객체지향디자인패턴 과목안내

2023년 1학기

김기석 교수

시편 91:1~6

- 91:1 지존자의 은밀한 곳에 거주하며 전능자의 그늘 아래에 사는 자여,
- 91:2 나는 여호와를 향하여 말하기를 그는 나의 피난처요 나의 요새요 내가 의뢰하는 하나님이라 하리니
- 91:3 이는 그가 너를 새 사냥꾼의 올무에서와 심한 전염병에서 건지실 것임이로다
- 91:4 그가 너를 그의 깃으로 덮으시리니 네가 그의 날개 아래에 피하리로다 그의 진실함은 방패와 손 방패가 되시나니
- 91:5 너는 밤에 찾아오는 공포와 낮에 날아드는 화살과
- 91:6 어두울 때 퍼지는 전염병과 밝을 때 닥쳐오는 재앙을 두려워하지 아니하리로다

강의 계획서

1. 과목 기본 정보(Basic Course Information)

교과목명	객체지향 설계패턴		코드	ECE30012
개설년도	2023		개설학기	1
개설학부	전산전자공학부		이수구분/영역	전공선택/
대상학년	3		분반	01
인정전공	컴퓨터공학(33), 컴퓨터공학(40), 컴퓨터공학(45), AI·컴퓨터공학심화(60),			
학점구성	총학점	이론	실험/실습	설계	기타()
	3	1	0	2	0

수업주유형	☑ 강의 ☑ Project ☐ 토론 ☐ 실험 ☐ 실습			
선수과목	필 수 : 자바 프로그래밍 권 장 :		병수과목	
주관교수성명				
담당교수 성명	담당교수 Email	담당교수 전화	Office 위치	Office Hour
김기석	peterkim@handong.edu	1398	NTH 403	목 4,5교시
TA 성명			TA email	
강의실	NTH 417		강의시간	화2,금2

2. 학습목표 및 개요(Course Objectives)

● 학습목표(Course Objective)

번호	학습목표	Line +
1	The students can build applications and applets using advanced Java constructs.	Line -
2	The students can understand popular design patterns based on object-oriented	Line -
3	The students can design and implement a small application using OOP techniques	Line -

● 연관 학습성과(Related Learning Outcomes)

번호	학습성과
입력된 내용이 없습니다.	

● 강의 개요(Course Description)

- The lecture covers theoretical aspects of widely used design patterns to improve software reusability and maintainability.
- As a team, the students will build a software system to solve a design problem using the design patterns covered in this course.
- The student team presents and demonstrates the internal designs as well as the functions of their product.

● 교재

주교재				Line +
서명	Java 객체지향 디자인 패턴	저자	정인상, 채흥석	Line −
출판사	한빛미디어	출판년도	2017(3쇄)	

부교재				Line +
서명	Object-Oriented Software Engineering Using UML, Patterns, and Java™	저자	Bernd Bruegge & Allen H	Line −
출판사	Prentice Hall	출판년도	Third Edition	
서명	Java Design Patterns	저자		Line −
출판사	Exelixis Media P.C	출판년도	2015	
서명	Object-Oriented Software Development Using Java,	저자	Xiaoping Jia	Line −
출판사	Addison Wesley	출판년도	Second Edition	

기자재	

● 평가

출석관리	- More than six times of absence will result in failure in this course. - Twice of lateness will be counted by one absence. - Any use of mobile phone or ringing in the class will be counted as single lateness (first time) or absence (second time or later). - Any behavior disturbs the class can be penalized.							
학점산출 평가 도구 및 비중	출석	중간시험	기말시험	퀴즈	팀프로젝트	개인과제	기타1 (기타1)	기타2 (기타2)
Total(100%)	10	25	25	0	35	5		
Honor Code 준수 및 평가방법 추가설명	개인과제로 assign되는 프로그래밍 과제 제출시 타인의 코드를 일부라도 그대로 복사하여 사용하는 경우, 양쪽 모두에게 미제출로 채점됨							

● 수업 활동유형

강의	60%	실험	%	실습	%
팀프로젝트	30%	발표	10%	토론	%
기타1()	%	기타2()	%	기타3()	%
총계	100 %				

● 과제 및 프로젝트(Assignments and Projects)

번호	내용	
1	-팀(3-4명)별 텀프로젝트를 수행함 - 텀프로젝트는 iterative development방식을 통하여 단계별로 개발결과를 제시함 - 단계별로 리팩토링 또는 디자인패턴의 적용을 시도함 - 디자인 패턴을 정확히, 그리고 다수 적용할때 좋은 평가를 받게 될 것임	

설계 계획서 (설계학점)

- 첫번째 프로젝트
  - AmaterasUML을 이용, 주어진 문제를 객체지향으로 설계함
    - 한동대학교 도서관관리시스템
    - 대한항공 티켓팅 시스템
- 두번째 프로젝트
  - 팀에서 설정한 문제(첫번째 프로젝트)를 Iterative Development 방식으로 개발하면서 리팩토링 및 디자인패턴을 적용하여 개발함

컴퓨터공학 교과목 이수체계도

컴퓨터공학심화 교과목 이수체계도 (2019년 2학기부터)

(2019.06.04 기준)

(주1) ICT융합기초 과목군 - 파이썬 프로그래밍, 파이썬으로 배우는 기계학습, R을 이용한 빅데이터 분석
(주2) 과학 과목군 - 물리학2, 물리학 실험2, 일반생물학, 일반화학, 일반화학실험
(주3) 수학 과목군 - Calculus1, Calculus3, 미분방정식과 응용, 공학수학, 정수론, 실해석학개론

(주4) 실전프로젝트1->공학프로젝트 기획, 실전프로젝트2->캡스톤디자인 선수관계는 17학번부터 적용

김기석 교수

- 서울대학교 컴퓨터공학과 학사,석사,박사
- 삼성SDS 책임연구원
- 국회 정보통신정책보좌관

- 현) 2000년 8월 ~ 한동대학교 전산전자공학부 교수
- 현) 한동대학교 국제개발협력대학원 교수
- 현) 한동전문인선교연구소장

- ACTS(아세아연합신학대학원) M.A. Missiology 수료
- 풀러신학교 Visiting Scholar
- 전) IT전문선교단체 설립 및 초대본부장
- 현) 4차산업혁명과 기독교 포럼 공동대표
- 현) 한국텐트메이커포럼 운영위원

- 사무실 : 054-260-1398
- 휴대폰 : 010-8701-8301
- peterkim@handong.edu

진행안내

▶ 텀프로젝트 참여에 대한 기여

텀프로젝트 채점시 각 팀원의 기여도 상호평가

오프라인 모임이 시작하는 싯점과 연관

▶ 강의안 내 Blank 부분은 추후 보여주지 않을 것임

▶ 출석 체크 안내

8~9번 결석시 Fail

(가. 16주 수업시 - 주2회, 총 32회 수업을 한 경우 : 수업일수 32회의 1/4은 8회이기에 8회까지는 성적부여 가능하나 9회 결석부터는 F 부여)

출석점수 → 3주차부터 출석점수

- 지각 2번 1회 결석

특별한 사정으로 출석에 재고를 요할경우 해당사유와 함께 정정요청을 3일안에 해야합니다,

공결의 경우 공식적인 근거자료를 서류로 1주일 이내에 꼭 제출하여야 합니다)

교실강의 원칙/ 발표시 줌강의

▶ 교실강의 출석

교실출석 : 태깅

출석인정시간 강의10분전부터 강의시작 3분후까지,

지각인정시간-강의시작 3분후부 터 강의시작 5분후까지

▶ 줌강의

프로젝트 발표시 줌강의 진행

▶ 중간고사, 기말고사 시험본 사람은 출석으로 추후 처리함)

출석에서 "미확인" 으로 되어 있는 것은 결석임

- 동영상 강의 출석관련
  - 해당 수업시간 전에 온라인 강의를 올림

    출석 : 해당수업시간 및 그 다음날 저녁 11시 50분까지 강의 듣기 완료

    지각 : 다음 수업시간 직전까지 완료

    결석 : 듣지 않거나, 그 이후에 들었을때

공결처리 안내

가) **직계가족의 사망 및 사고, 학생 입원 또는 통원치료, 예비군훈련**은 첨부1번 파일과 함께 학생이 제출하는 증빙서류를 반드시 해당사안 전후 1주일 이내에 꼭 제출할 것

- 담임교수, 학부장 싸인을 꼭 받아야 함
- 싸인 받은 이후 (해당 사안 전후 1주일 이내) 제출할 것

나) 첨부 2번의 공결은 학부견학, 조기취업 등의 학부의 공식적인 허락 하에 공결처리 되는 건임. 이는 교무처에 제출하여 허락을 받은뒤, 해당 서류를 제출할 것

- 5. 졸업여행, 교육실습, 야외실습 및 각 학과(부) 학술여행
- 6. 정부기관의 요청에 의한 동원 및 특별회합
- 7. 학생활동 부서임원의 국제회합 및 이에 준하는 사유
- 8. 졸업예정자로서 국가 또는 기업체의 채용시험에 응시하는 사유
- 9. 졸업예정자(8학기이상 등록자)의 조기취업 등으로 인한 사유

