논리 파일 : $15 \times 7.5 = 112.5$

외부 인터페이스 : $4 \times 5.4 = 21.6$

$GFP = 64 + 83.2 + 39 + 112.5 + 21.6 = 320.3$

따라서 GFP는 320.3으로 산정했다.

노력 분석

FP

381.157

17.4 MM

5 점 → 6개

4 점 → 1개

3 점 → 6개

2 점 → 1개

1 점 → 0개

0 점 → 0개

평가된 항목점수에 따라 PCA를 계산하였다.

$$PCA = 0.65 + 0.01 \times (5 \times 6 + 4 \times 1 + 3 \times 6 + 2 \times 1) = 1.19$$

따라서 **PCA**는 **1.19**로 설정하였다.

설정한 GPF와 PCA의 값을 바탕으로 FP를 계산하였다.

$$FP = 320.3 \times 1.19 = 381.157$$

소수점은 첫번째 자리수까지 표시하여 **FP**는 **381.1**로 설정하였다.

국내 기능 점수 당 평균 생산성은 개발자의 역량을 고려하여 **예외값 하한**으로 잡았다. 따라서 **생산성**은 **21.8**로 설정했다.

아래는 위에서 설정한 FP와 생산성으로 MM을 구하는 과정과 결과이다.

$$MM = FP / 생산성 = 381.1 / 21.8 = 17.481$$

따라서 노력치(E)는 17.4 MM으로 산정했다.

프로젝트 헌장

자원 분석

개발 인력은 백엔드 개발자 2명, 프론트 엔드 개발자 2명, UI/UX 디자이너 1명, 서버 관리자 1명, 프로젝트 관리 및 QA(Quality Assurance) 1명 총 7명으로 각 파트별로 투입 비율을 판단하여 자원(R)을 설정하였다.

백엔드 개발자 : 50% → 1명

프론트 엔드 개발자 : 30% → 0.6명

UI/UX 디자이너 : 20% → 0.2명

서버 관리자 : 10% → 0.1명

프로젝트 관리 및 QA : 10% → 0.1명

$$1 + 0.6 + 0.2 + 0.1 + 0.1 = 2.0$$

위의 결과에 따라 **자원은 2.0명**으로 산정했다.

2.0 명

기간 분석

노력(Effort)과 자원(Recourse)를 나눠서 기간(Duration)을 계산하였다.

노력 : 17.4 MM

자원 : 2.0 명

$$D = E/M = 17.4/2.0 = 8.7$$

위의 결과에 따라 기간은 8.7개월로 산정했다.

8.7 개월

프로젝트 헌장

비용 예측 결과

인력 비용 : 22,010 만원
기타 비용 : 1,681만원

총예산 **23,691만원**

노력(Effort) : 17.4 MM

자원(Recourse) : 2.0 명

기간(Duration) : 8.7개월

위에서 예측한 노력, 자원, 기간을 토대로 인력 비용과 기타 비용을 산정하였다.

인력비용은 기본 월 300만원을 할당하고 인력 비율을 100만원에 나눠서 추가로 할당하였다.

인력 비용 산정:

- 백엔드 개발자 2인 : $((300\,만원 + (100\,만원 \times 0.5)) \times 2 = 700\,만원$
- 프론트 엔드 개발자 2인 : $(300\,만원 + (100\,만원 \times 0.3)) \times 2 = 660\,만원$
- UI/UX 디자이너 1인 : $300\,만원 + (100\,만원 \times 0.2) = 320\,만원$
- 서버 관리자 1인 : $300\,만원 + (100\,만원 \times 0.1) = 310\,만원$
- 프로젝트 관리 및 QA 1인 : :$300만원 +(100만원×0.1)=310만원$

1개월당 인력비용 : $700\,만원 + 660\,만원 + 320\,만원 + 310\,만원 + 310\,만원 = 2300\,만원$

프로젝트 기간 인력비용 : $2300\,만원 \times 8.7 = 20,010\,만원$

또한 추가 작업이 생길 사항을 고려해 2000만원을 추가하여 총 **22,010만원**으로 프로젝트 인력 비용을 산정했다.

기타 비용 산정

- 테스트 비용 : $300\,만원(테스트인원) + 250\,만원(테스트도구) = 550\,만원$
- 서버 유지 비용 : $80\,만원(월) \times 8.7 = 696\,만원$
- 데이터베이스 관리 비용 : $50\,만원(월) + 8.7 = 435\,만원$

총 기타 비용 : $550\,만원 + 696\,만원 + 435\,만원 = 1681\,만원$

총 예산 계산:

- 총 인력 비용: 22,010만원
- 기타 비용: 1681만원
- 총 예산 : $22,010\,만원 + 1681\,만원 = 23,691\,만원$

따라서 총 예산 예측치는 **23,691만원**으로 산정했다.