※ 공결처리를 하기 원할 경우, 서류를 반드시 제대로 만들어 제출
그렇지 않은 경우, 나에게 직접 설명하면 출석으로 처리할수도 있음

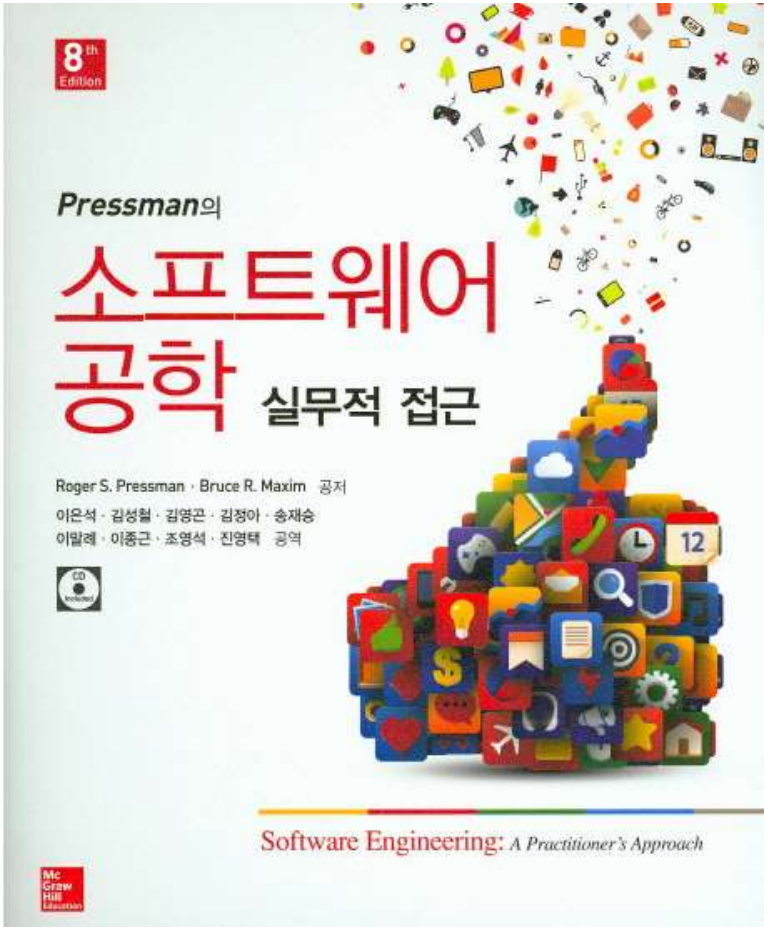
강의

방향

- 객체지향기반 개발 & UML 모델링 (20%)
- 리팩토링 (5%)
- 디자인패턴 (75%)
- Iterative development
- 설계프로젝트 (팀을 통한 실습)

- (X) Software Engineering
- (ㅇ) 객체지향기반 소프트웨어 개발

“소프트웨어공학”과의 비교



- CHAPTER 01 소프트웨어의 본질
  CHAPTER 02 소프트웨어 공학

  PART 1 소프트웨어 프로세스
  CHAPTER 03 소프트웨어 프로세스 구조
  CHAPTER 04 프로세스 모델
  CHAPTER 05 애자일 개발
  CHAPTER 06 소프트웨어 공학의 인간적인 측면

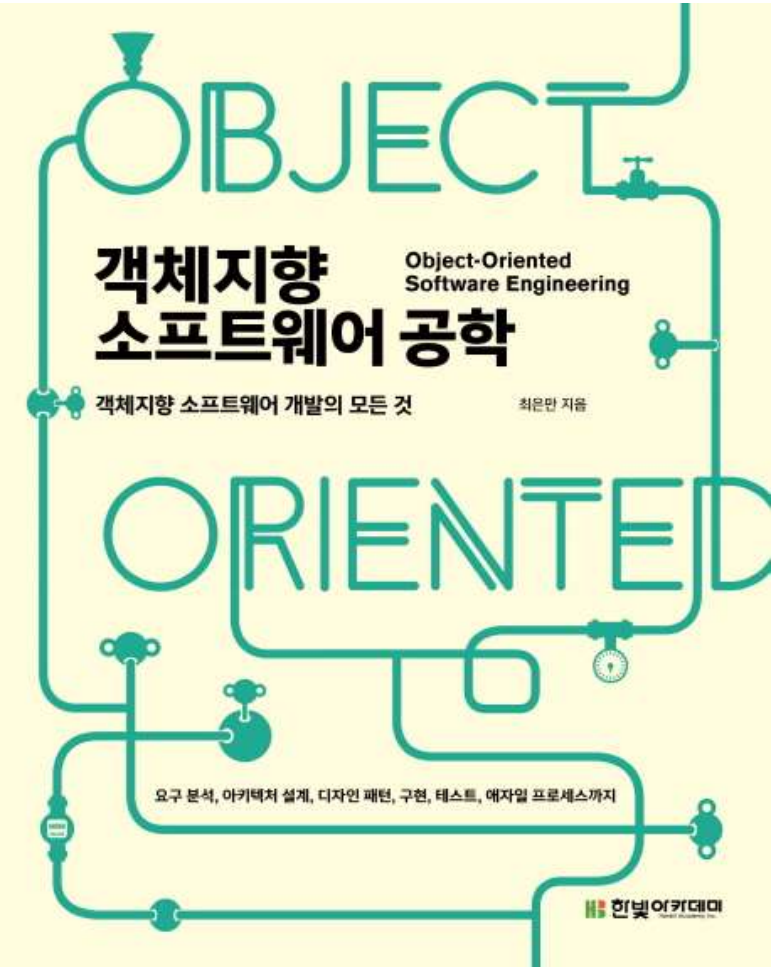
- PART 2 모델링
  CHAPTER 07 실무가이드 원칙
  CHAPTER 08 요구사항 이해
  CHAPTER 09 요구사항 모델링: 시나리오 기반 방법론
  CHAPTER 10 요구사항 모델링: 클래스 기반 방법론
  CHAPTER 11 요구사항 모델링: 동작, 패턴 그리고 웹 응용 소프트웨어
  CHAPTER 12 설계 개념
  CHAPTER 13 아키텍처 설계
  CHAPTER 14 컴포넌트-수준 설계
  CHAPTER 15 사용자 인터페이스
  CHAPTER 16 패턴 기반 설계
  CHAPTER 17 웹앱 설계
  CHAPTER 18 모바일 앱 설계

  PART 3 품질 관리
  CHAPTER 19 품질 개념
  CHAPTER 20 검토 기술
  CHAPTER 21 소프트웨어 품질 보증
  CHAPTER 22 소프트웨어 테스팅 전략
  CHAPTER 23 전통적인 애플리케이션 테스팅
  CHAPTER 24 객체지향 응용프로그램 테스팅
  CHAPTER 25 웹앱 테스팅
  CHAPTER 26 모바일 앱 테스팅
  CHAPTER 27 보안성 공학
  CHAPTER 28 정형 모델링과 검증
  CHAPTER 29 소프트웨어 형상관리
  CHAPTER 30 제품 척도

  PART 4 소프트웨어 프로젝트 관리
  CHAPTER 31 프로젝트 관리의 개념
  CHAPTER 32 프로세스와 프로젝트 척도
  CHAPTER 33 소프트웨어 프로젝트의 산정
  CHAPTER 34 프로젝트 스케쥴링
  CHAPTER 35 위험 관리

  PART 5 고급 주제
  CHAPTER 36 유지보수와 재공학
  CHAPTER 37 소프트웨어 프로세스 개선
  CHAPTER 38 소프트웨어공학의 최신경향
  CHAPTER 39 결론

“객체지향 소프트웨어공학”과의 비교



- Chapter 01 소프트웨어 공학의 소개
  00 개요
  01 소프트웨어란?
  02 소프트웨어 공학이란?
  03 소프트웨어 품질
  04 소프트웨어 프로젝트의 유형
  05 소프트웨어 프로젝트 작업
  06 객체지향 소프트웨어 공학
  07 최신 소프트웨어 공학의 핵심 기술
  요약/연습문제

  Chapter 02 객체지향의 개념
  00 개요
  01 객체지향이란?
  02 클래스와 객체
  03 인스턴스 변수, 클래스 변수, 메소드, 오퍼레이션
  04 상속
  05 다형성
  06 UML
  요약/연습문제

  Chapter 03 요구 분석
  00 개요
  01 도메인 분석
  02 문제 정의와 범위 설정
  03 요구 추출
  04 요구 추출 방법
  05 사용 사례 분석
  06 요구 문서화
  07 요구 검토
  08 [사례 연구] 내비게이션 시스템
  요약/연습문제

- Chapter 04 클래스 모델링
  00 개요
  01 클래스 다이어그램의 기초
  02 클래스와 가시성
  03 연관 관계와 다중도
  04 일반화 관계와 전체/부분 관계
  05 클래스 다이어그램의 고급 표현
  06 클래스 다이어그램 작성 과정
  07 코드 매핑
  요약/연습문제

  Chapter 05 동적 모델링
  00 개요
  01 시퀀스 다이어그램
  02 커뮤니케이션 다이어그램
  03 상태 다이어그램
  04 액티비티 다이어그램
  05 동적 다이어그램의 구현
  요약/연습문제

  Chapter 06 아키텍처 설계
  00 개요
  01 설계 과정
  02 설계 원리
  03 설계안 결정
  04 아키텍처 모델
  05 패키지 다이어그램과 배치 다이어그램
  06 아키텍처 패턴
  07 아키텍처 문서화
  요약/연습문제

- Chapter 07 디자인 패턴
  00 개요
  01 디자인 패턴 소개
  02 기본 패턴
  03 생성 패턴
  04 구조 패턴
  05 행위 패턴
  요약/연습문제

  Chapter 08 구현
  00 개요
  01 구현이란?
  02 코딩 원리
  03 코딩 표준
  04 설계와의 매핑
  05 리팩토링
  06 검토
  요약/연습문제

  Chapter 09 테스트
  00 개요
  01 테스트란?
  02 블랙박스 테스트
  03 화이트박스 테스트
  04 객체지향 테스트
  05 통합 및 시스템 테스트
  06 테스트 관리
  07 테스트 자동화 도구
  요약/연습문제

- Chapter 10 유지보수
  00 개요
  01 유지보수란?
  02 유지보수 작업 단계와 모델
  03 형상 관리
  04 리엔지니어링
  05 유지보수 도구
  요약/연습문제

  Chapter 11 프로젝트 관리
  00 개요
  01 프로젝트 관리란?
  02 소프트웨어 개발 프로세스
  03 개발 노력의 추정
  04 일정 계획과 관리
  05 팀 구성
  06 프로젝트 계획서
  요약/연습문제

- Chapter 12 품질 보증
  00 개요
  01 품질이란?
  02 품질 보증
  03 확인과 검증
  04 품질 측정
  05 프로세스 품질 개선
  06 품질 보증 계획과 품질 제어
  요약/연습문제