SECTION 02

유스케이스 다이어그램

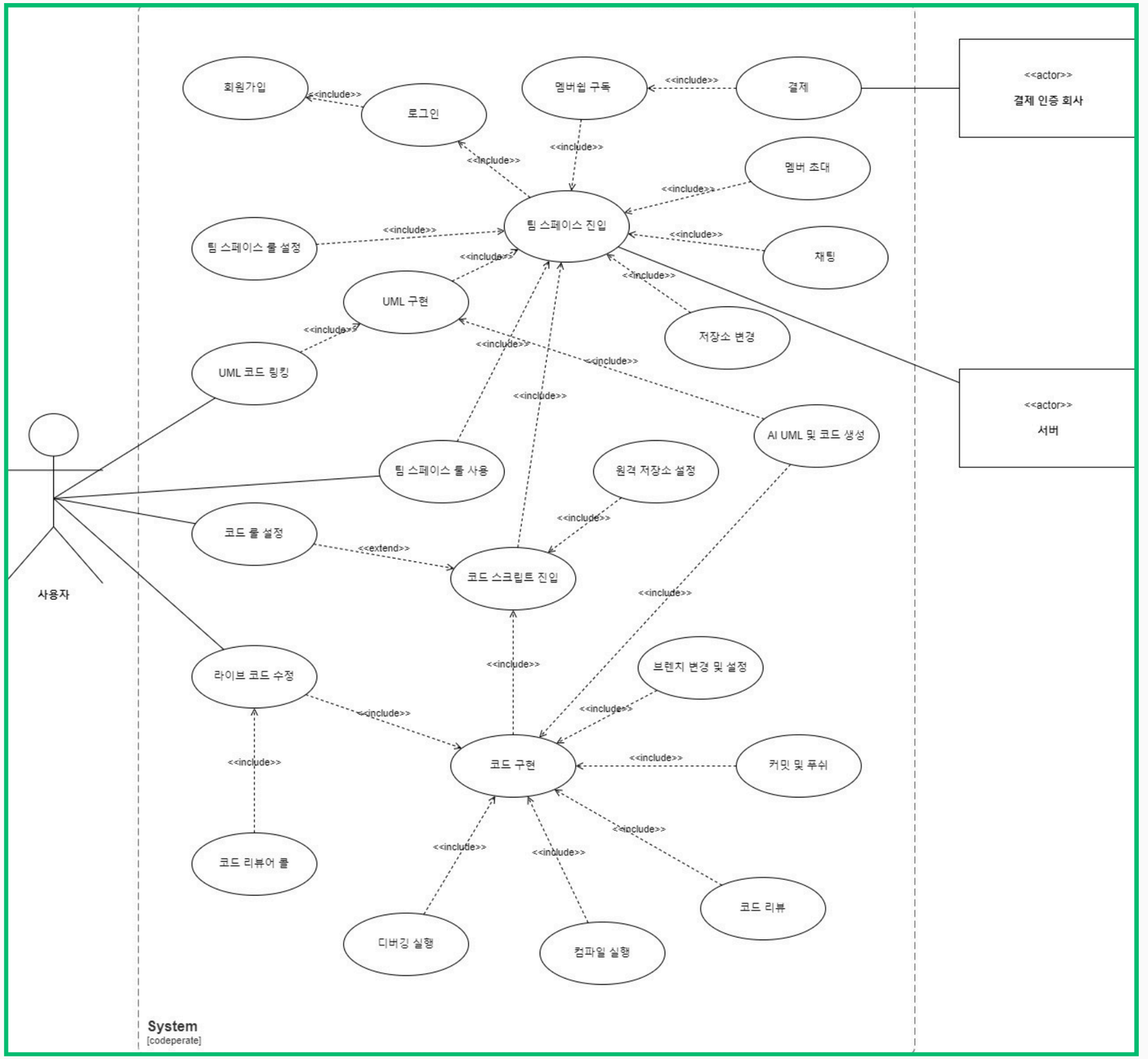
유스케이스 다이어그램

주요 기능

01 UML 코드 링킹

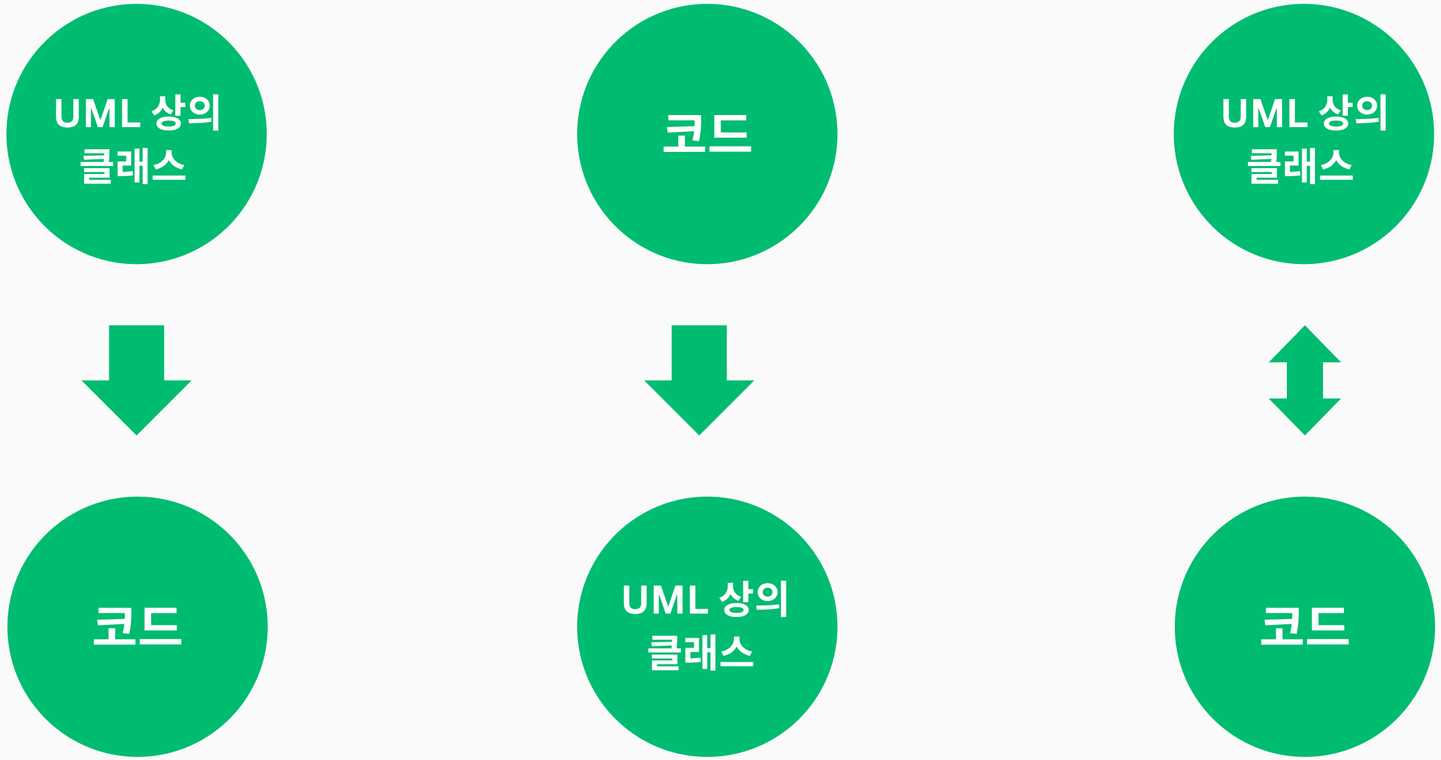
02 코드 룰 설정

03 코드 라이브 수정



UML 코드 링킹

UML데이터와 코드 데이터를 연결하는 기능

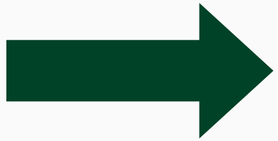


유스케이스 다이어그램

코드 룰 설정

팀 스페이스 내에 코드 컨벤션을 설정하는 기능

Code Rule



파일 나누는 법

모듈 구성하는 법

들여쓰기 규칙

주석 다는 법

함수, 변수, 클래스 이름 짓는 법

유스케이스 다이어그램

코드 라이브 수정

라이브 환경의 IDE를 제공하는 기능

IDE log

Live Code Script

Reviewer call

SECTION 03

유스케이스 명세서

UML 코드 링킹

UML을 코드로 / 코드를 UML로 변환해주는 기능

Actor	**사용자**
Entry Point	**팀 멤버인 사용자가 로그인 상태에서 팀 스페이스 내의 UML페이지 혹은 스크립트 페이지로 진입**

기본 흐름

사용자는 UML페이지 사이드 바에 있는 “UML Linking Code”를 선택

시스템은 팀 스페이스 데이터베이스 내에서 관계가 일치하는 UML과 코드를 탐색하여 링킹

사용자가 링킹된 UML / 코드을 수정하면 수정 내용이 링킹된코드 / UML에 반영됨

시스템은 링킹 데이터를 팀 스페이스 데이터베이스에 저장

대안 흐름

시스템이 서로 일치하는 코드 / UML을 탐색하지 못한 경우

링킹된 상태의 UML과 코드를 삭제 할 경우

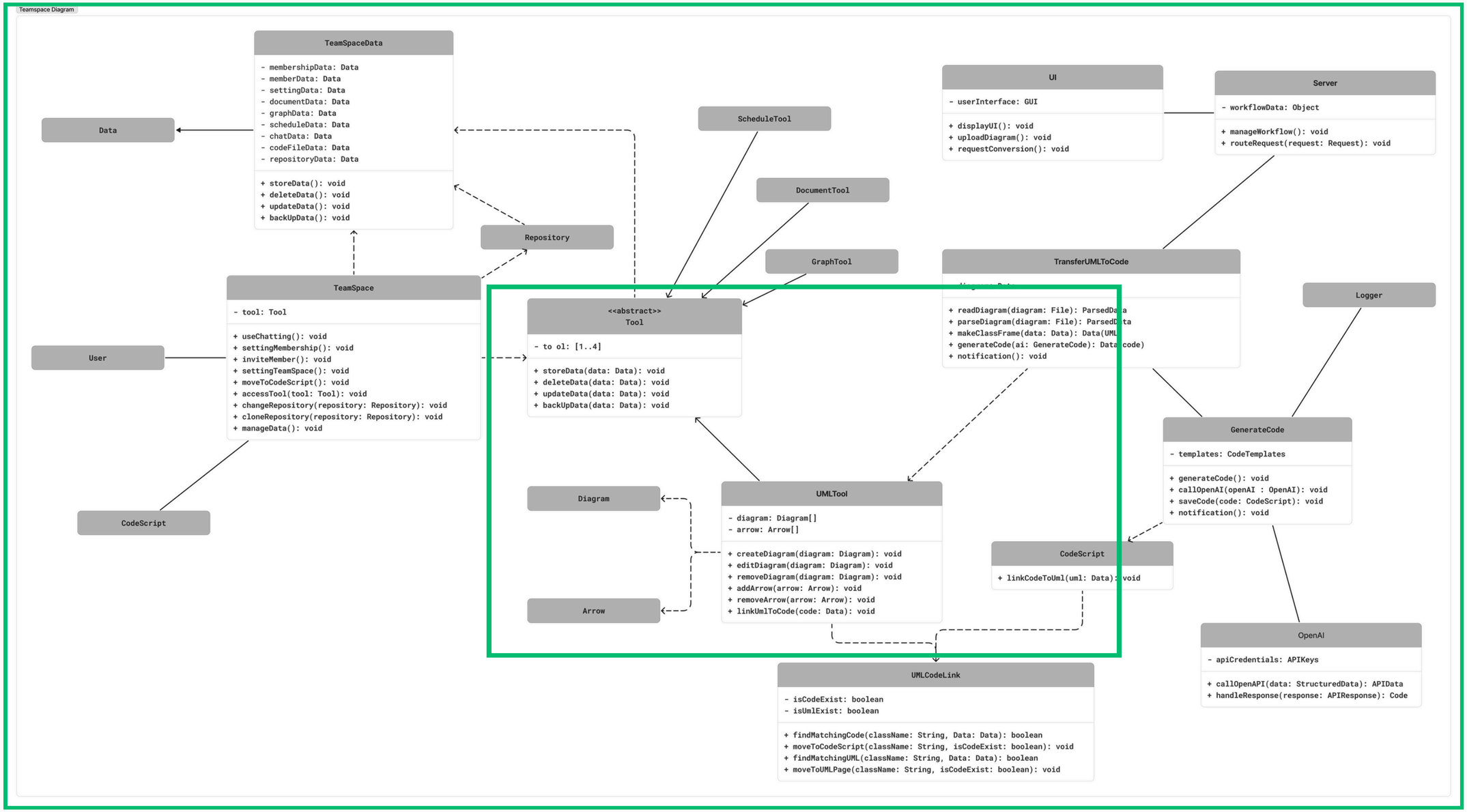
종료 조건

사용자가 다른 페이지로 이동하거나 프로그램을 종료할 경우

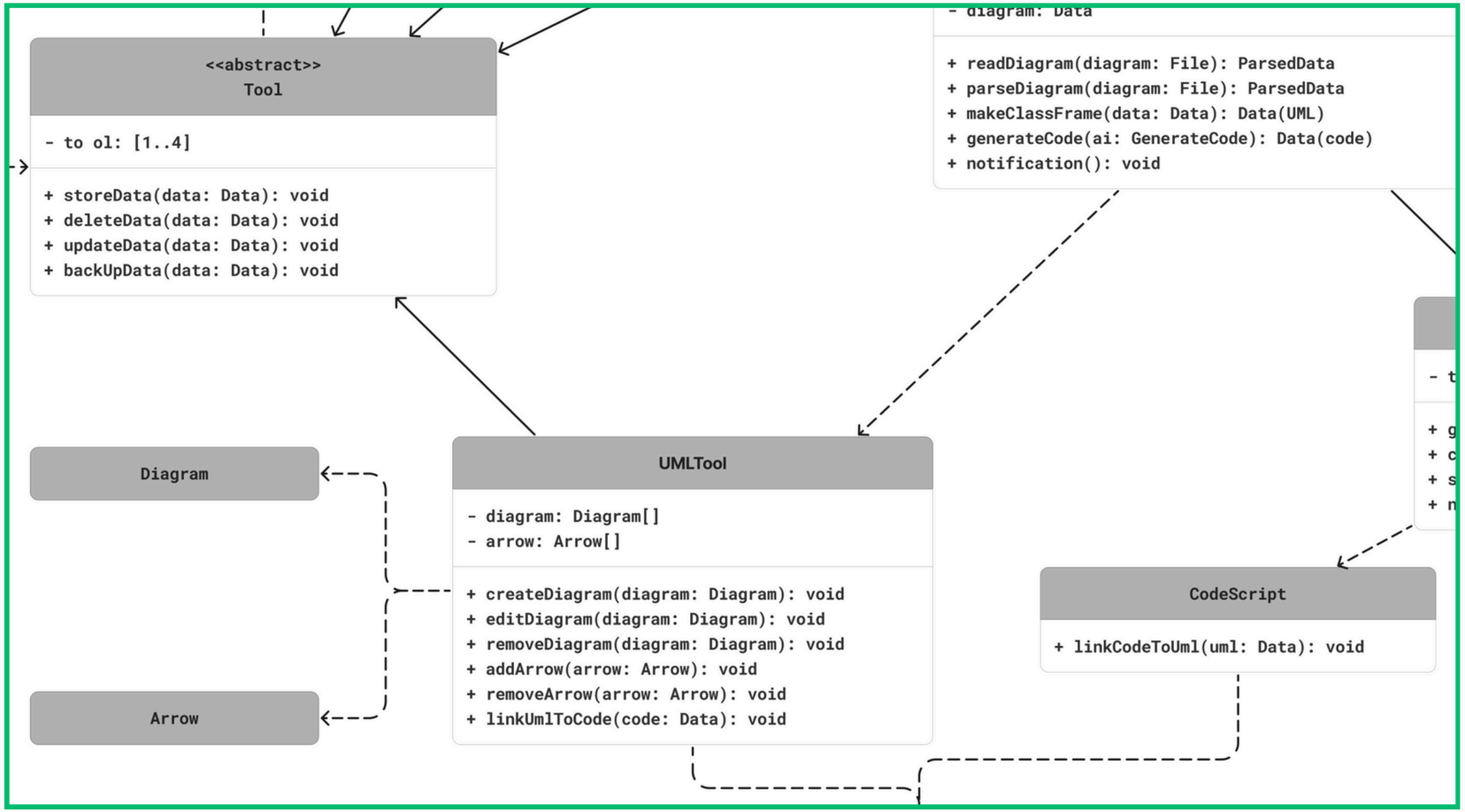
SECTION 04

디자인 패턴

디자인 패턴


Teamspace Diagram
TeamSpaceData
- membershipData: Data
- memberData: Data
- settingData: Data
- documentData: Data
- graphData: Data
- scheduleData: Data
- chatData: Data
- codeFileData: Data
- repositoryData: Data
+ storeData(): void
+ deleteData(): void
+ updateData(): void
+ backUpData(): void
Data
ScheduleTool
DocumentTool
GraphTool
Repository
UI
- userInterface: GUI
+ displayUI(): void
+ uploadDiagram(): void
+ requestConversion(): void
Server
- workflowData: Object
+ manageWorkflow(): void
+ routeRequest(request: Request): void
TeamSpace
- tool: Tool
+ useChatting(): void
+ settingMembership(): void
+ inviteMember(): void
+ settingTeamSpace(): void
+ moveToCodeScript(): void
+ accessTool(tool: Tool): void
+ changeRepository(repository: Repository): void
+ cloneRepository(repository: Repository): void
+ manageData(): void
<<abstract>>
Tool
- to ol: [1..4]
+ storeData(data: Data): void
+ deleteData(data: Data): void
+ updateData(data: Data): void
+ backUpData(data: Data): void
TransferUMLToCode
+ readDiagram(diagram: File): ParsedData
+ parseDiagram(diagram: File): ParsedData
+ makeClassFrame(data: Data): Data(UML
+ generateCode(ai: GenerateCode): Data(code)
+ notification(): void
Logger
User
CodeScript
Diagram
Arrow
UMLTool
- diagram: Diagram[]
- arrow: Arrow[]
+ createDiagram(diagram: Diagram): void
+ editDiagram(diagram: Diagram): void
+ removeDiagram(diagram: Diagram): void
+ addArrow(arrow: Arrow): void
+ removeArrow(arrow: Arrow): void
+ linkUmlToCode(code: Data): void
GenerateCode
- templates: CodeTemplates
+ generateCode(): void
+ callOpenAI(openAI : OpenAI): void
+ saveCode(code: CodeScript): void
+ notification(): void
CodeScript
+ linkCodeToUml(uml: Data): void
OpenAI
- apiCredentials: APIKeys
+ callOpenAPI(data: StructuredData): APIData
+ handleResponse(response: APIResponse): Code
UMLCodeLink
- isCodeExist: boolean
- isUmlExist: boolean
+ findMatchingCode(className: String, Data: Data): boolean
+ moveToCodeScript(className: String, isCodeExist: boolean): void
+ findMatchingUML(className: String, Data: Data): boolean
+ moveToUMLPage(className: String, isCodeExist: boolean): void