  부록
  01 UML 표현 방법
  02 용어 설명

Iterative Development

- Jia 9.1 ~ 9.6
  - Scribble1,Scribble2, Scribble3
  - Draw1 , Draw2 , Draw3
- Jia 10.1 ~ 10.3
  - SimpleMazeGame(iter#1)
  - MazeGameAbstractFactory(iter#2)
  - MazeGameCreator(Iter#3)
  - MazePrototypeFactory(iter#4)
  - MazeGameBuiler(iter#5)
  - UndoableMazeGame(iter#6)

Iterative development 기반 팀 프로젝트 (설계)

---

0) Proposal(#1): 7th week(4/10 월 저녁6시까지)
1) Proposal(#2): 8th week(4/21 목 저녁6시까지)
2) 1st iteration : 10th Week(5/1 월 저녁6시까지)
3) 2nd iteration : 12th week(5/15 월 저녁6시까지)
4) 3rd iteration : 14th week(5/29 월 저녁6시까지)
5) Final iteration : 15th week(6/05 월 저녁6시까지)
   . 최종 presentation & demonstration: 15th week중
   . final report: 17th week(6/19일 월 저녁6시까지)

팀 구성

- 3주차 싯점에서 팀 구성
  - 임의 할당 방식
  - 4명 또는 3명

- 팀활동 예
  - 디자인패턴 1~2개씩 발표 (java design patterns)
  - 기여도 상호 평가 (기말고사)

리팩터링(refactoring)

- 리팩터링(refactoring)은 소프트웨어 공학에서 '결과의 변경 없이 코드의 구조를 재조정함'을 뜻한다. 주로 가독성을 높이고 유지보수를 편하게 한다. 버그를 없애거나 새로운 기능을 추가하는 행위는 아니다. 사용자가 보는 외부 화면은 그대로 두면서 내부 논리나 구조를 바꾸고 개선하는 유지보수 행위이다.
- 리팩터링의 잠재적인 목표는 소프트웨어의 설계, 구조 및 구현을 개선하는 동시에 소프트웨어의 기능을 보존하는 것이다. 리펙터링은 코드의 가독성을 향상시키고 복잡성을 감소시키는 효과를 가지며, 이러한 이점은 소스 코드의 유지 보수성을 개선하고 확장성을 개선하기 위해 더 단순하고, 깔끔하거나, 표현력이 뛰어난 내부 아키텍처 또는 객체 모델을 만들 수 있게 한다. 그리고 소프트웨어 엔지니어는 더 빠르게 수행되거나 더 적은 메모리를 사용하는 프로그램을 작성해야 하는 지속적인 과제에 직면해 있기에 성능 향상이 리팩터링의 또다른 목표가 된다.

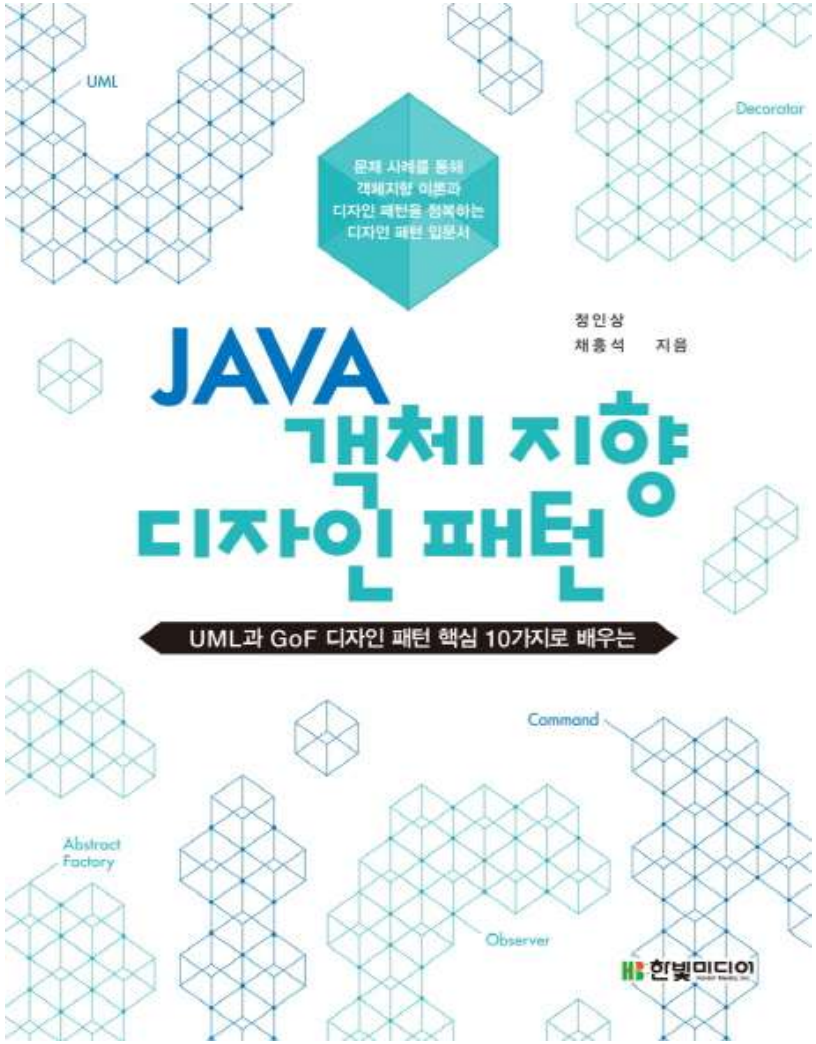
Refactoring 교과서 사례

- Jia 9.3 Refactoring
    - Scribble3.java

가능한한 개인별 구매 추천

주교재

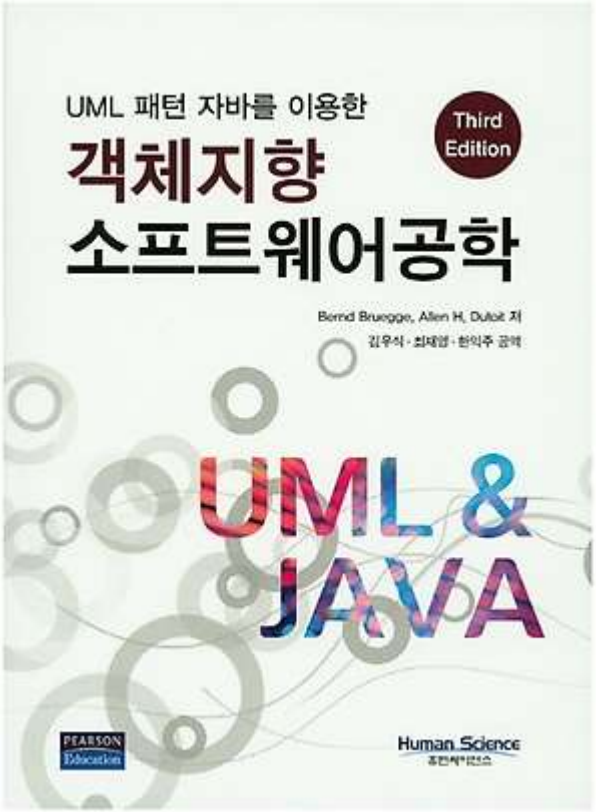
JAVA 객체지향 디자인 패턴(###)
-UML과 GoF 디자인 패턴 핵심 10가지로 배우는
(@@@) (2014)

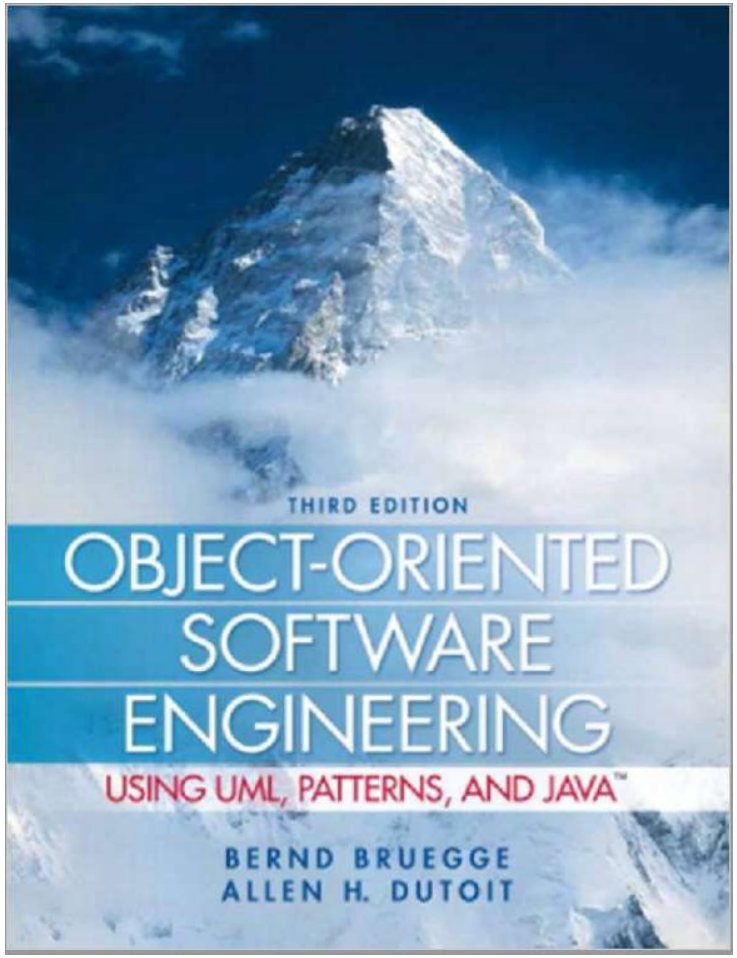


가능한한 개인별 구매 추천

주교재

UML 패턴 자바를 이용한 객체지향 소프트웨어공학(2014)