디자인 패턴

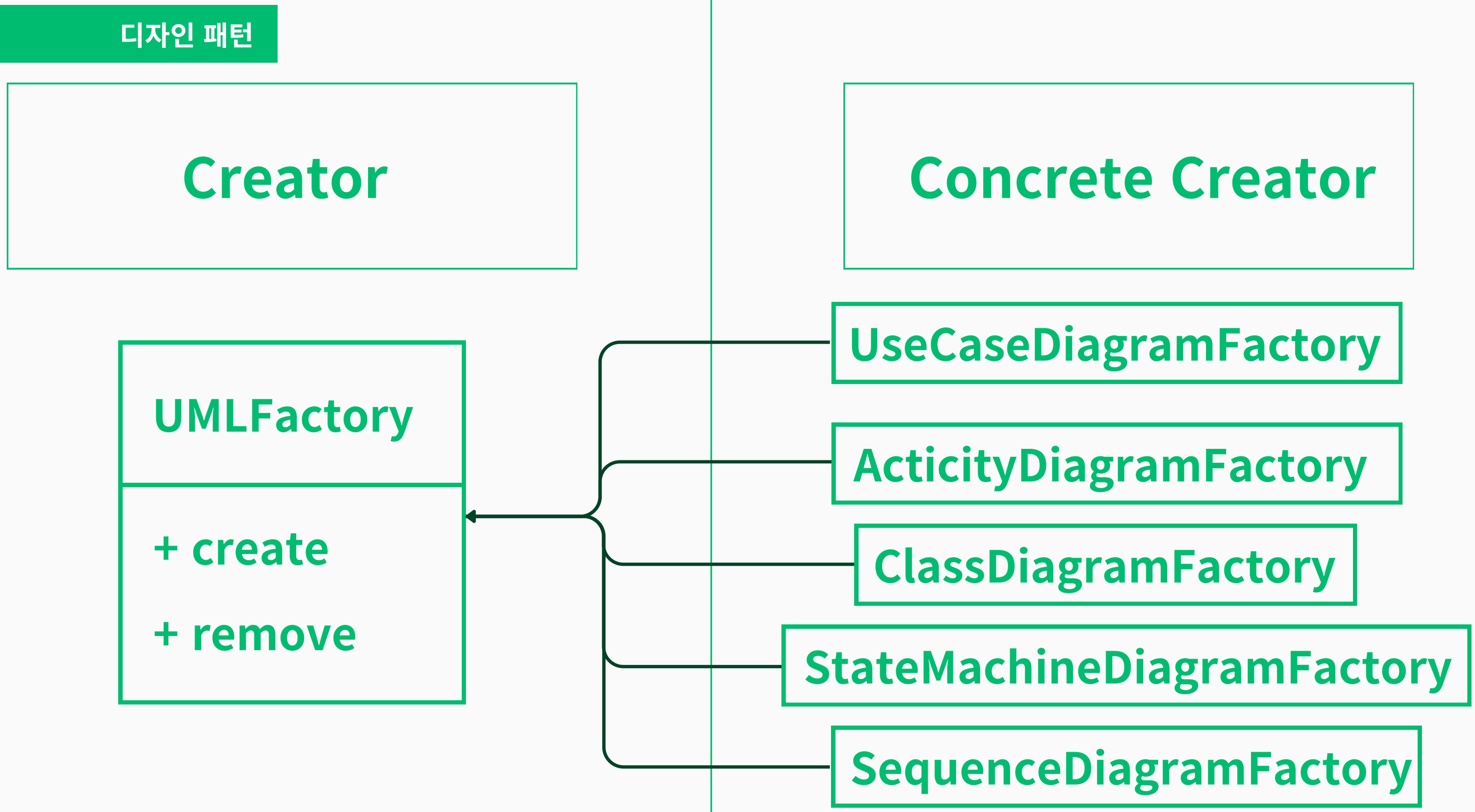

- diagram: Data
+ readDiagram(diagram: File): ParsedData
+ parseDiagram(diagram: File): ParsedData
+ makeClassFrame(data: Data): Data(UML)
+ generateCode(ai: GenerateCode): Data(code)
+ notification(): void
<<abstract>>
Tool
- to ol: [1..4]
+ storeData(data: Data): void
+ deleteData(data: Data): void
+ updateData(data: Data): void
+ backUpData(data: Data): void
Diagram
Arrow
UMLTool
- diagram: Diagram[]
- arrow: Arrow[]
+ createDiagram(diagram: Diagram): void
+ editDiagram(diagram: Diagram): void
+ removeDiagram(diagram: Diagram): void
+ addArrow(arrow: Arrow): void
+ removeArrow(arrow: Arrow): void
+ linkUmlToCode(code: Data): void
CodeScript
+ linkCodeToUml(uml: Data): void