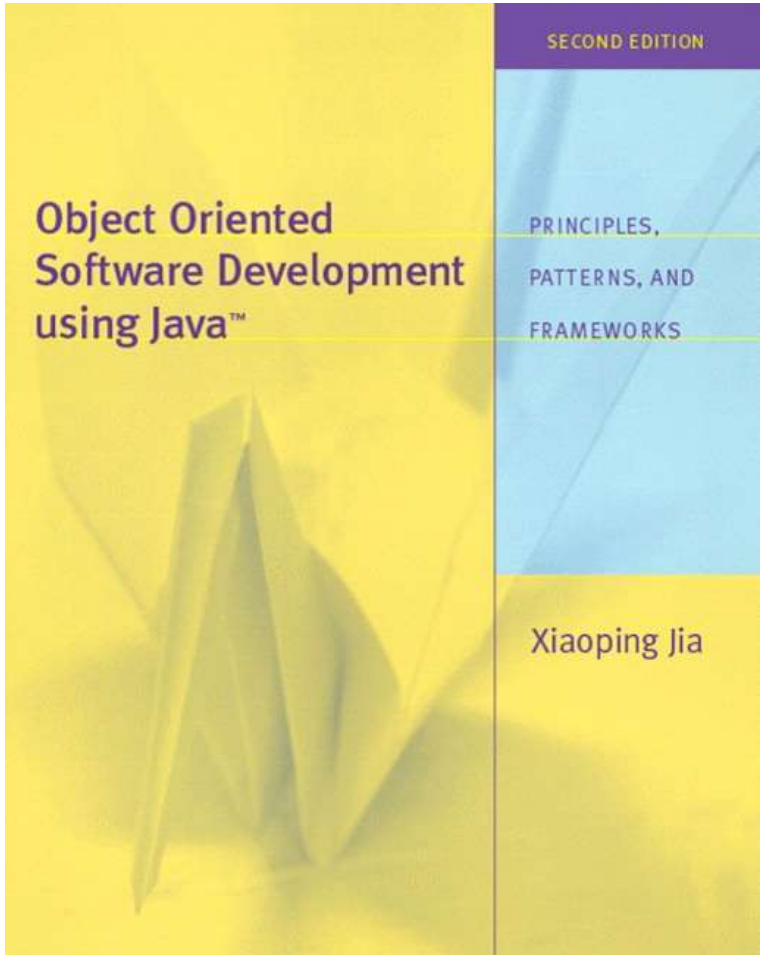



THIRD EDITION
OBJECT-ORIENTED
SOFTWARE
ENGINEERING
USING UML, PATTERNS, AND JAVA™
BERND BRUEGGE
ALLEN H. DUTOIT

구매 불필요

부교재

Object-Oriented Software Development Using Java - Principles, Patterns, and Frameworks, (2nd edition)



Chapter 1: Object-Oriented Software Development

1.1: The Challenges of Software Development

1.2: An Engineering Perspective

1.3: Object Orientation

1.4: Iterative Development Processes: RUP and XP

Chapter 2: Object-Oriented Modeling using UML

2.1: Principles and Concepts

2.2: Modeling Relationships and Structures

2.3: Modeling Dynamic Behaviors

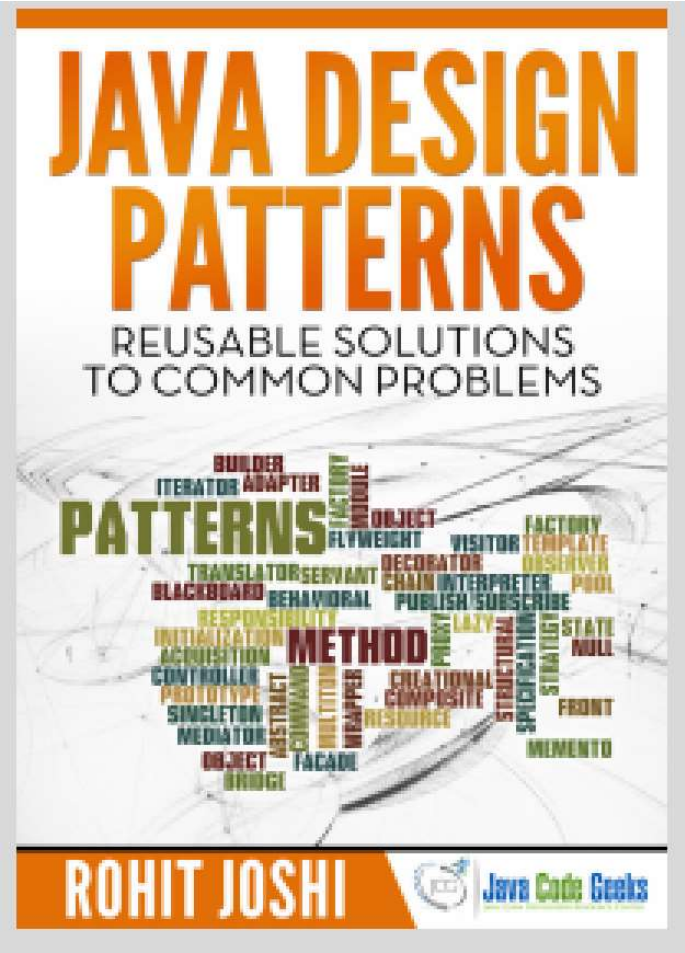
2.4: Modeling Requirements with Use Cases

2.5: Case Study: An E-Bookstore

Jia-다루는 디자인 패턴

- Chap07A - 싱글레톤(7.1.1), 리팩토링 (7.2.1),
- Chap07B -리팩토링 (7.2.1),템플레이트 메소드 (7.2.2)
- Chap07C - 스트래티지 (7.2.4)
- Chap07D - 아이터레이터(7.3.2) 팩토리 (7.4.3)
- Chap07E
- Chap08A - 아이터레이터 (8.2.4)
- Chap08B- 컴포지트 (8.3.2)
- Chap08C - 데코레이터 (8.4..2)
- Chap09A
- Chap09B- 스테이트(9.5.3)
- Chap09C-팩토리 메소드(9.5.6)
- Chap10A1-애브스트랙트 팩토리(10.2.2)
- Chap10A2-팩토리메소드(10.2.3) 프로토타입(10.2.4) 빌더 (10.2.5)
- Chap10B1-커맨드(10.3.1) 어댑터(10.4.1)
- Chap10B2-컴포지트(10.4.2)

부교재



- 1 Introduction to Design Patterns
- 2 Adapter Design Pattern
- 3 Facade Design Pattern
- 4 Composite Design Pattern
- 5 Bridge Design Pattern
- 6 Singleton Design Pattern
- 7 Observer Design Pattern
- 8 Mediator Design Pattern

- 9 Proxy Design Pattern
- 10 Chain of Responsibility Design Pattern
- 11 Flyweight Design Pattern
- 12 Builder Design Pattern
- 13 Factory Method Design Pattern
- 14 Abstract Factory Method Design Pattern
- 15 Prototype Design Pattern
- 16 Memento Design Pattern
- 17 Template Design Pattern
- 18 State Design Pattern
- 19 Strategy Design Pattern
- 20 Command Design Pattern
- 21 Interpreter Design Pattern
- 22 Decorator Design Pattern
- 23 Iterator Design Pattern
- 24 Visitor Design Pattern

(iterative developement)

참고

도서

실전 개발을 위한 자바 디자인 패턴(###$$)

1부 : 디자인패턴 (생성패턴)
1. 싱글턴(Singleton)
2. 빌더(Builder)
3. 팩토리 메서드(Factory Method)
4. 프로토타입(Prototype)
5. 추상팩토리(Abstract Factory)

2부 : 디자인패턴 (구조패턴)
1. 어댑터(Adapter)
2. 브리지(Bridge)
3. 컴포지트(Composite)
4. 데커레이터(Decorator)
5. 퍼사드(Facade)
6. 플라이웨이트(Flyweight)
7. 프록시(Proxy)

(###)

3부 : 디자인패턴 (행위패턴)
1. 책임 연쇄(Chain of Responsibillity)
2. 커맨드(Command)
3. 인터프리터(Interpreter)
4. 이터레이터(Iterator)
5. 미디에이터(Mediator)
6. 메멘토(Memento)
7. 옵저버(Observer)
8. 스테이트(State)
9. 스트래티지(Strategy)
10. 템플릿 메서드(Template Method)
11. 비지터(Visitor)

4부 : 실전프로젝트 (페인터)
1. 페인터 설명
2. 무작정 그려보기
3. 모델을 만들어 보자
4. Factory Method 패턴 적용하기
5. MVC(Model-View-Controller) 패턴 적용하기
6. Observer 패턴 적용하기
7. Strategy 패턴 적용하기
8. State 패턴 적용하기
9. Memento 패턴 적용하기

5부 : 실전프로젝트 (게임)
1. 게임 설명
2. 무작정 게임 만들기
3. 모델을 만들어 보자
4. MVC(Model-View-Controller) 패턴 적용하기
5. 타원형 스프라이트 만들기
6. 이미지 스프라이트를 위한 Flyweight 패턴 적용하기
7. 텍스트 스프라이트 만들기
8. Factory를 이용한 객체 생성하기
9. 컴포지트 패턴의 복합 스프라이트 만들기
10. Abstract Factory 패턴 적용하기
11. Strategy 패턴 적용하기
12. State 패턴 적용하기
13. 마우스 이벤트 처리하기
14. 키 이벤트 처리하기
15. 게임로직 구현하기

실전 개발을 위한 자바 디자인 패턴 2 (###$$)

1장 : 객체 지향
1. 클래스
2. 클래스 상속
3. 오버로딩과 오버라이딩
4. 추상클래스
5. 인터페이스
6. 인터페이스 구현

2장 : 클래스 관계와 UML
1. 상속(generalization)
2. 구현(realization)
3. 연관(association)
4. 집합(aggregation)
5. 합성(composition)
6. 의존(dependency)

3장 : 객체 지향 설계 원칙
1. 단일 책임 원칙
2. 개방 폐쇄 원칙
3. 리스코프 치환 원칙
4. 인터페이스 분리 원칙
5. 의존 역전 원칙

4장 : 생성 패턴
1. 싱글턴(Singleton)
2. 빌더(Builder)
3. 팩토리 메서드(Factory Method)
4. 프로토타입(Prototype)
5. 추상팩토리(Abstract Factory)

5장 : 구조 패턴
1. 어댑터(Adapter)
2. 브리지(Bridge)
3. 컴포지트(Composite)
4. 데커레이터(Decorator)
5. 퍼사드(Facade)
6. 플라이웨이트(Flyweight)
7. 프록시(Proxy)

6장 : 행위 패턴
1. 책임 연쇄(Chain of Responsibillity)
2. 커맨드(Command)
3. 인터프리터(Interpreter)
4. 이터레이터(Iterator)
5. 미디에이터(Mediator)
6. 메멘토(Memento)
7. 옵저버(Observer)
8. 스테이트(State)
9. 스트래티지(Strategy)
10. 템플릿 메서드(Template Method)
11. 비지터(Visitor)

7장 : 모델-뷰-XXX 패턴
1. 모델-뷰(Model-View)
2. MVC(Model-View-Controller)
3. MVP(Model-View-Presenter)
4. MVVM(Model-View-ViewModel)

8장 : 모던 자바 패턴
1. 지네릭스(Generics)
2. 열거형(Enums)
3. 애너테이션(Annotation)
4. 함수형 인터페이스(Functional Interface)
5. 스트림(Stream)

(###)

9장 : 계산기 프로젝트
1. 계산기 설명
2. 책임 연쇄 패턴 적용하기
3. 수학함수 피연산자 정의하기
4. 계산기 화면 만들기
5. 모델 만들기
6. Command 패턴 적용하기
7. 사칙연산 처리하기
8. State 패턴 적용하기
9. 계산 초기화
10. 수학함수 처리하기
11. +-부호 지정하기
12. 전체 소스

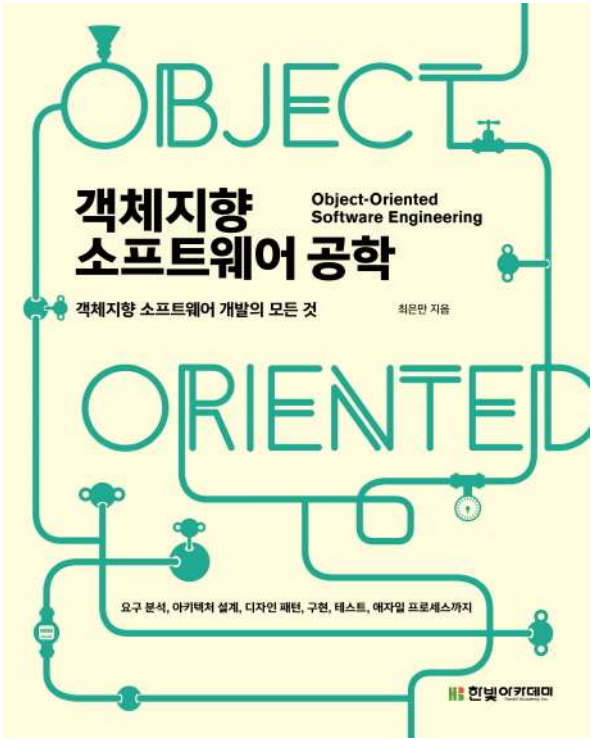
10장 : 그래프 프로젝트
1. 그래프 설명
2. Interpreter 패턴 적용하기
3. Decorator 패턴 적용하기
4. 변수 정의하기
5. Visitor 패턴 적용하기
6. 그래프 화면 만들기
7. 모델 만들기
8. MVC 패턴 적용하기
9. 무한대 처리하기
10. 그래픽점 데이터 정의하기
11. 유효하지 않은 값 처리하기
12. 전체 소스

(객체지향 일반)

참고

도서

객체지향 소프트웨어 공학 객체지향 소프트웨어 개발의 모든 것(2017) (@@@-%%)



Chapter 01 소프트웨어 공학의 소개
00 개요
01 소프트웨어란?
02 소프트웨어 공학이란?
03 소프트웨어 품질
04 소프트웨어 프로젝트의 유형
05 소프트웨어 프로젝트 작업
06 객체지향 소프트웨어 공학
07 최신 소프트웨어 공학의 핵심 기술
요약/연습문제

Chapter 02 객체지향의 개념
00 개요
01 객체지향이란?
02 클래스와 객체
03 인스턴스 변수, 클래스 변수, 메소드, 오퍼레이션
04 상속
05 다형성
06 UML
요약/연습문제

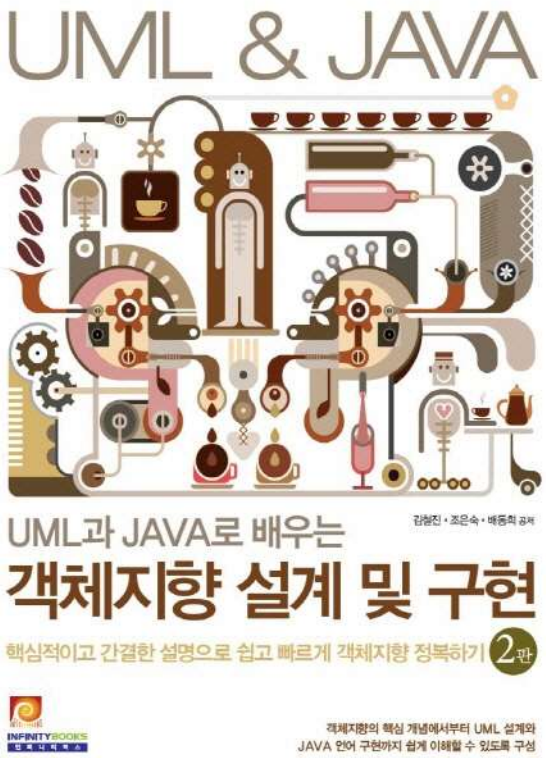
부록

정답 및 해설

찾아보기

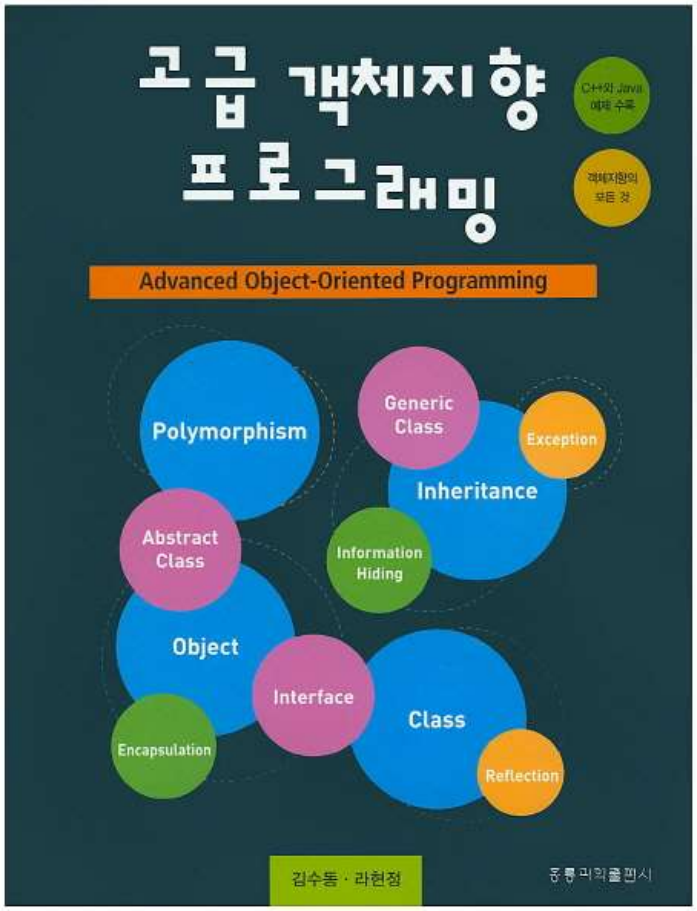
UML과 JAVA로 배우는 객체지향 설계 및 구현 핵심적이고 간결한 설명으로 쉽고 빠르게 객체지향 정복하기 2판

(2013) (X) (@@@)



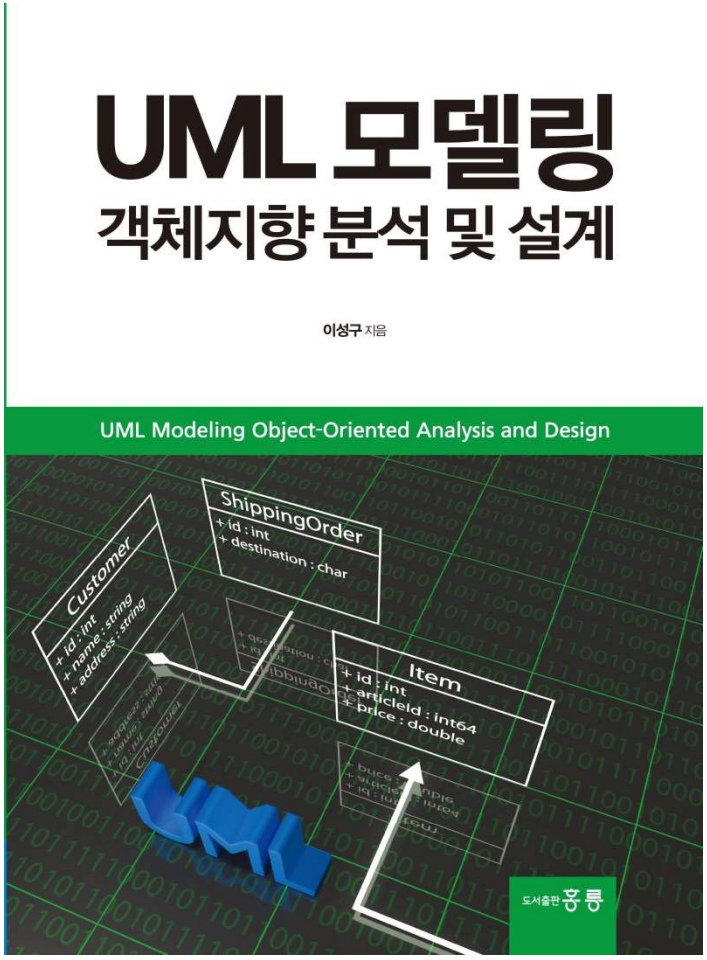
Chapter 02 클래스
Chapter 03 객체
Chapter 04 캡슐화
Chapter 05 정보은닉
Chapter 06 메시지
Chapter 07 복합객체
Chapter 08 상속
Chapter 09 추상 클래스
Chapter 10 인터페이스
Chapter 11 다형성
Chapter 12 컴포넌트
Chapter 13 설계 및 구현 환경 구축