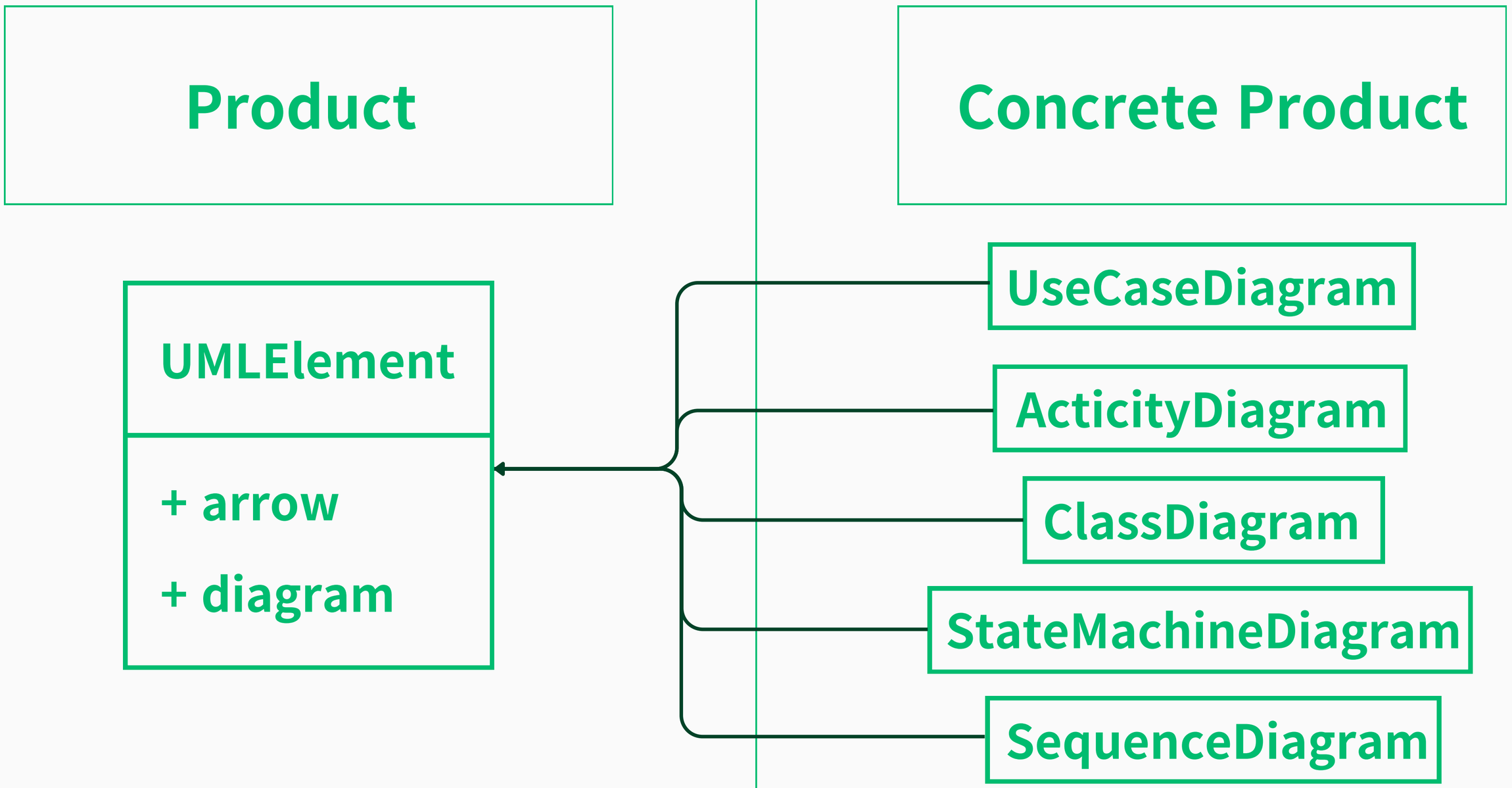
Factory Method

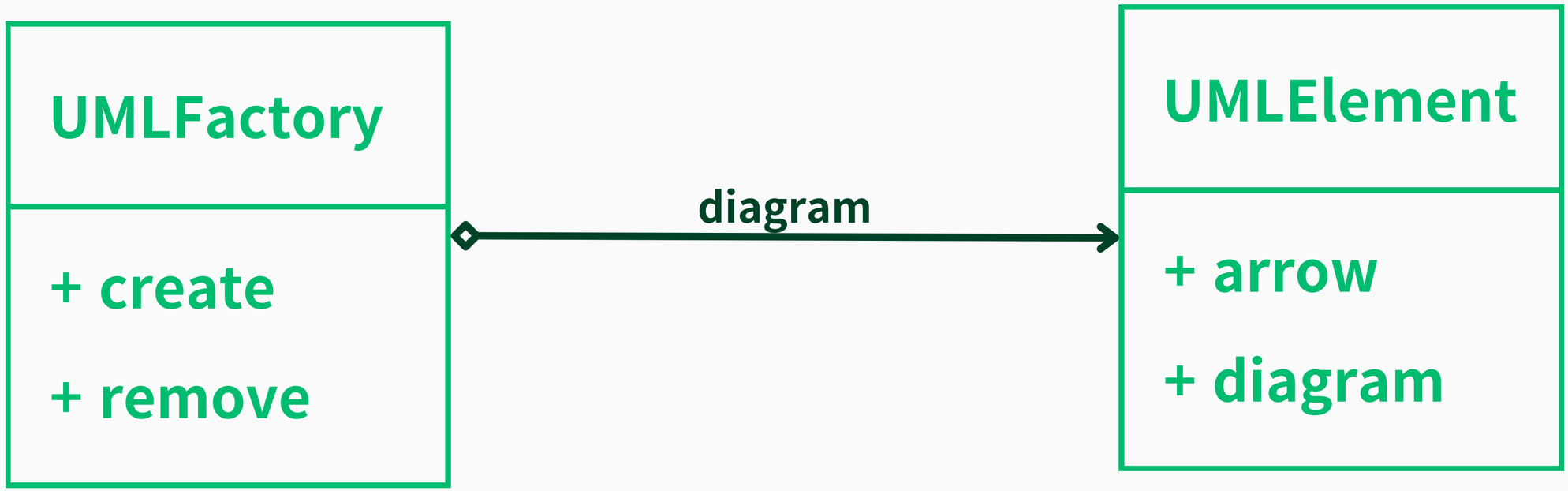
객체 생성은 미리하지만 객체의 구체적인 요소는 하위 클래스에서 결정하는 메소드

Creator class

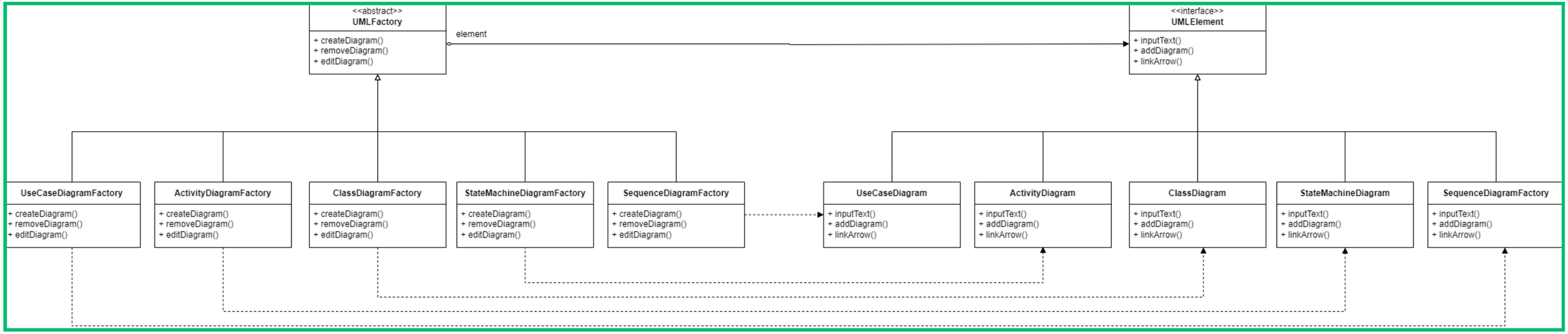
Product class


Creator
Concrete Creator
UMLFactory
+ create
+ remove
UseCaseDiagramFactory
ActicityDiagramFactory
ClassDiagramFactory
StateMachineDiagramFactory
SequenceDiagramFactory