고급 객체지향 프로그래밍 (2015) (X)-java과목과 유사

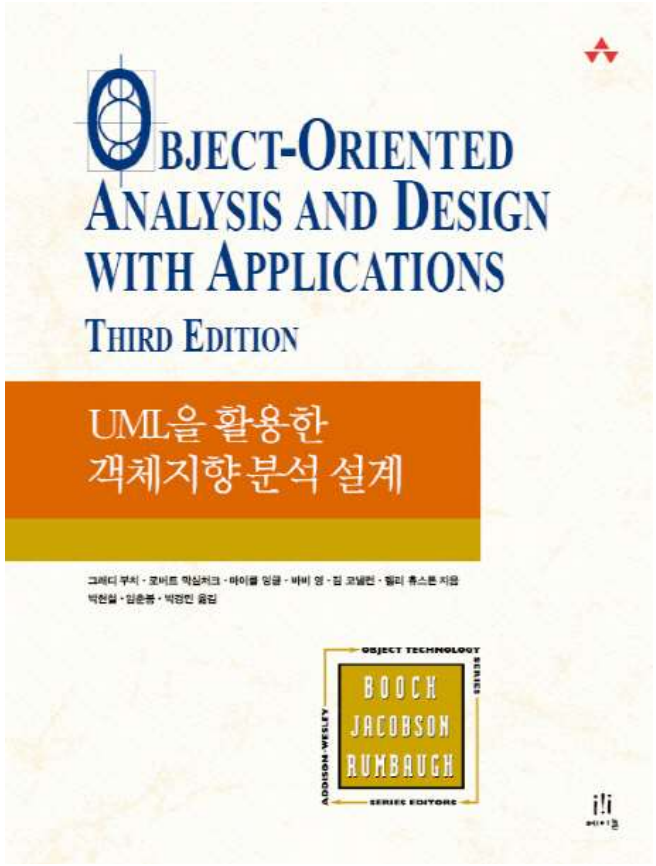


제1장 객체 지향 패러다임
제2장 객체(Object)
제3장 캡슐화(Encapsulation)
제4장 정보 은닉(Information Hiding)
제 5장 클래스(Class)
제 6장 상속(Inheritance)
제 7장 다형성(Polymorphism)
제 8장 추상 클래스(Abstract Class)
제 9장 인터페이스(Interface)
제 10장 제네릭 클래스(Generic Class)
제 11장 객체지향 예외 처리 (Exception Handling)
제 12장 리플랙션(Reflection)

UML 모델링 객체지향 분석 및 설계 (2020) (X)



UML을 활용한 객체지향 분석 설계(2013)



1부 개념

1장 복잡성
___1.1 복잡한 시스템 구조
___1.2 소프트웨어의 내재된 복잡성
___1.3 복잡한 시스템에 있는 다섯 가지 성분
___1.4 복잡성 제어
___1.5 혼돈에서 질서 찾기
___1.6 복잡한 시스템 설계하기

2장 객체모델
___2.1 객체모델 진화
___2.2 객체모델 기초
___2.3 객체모델 요소
___2.4 객체모델 적용하기

3장 클래스와 객체
___3.1 객체의 특성
___3.2 객체 간 관계
___3.3 클래스 성질
___3.4 클래스 간 관계
___3.5 클래스와 객체 간 작용
___3.6 품질 좋은 클래스와 객체 구축하기

4장 분류법

2부 방법

5장 표기
___5.1 UML
___5.2 패키지 다이어그램
___5.3 컴포넌트 다이어그램
___5.4 배치 다이어그램
___5.5 유스케이스 다이어그램
___5.6 활동 다이어그램
___5.7 클래스 다이어그램
___5.8 시퀀스 다이어그램
___5.9 상호작용개요 다이어그램
___5.10 복합구조 다이어그램
___5.11 상태 다이어그램
___5.12 타이밍 다이어그램
___5.13 객체 다이어그램
___5.14 커뮤니케이션 다이어그램

6장 프로세스
7장 화용론

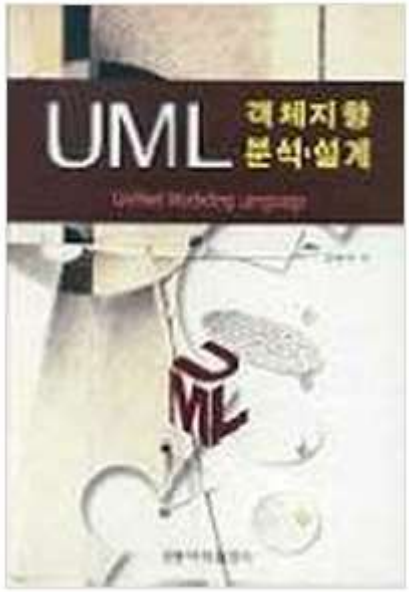
3부 애플리케이션

8장 시스템 아키텍처: 위성기반 항법장치
9장 제어 시스템: 교통관리
10장 인공지능: 암호해독
11장 데이터 획득: 기후 관찰 스테이션
12장 웹 애플리케이션: 휴가 관리 시스템

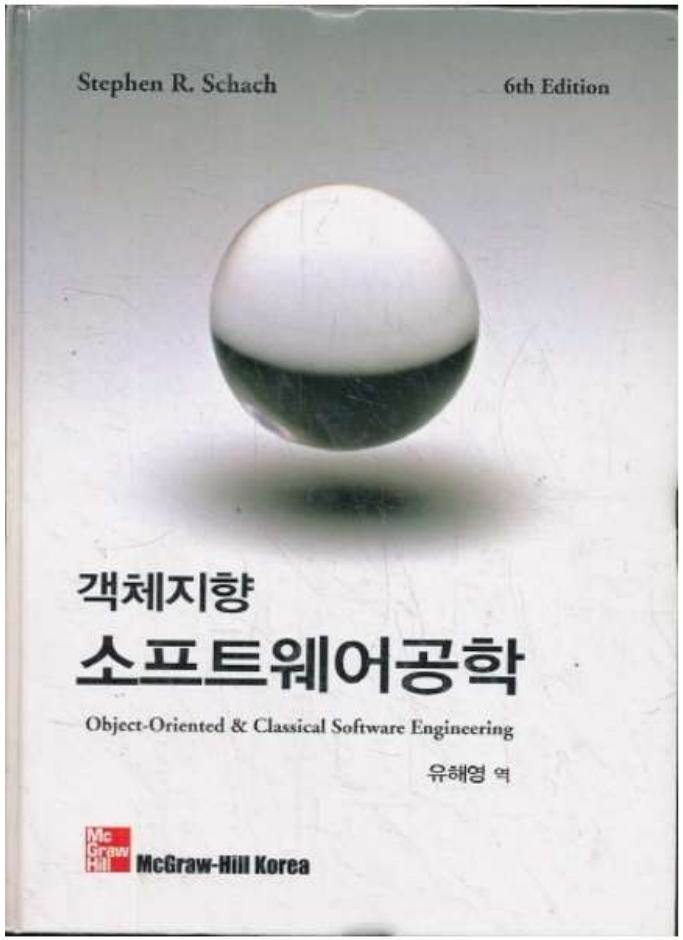
부록A 객체지향 프로그래밍 언어
___A.1 언어의 발전
___A.2 스몰토크
___A.3 C++
___A.4 자바

부록B 권장도서 안내

UML 객체지향 분석.설계 (2000)



객체지향 소프트웨어공학 (2005)(@@@) (X)



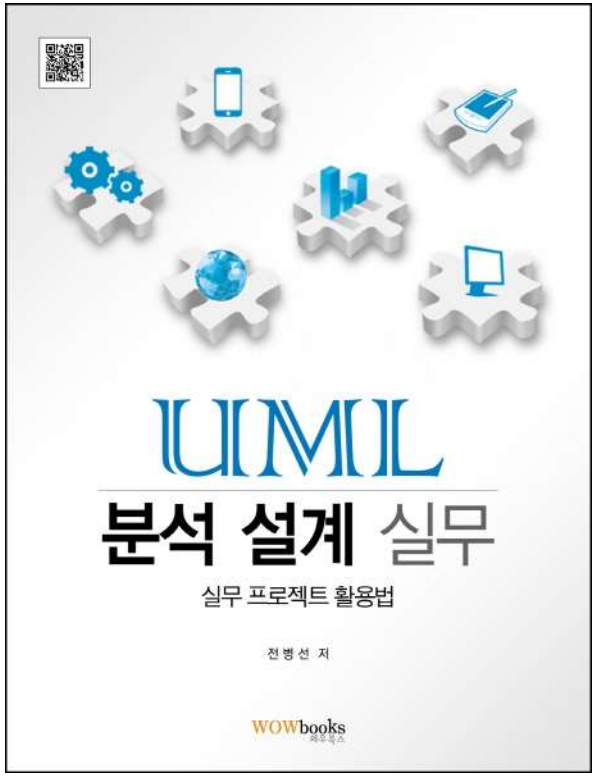
1부 소프트웨어 공학 개요

1. 소프트웨어 공학의 영역
2. 소프트웨어 생명주기 모델
3. 소프트웨어 프로세스
4. 팀
5. 툴의 선택
6. 테스팅
7. 모듈에서 객체까지
8. 재사용성과 이식성
9. 계획수립과 추정

2부 소프트웨어 생명주기의 단계

10. 요구사항
11. 고전적 분석
12. 객체-지향 분석
13. 설계단계
14. 구현
15. 인도 후 유지보수
16. UML의 세부사항

UML 분석 설계 실무: 실무 프로젝트 활용법 (2014)(X)(@@@)