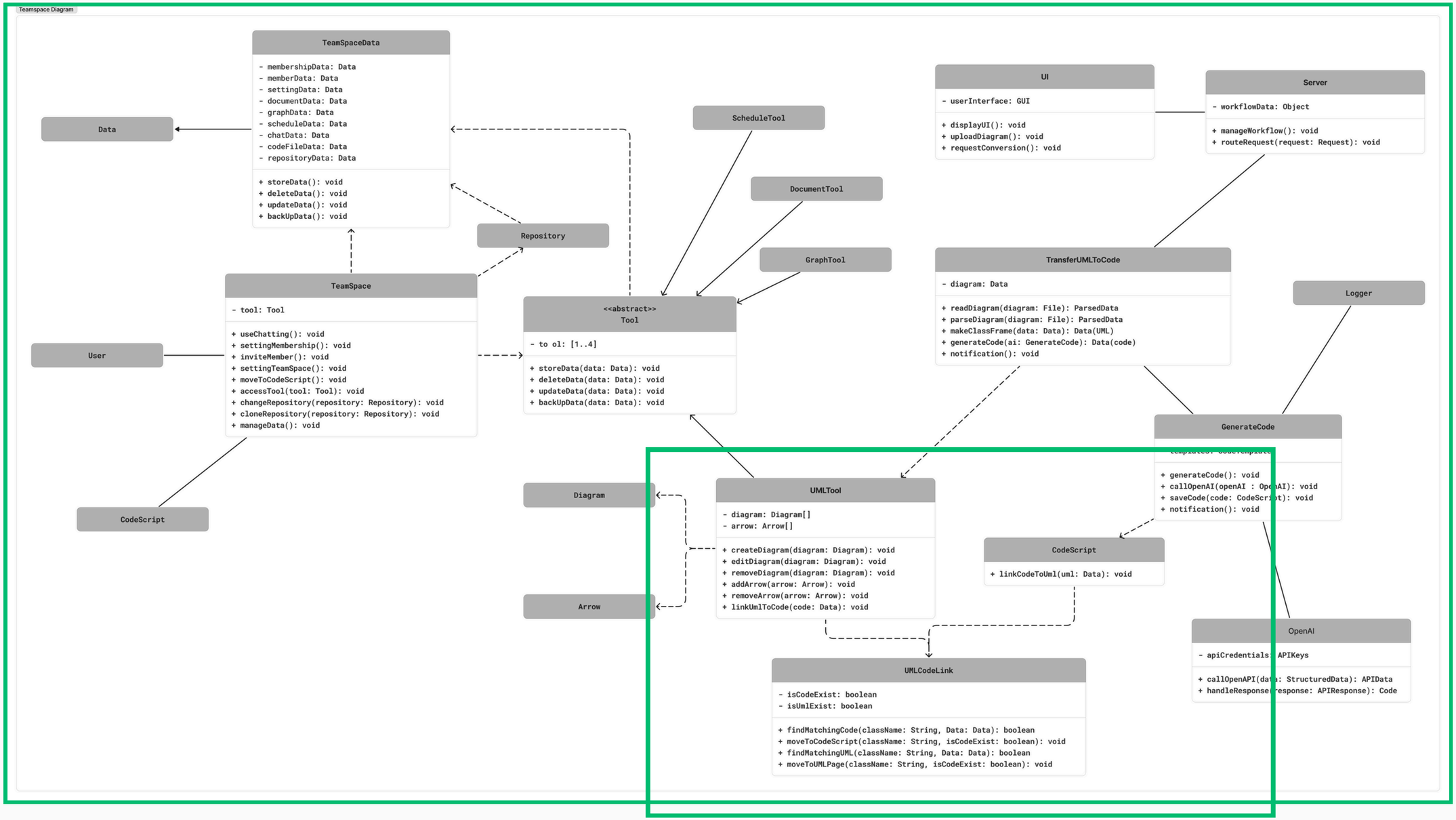

Product
Concrete Product
UMLElement
+ arrow
+ diagram
UseCaseDiagram
ActicityDiagram
ClassDiagram
StateMachineDiagram
SequenceDiagram


UMLFactory
+ create
+ remove
diagram
UMLElement
+ arrow
+ diagram

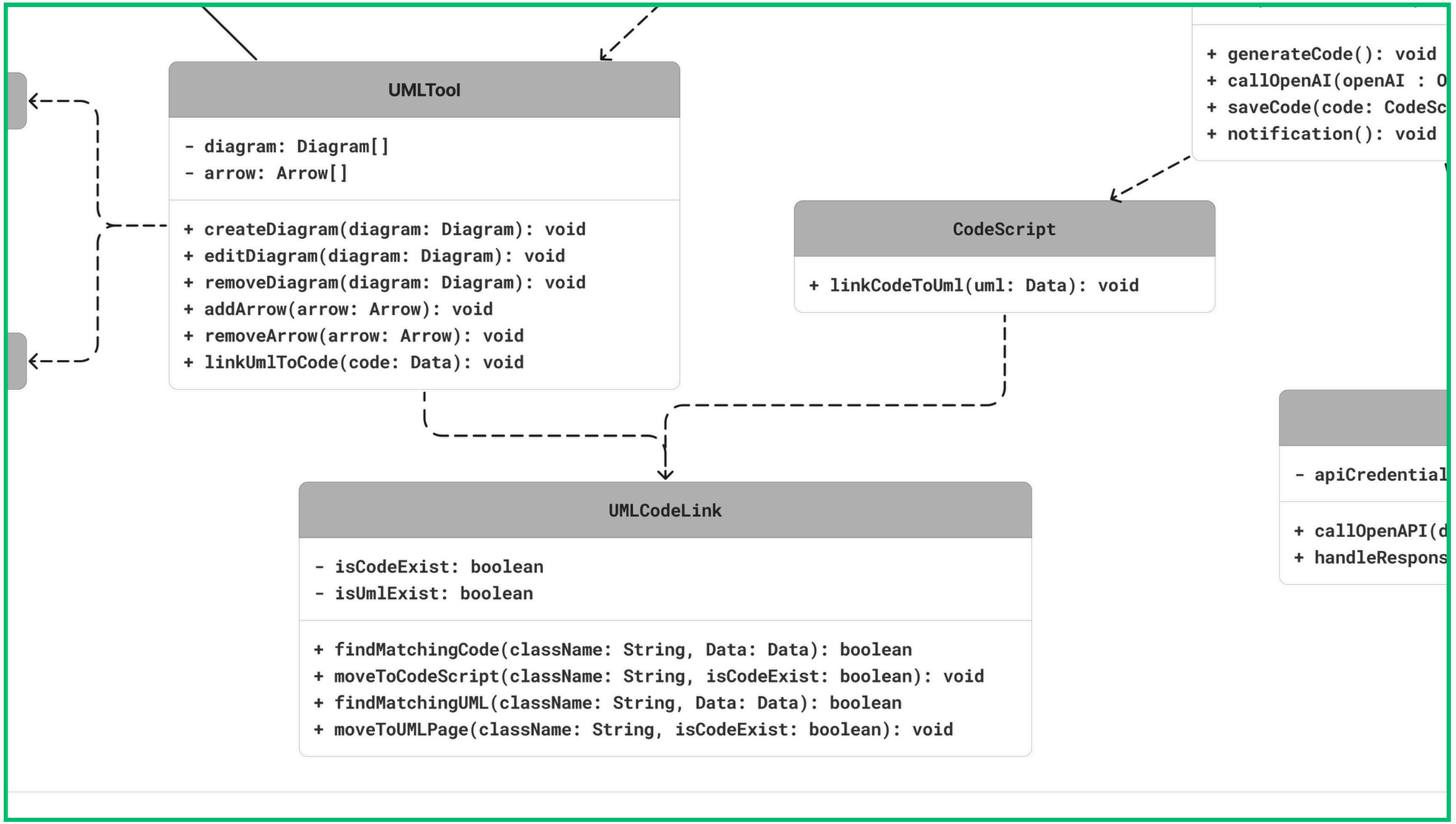
디자인 패턴


<<abstract>>
UMLFactory
+ createDiagram()
+ removeDiagram()
+ editDiagram()
element
<<interface>>
UMLElement
+ inputText()
+ addDiagram()
+ linkArrow()
UseCaseDiagramFactory
+ createDiagram()
+ removeDiagram()
+ editDiagram()
ActivityDiagramFactory
+ createDiagram()
+ removeDiagram()
+ editDiagram()
ClassDiagramFactory
+ createDiagram()
+ removeDiagram()
+ editDiagram()
StateMachineDiagramFactory
+ createDiagram()
+ removeDiagram()
+ editDiagram()
SequenceDiagramFactory
+ createDiagram()
+ removeDiagram()
+ editDiagram()
UseCaseDiagram
+ inputText()
+ addDiagram()
+ linkArrow()
ActivityDiagram
+ inputText()
+ addDiagram()
+ linkArrow()
ClassDiagram
+ inputText()
+ addDiagram()
+ linkArrow()
StateMachineDiagram
+ inputText()
+ addDiagram()
+ linkArrow()
SequenceDiagramFactory
+ inputText()
+ addDiagram()
+ linkArrow()

디자인 패턴


Teamspace Diagram
TeamSpaceData
- membershipData: Data
- memberData: Data
- settingData: Data
- documentData: Data
- graphData: Data
- scheduleData: Data
- chatData: Data
- codeFileData: Data
- repositoryData: Data
+ storeData(): void
+ deleteData(): void
+ updateData(): void
+ backUpData(): void
Data
Repository
ScheduleTool
DocumentTool
GraphTool
UI
- userInterface: GUI
+ displayUI(): void
+ uploadDiagram(): void
+ requestConversion(): void
Server
- workflowData: Object
+ manageWorkflow(): void
+ routeRequest(request: Request): void
TeamSpace
- tool: Tool
+ useChatting(): void
+ settingMembership(): void
+ inviteMember(): void
+ settingTeamSpace(): void
+ moveToCodeScript(): void
+ accessTool(tool: Tool): void
+ changeRepository(repository: Repository): void
+ cloneRepository(repository: Repository): void
+ manageData(): void
<<abstract>>
Tool
- to ol: [1..4]
+ storeData(data: Data): void
+ deleteData(data: Data): void
+ updateData(data: Data): void
+ backUpData(data: Data): void
TransferUMLToCode
- diagram: Data
+ readDiagram(diagram: File): ParsedData
+ parseDiagram(diagram: File): ParsedData
+ makeClassFrame(data: Data): Data(UML)
+ generateCode(ai: GenerateCode): Data(code)
+ notification(): void
Logger
User
CodeScript
GenerateCode
+ generateCode(): void
+ callOpenAI(openAI : OpenAI): void
+ saveCode(code: CodeScript): void
+ notification(): void
Diagram
Arrow
UMLTool
- diagram: Diagram[]
- arrow: Arrow[]
+ createDiagram(diagram: Diagram): void
+ editDiagram(diagram: Diagram): void
+ removeDiagram(diagram: Diagram): void
+ addArrow(arrow: Arrow): void
+ removeArrow(arrow: Arrow): void
+ linkUmlToCode(code: Data): void
CodeScript
+ linkCodeToUml(uml: Data): void
OpenAI
- apiCredentials: APIKeys
+ callOpenAPI(data: StructuredData): APIData
+ handleResponse(response: APIResponse): Code
UMLCodeLink
- isCodeExist: boolean
- isUmlExist: boolean
+ findMatchingCode(className: String, Data: Data): boolean
+ moveToCodeScript(className: String, isCodeExist: boolean): void
+ findMatchingUML(className: String, Data: Data): boolean
+ moveToUMLPage(className: String, isCodeExist: boolean): void

디자인 패턴


UMLTool
- diagram: Diagram[]
- arrow: Arrow[]
+ createDiagram(diagram: Diagram): void
+ editDiagram(diagram: Diagram): void
+ removeDiagram(diagram: Diagram): void
+ addArrow(arrow: Arrow): void
+ removeArrow(arrow: Arrow): void
+ linkUmlToCode(code: Data): void
CodeScript
+ linkCodeToUml(uml: Data): void
UMLCodeLink
- isCodeExist: boolean
- isUmlExist: boolean
+ findMatchingCode(className: String, Data: Data): boolean
+ moveToCodeScript(className: String, isCodeExist: boolean): void
+ findMatchingUML(className: String, Data: Data): boolean
+ moveToUMLPage(className: String, isCodeExist: boolean): void
+ generateCode(): void
+ callOpenAI(openAI : O
+ saveCode(code: CodeSc
+ notification(): void
- apiCredential
+ callOpenAPI(d
+ handleRespons