제3장. 행위 다이어그램
3.1 활동 다이어그램
3.2 시퀀스 다이어그램
3.3 상태 다이어그램
3.4 유스케스 다이어그램

제4장. 유스케이스 모델링
4.1 유스케이스 개요
4.2 유스케이스 명세
4.3 유스케이스 구조화

제5장. 분석 클래스 모델링
5.1 분석 클래스 식별
5.2 유스케이스 실현-분석
5.3 분석 클래스 설계

제6장. 컴포넌트 모델링
6.1 컴포넌트 식별
6.2 유스케이스 실현-설계
6.3 컴포넌트 구조 설계
6.4 컴포넌트 행위 설계

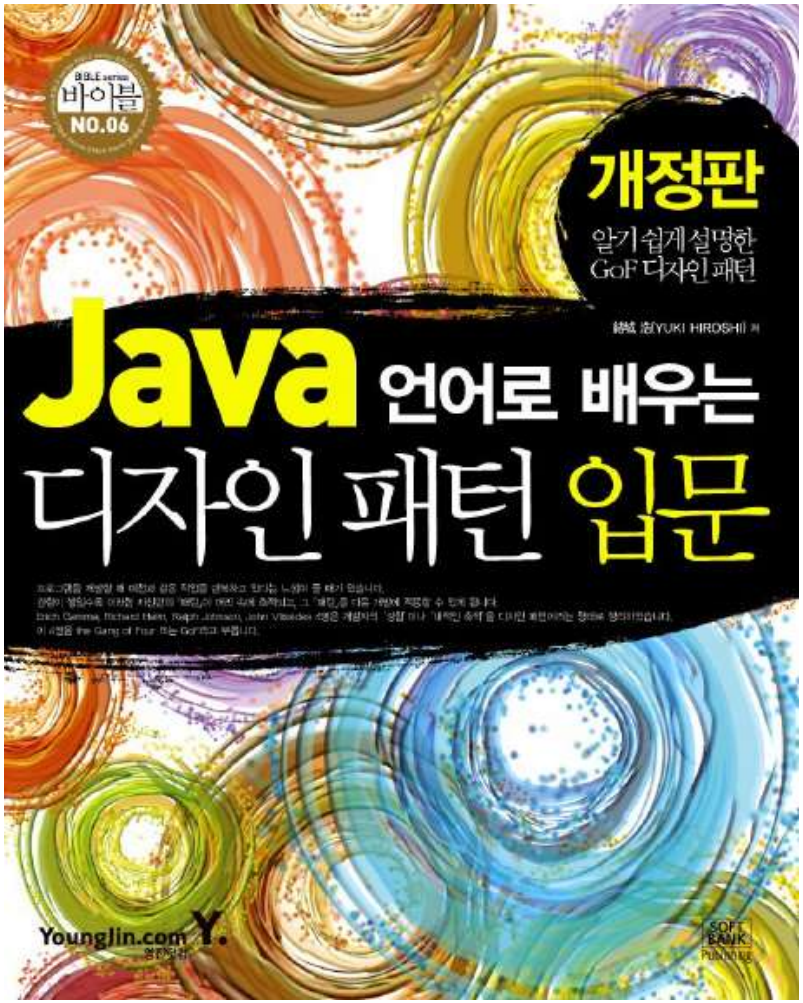
제7장 배포 모델링
1.1 배포 모델링

(디자인 패턴)

참고

도서

JAVA 언어로 배우는 디자인 패턴 입문(유키히로시, 영진닷컴)



목차

빨간색 – 수업시간에 다루지 않는 디자인 패턴

Part 2 하위 클래스에게 위임하기

Chapter 03 Template Method – 하위 클래스에서 구체적으로 처리하기

Chapter 04 Factory Method – 하위 클래스에서 인스턴스 만들기

Part 3 인스턴스 만들기

Chapter 05 Singleton – 인스턴스를 한 개만 만들기

Chapter 06 Prototype – 복사해서 인스턴스 만들기

Chapter 07 Builder – 복잡한 인스턴스 조립하기

Chapter 08 Abstarct Factory – 관련 부품을 조합해서 제품 만들기

Part 4 분리해서 생각하기

Chapter 09 Bridge – 기능 계층과 구현 계층 분리하기

Chapter 10 Strategy – 알고리즘을 모두 바꾸기

Part 5 동일시하기

Chapter 11 Composite – 그릇과 내용물을 동일시하기

Chapter 12 Decorator – 장식과 내용물을 동일시하기

Part 6 구조를 돌아다니기

Chapter 13 Visitor – 데이터 구조를 돌아다니면서 처리하기

Chapter 14 Chain of Responsibility – 책임 떠넘기기

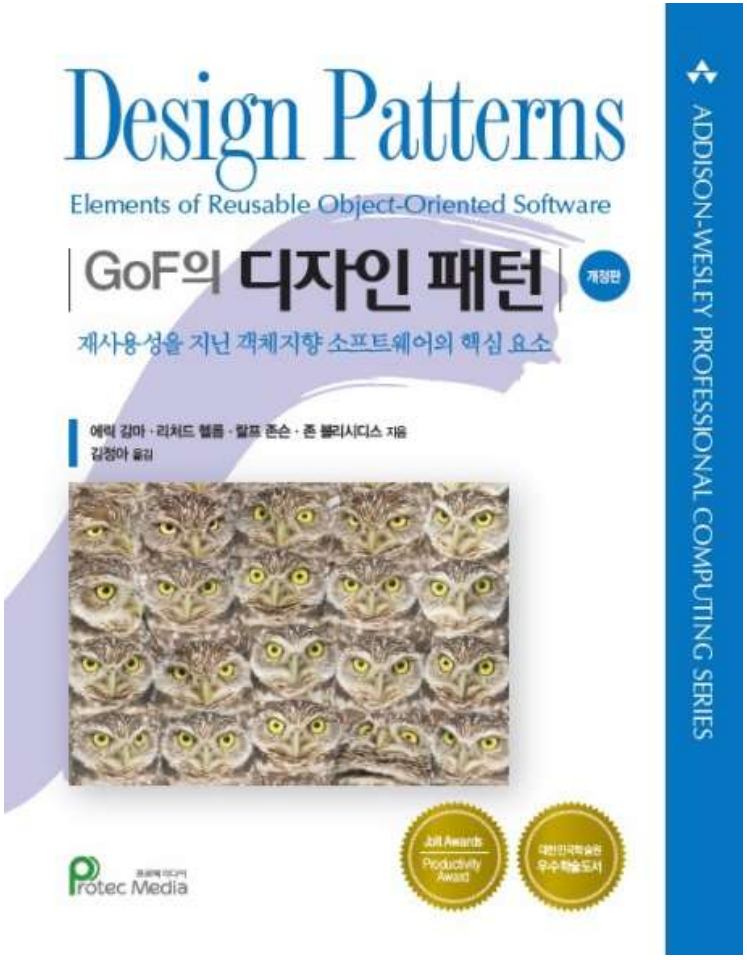
Part 7 단순화하기

Chapter 15 Facade – 단순한 창구

Chapter 16 Mediator – 중개인을 통해서 처리하기

빨간색 – 수업시간에 다루지 않는 디자인 패턴

GoF의 디자인 패턴 :재사용성을 지닌 객체 지향 소프트웨어의 핵심요소(2015) (#)



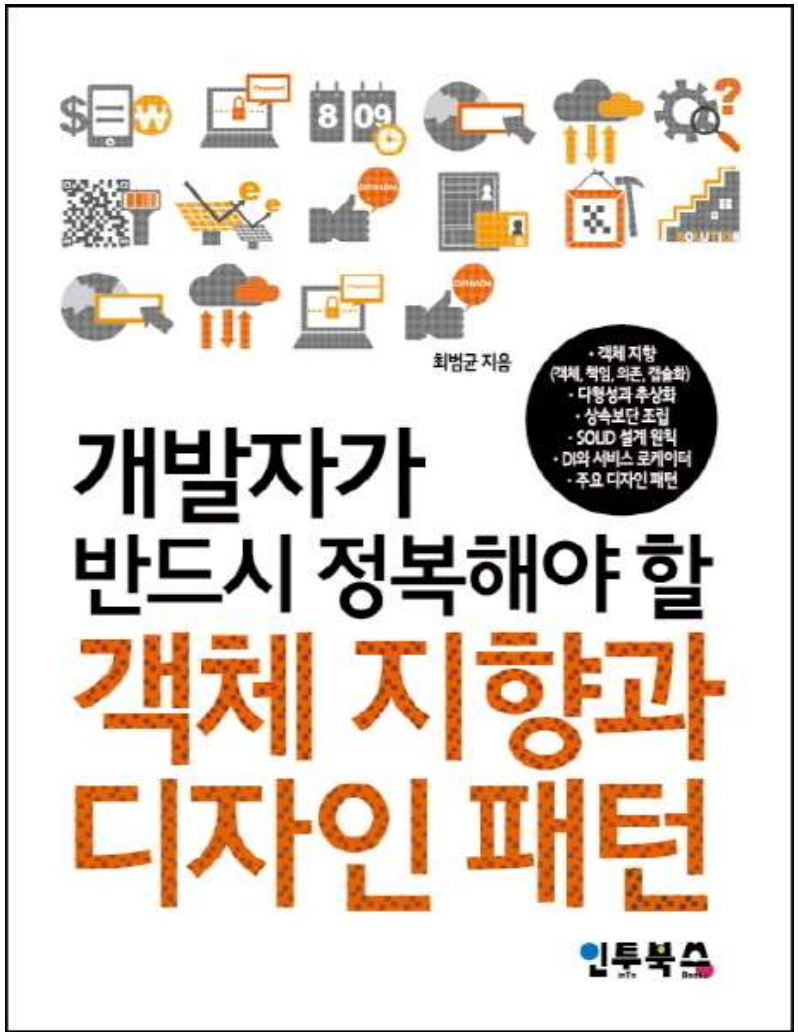
- Chapter1 서론
  Chapter2 사례 연구: 문서 편집기 설계

  디자인 패턴 카탈로그

  Chapter3 생성 패턴
  Chapter4 구조 패턴
  Chapter5 행동 패턴
  Chapter6 결론

  Appendix A 용어정리
  Appendix B 표기법 가이드
  Appendix C 예제 코드에 사용된 기본 클래스

개발자가 반드시 정복해야 할 객체 지향과 디자인 패턴(최범균, 한산) (###) (2013) (@@@--도서관)