Mediator Method

서로 공유하는 기능을 제공하는 객체들이 직접 참조를 하지않게 하는 메소드

Mediator class

Colleague class

Mediator

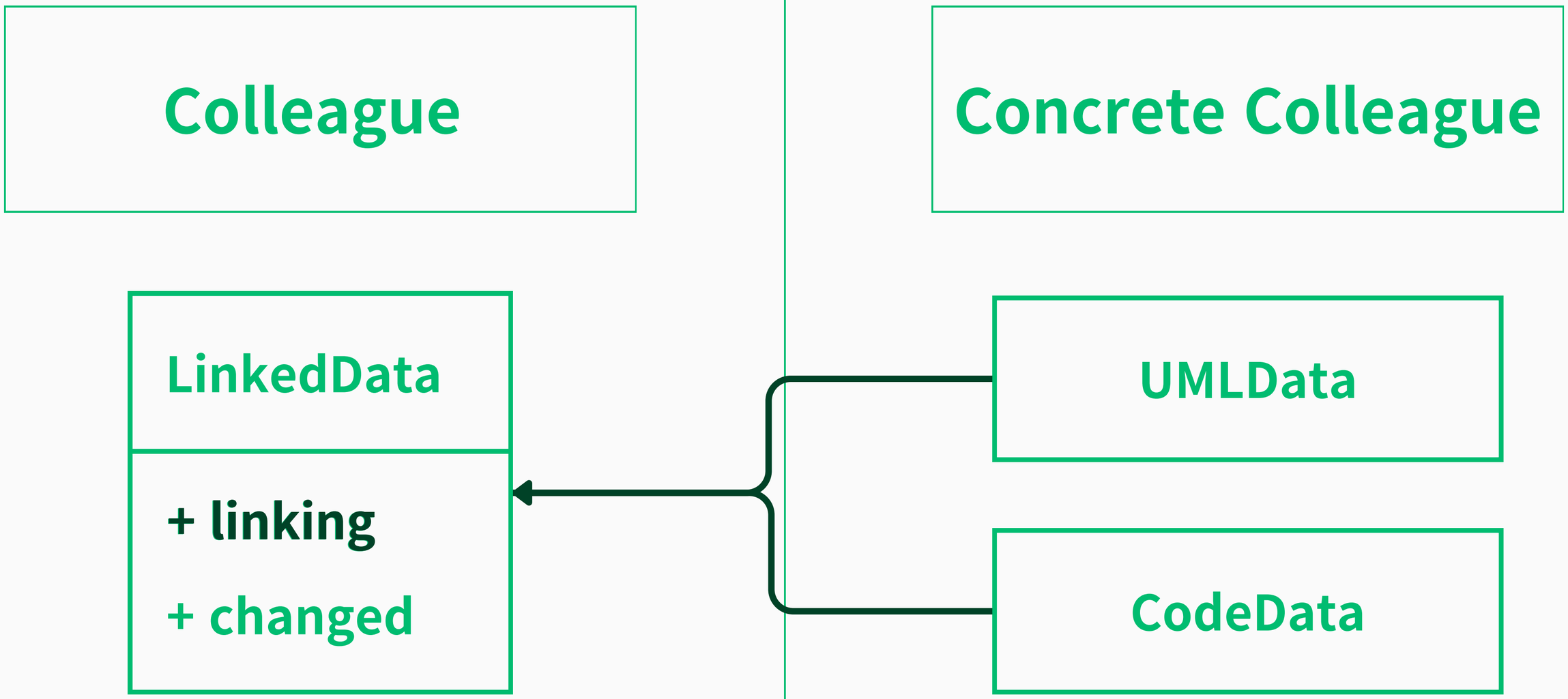
Concrete Mediator

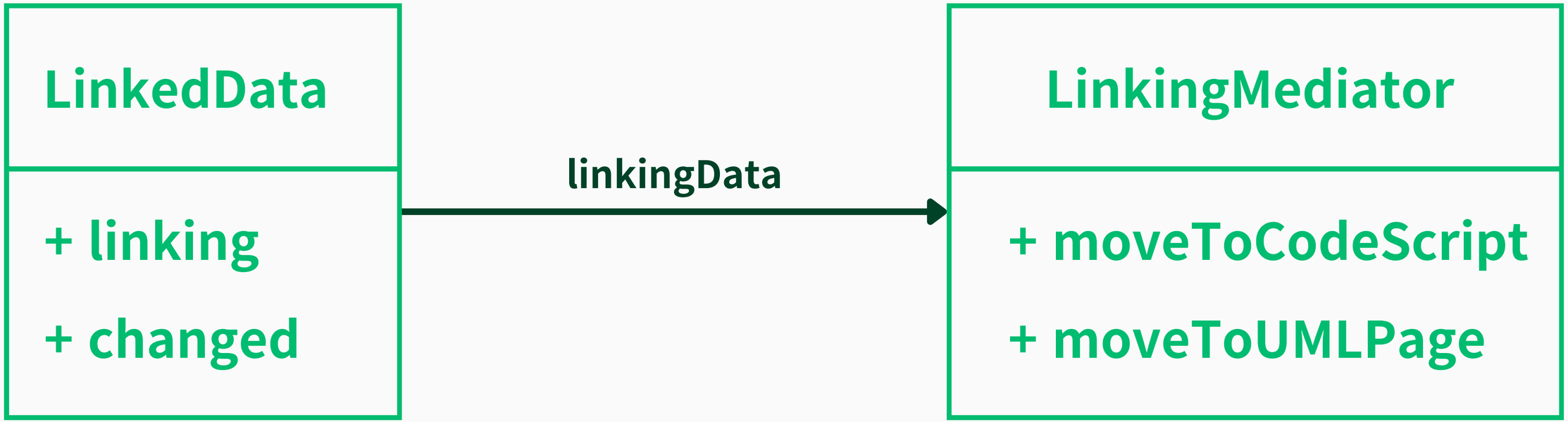
LinkingMediator

+ moveToCodeScript

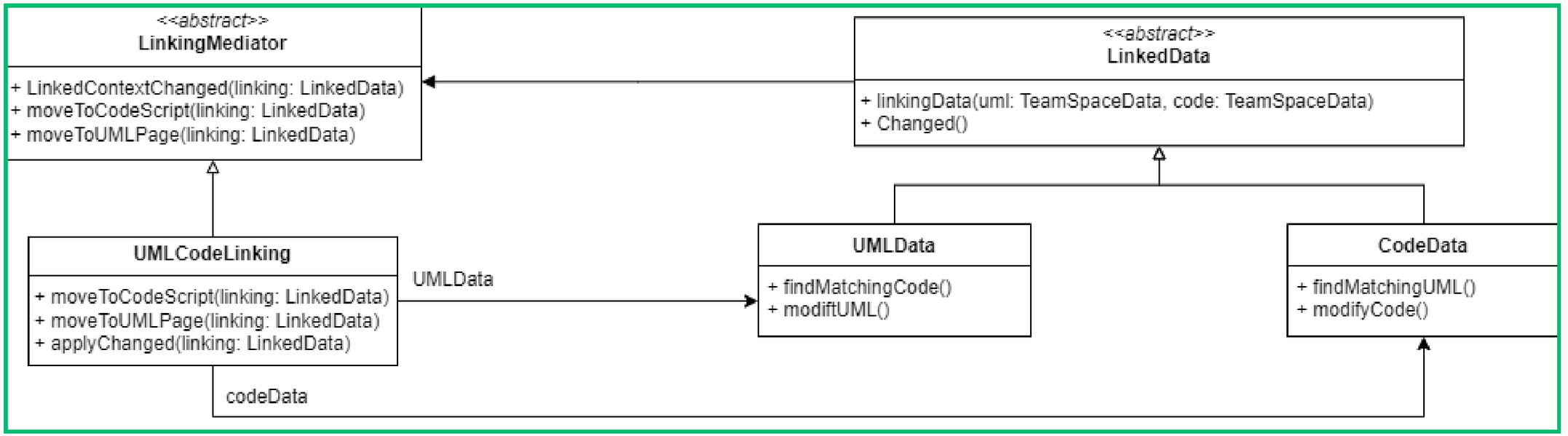
+ moveToUMLPage

UMLCodeLinking


Colleague
Concrete Colleague
LinkedData
+ linking
+ changed
UMLData
CodeData


LinkedData
+ linking
+ changed
linkingData
LinkingMediator
+ moveToCodeScript
+ moveToUMLPage

디자인 패턴


<<abstract>>
LinkingMediator
+ LinkedContextChanged(linking: LinkedData)
+ moveToCodeScript(linking: LinkedData)
+ moveToUMLPage(linking: LinkedData)
<<abstract>>
LinkedData
+ linkingData(uml: TeamSpaceData, code: TeamSpaceData)
+ Changed()
UMLCodeLinking
+ moveToCodeScript(linking: LinkedData)
+ moveToUMLPage(linking: LinkedData)
+ applyChanged(linking: LinkedData)
UMLData
UMLData
+ findMatchingCode()
+ modiftUML()
CodeData
+ findMatchingUML()
+ modifyCode()
codeData

SECTION 05

테스트 케이스 명세서

테스트 케이스 명세서

코드 정상 설정 테스트

테스트 케이스 이름	코드 룰 설정
시작 조건	사용자가 팀스페이스 멤버다 , 로그인상태에서 팀스페이스 화면에 진입해야 한다, 팀스페이스 사이드바에 있는 코드 스크립트를 선택 후에 코드 스크립트 화면에 진입해야 한다.
사건의 흐름	1.시스템이 코드 스크립트 화면과 사이드바를 표시한다 2.사용자가 코드 스크립트 사이드바에 존재하는 코드 룰 설정을 선택한다. 3.시스템이 코드 룰 설정 화면으로 이동하고 코드 룰 설정 화면을 표시한다. 4.시스템이 '코드 룰 생성 및 변경', '코드 룰 저장', '코드 룰 적용'을 차례로 표시한다. 5.사용자가 '코드 룰 생성 및 변경'을 선택하고 시스템은 코드 룰 생성 및 변경 화면을 표시한다. 6.사용자가 코드 룰 생성 및 변경 설정한다. 설정 완료를 선택한다. 7.사용자가 코드 룰 설정 화면으로 돌아가서 '코드 룰 저장'을 선택한다. 8.시스템은 설정된 코드 룰을 팀스페이스 데이터베이스에 저장한다. 그 후 시스템이 코드 룰이 저장 알림창을 표시한다. 9.사용자는 '코드 룰 적용'을 선택한다. 10. 시스템은 팀 스페이스 데이터베이스에 저장된 코드 룰 설정을 불러와서 코드 스크립트 환경에 적용한다. 그 후 시스템이 코드 룰 적용이 완료되었다는 알림창을 표시한다.
종료 조건	코드 룰 적용이 완료되었다는 알림창이 표시되거나 사용자가 다른 페이지로 이동한다.

테스트 케이스 명세서

코드 비정상 설정 테스트

테스트 케이스 이름	코드 룰 설정
시작 조건	사용자가 팀스페이스 멤버다 , 로그인상태에서 팀스페이스 화면에 진입해야 한다, 팀스페이스 사이드바에 있는 코드 스크립트를 선택 후에 코드 스크립트 화면에 진입해야 한다.
사건의 흐름	1.시스템이 코드 스크립트 화면과 사이드바를 표시한다. 2.사용자가 코드 스크립트 사이드바에 존재하는 코드 룰 설정을 선택한다. 3.시스템이 코드 룰 설정 화면으로 이동하고 코드 룰 설정 화면을 표시한다. 4.시스템이 ‘코드 룰 생성 및 변경’, ‘코드 룰 저장’, ‘코드 룰 적용’을 차례로 표시한다. 5.사용자가 코드 룰 생성하지 않고 ‘코드 룰 저장’을 선택한다. 6.생성된 코드 룰이 없기 때문에 시스템은 오류 메세지를 표시한다. 7.사용자가 코드 룰 생성하지 않고 ‘코드 룰 적용’을 선택한다. 8.생성된 코드 룰이 없기 때문에 시스템은 오류 메시지를 표시한다. 9.사용자가 코드 룰 저장을 하지 않고 ‘코드 룰 적용’을 선택한다. 10. 저장된 코드 룰이 없기 때문에 시스템은 오류 메시지를 표시한다.
종료 조건	코드 룰 적용이 완료되었다는 알림창이 표시되거나 사용자가 다른 페이지로 이동한다

THANK YOU