Chapter 06 DI(Dependency Injection)와 서비스 로케이터

1. 어플리케이션 영역과 메인 영역

2. DI(Dependency Injection)을 이용한 의존 객체 사용

2.1 생성자 방식과 설정 메서드 방식

2.2 DI와 테스트

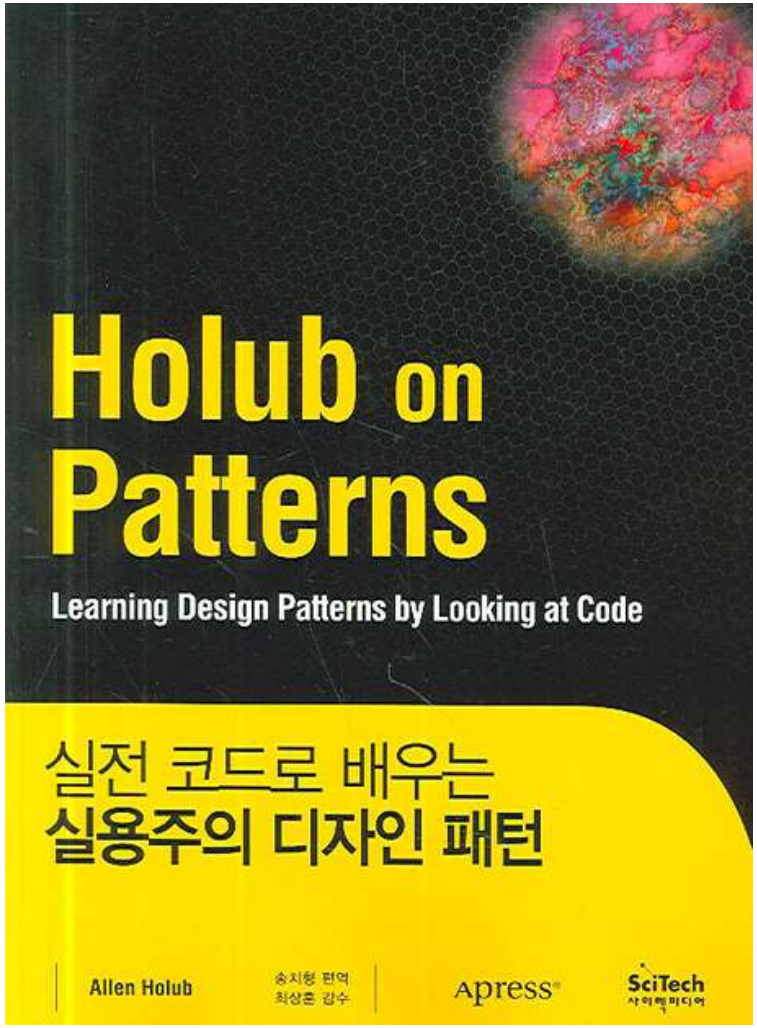
2.3 스프링 프레임워크 예

3. 서비스 로케이터를 이용한 의존 객체 사용

3.1 서비스 로케이터의 구현

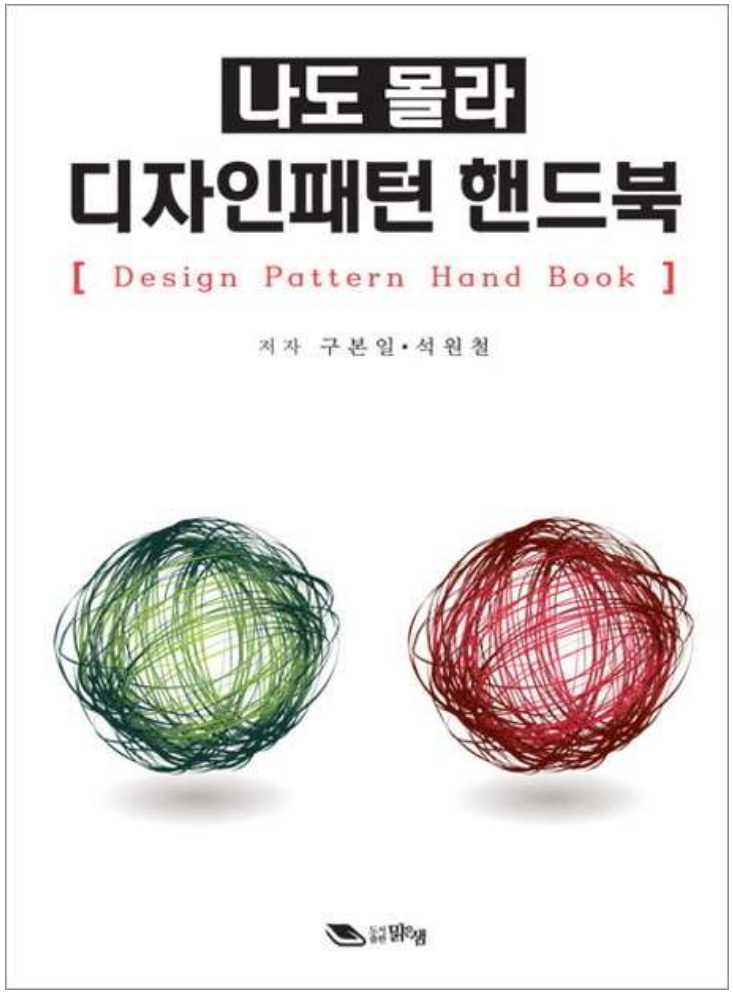
3.2 서비스 로케이터의 단점

실전 코드로 배우는 실용주의 디자인 패턴(X)

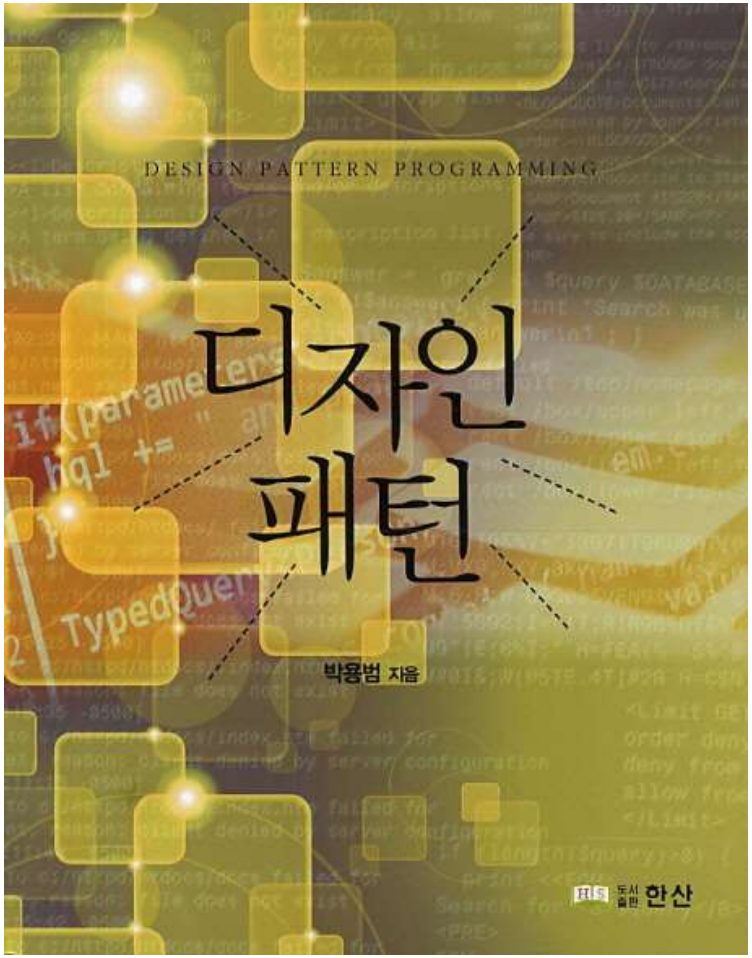


목차

나도 몰라 디자인 패턴 핸드북

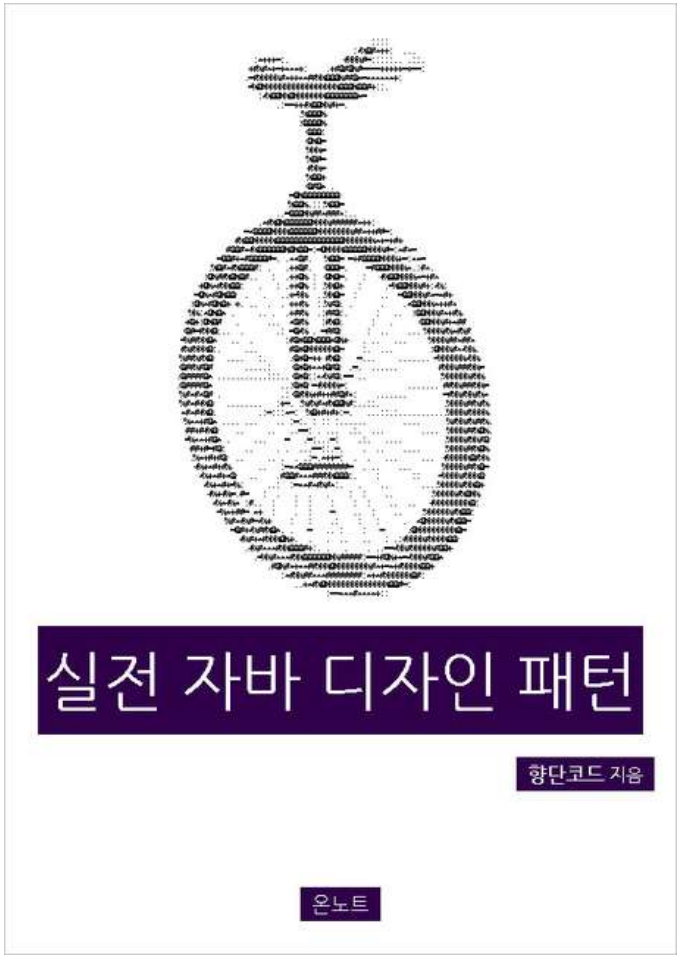


디자인 패턴(박용범, 한산)



목차

실전자바디자인패턴(###)

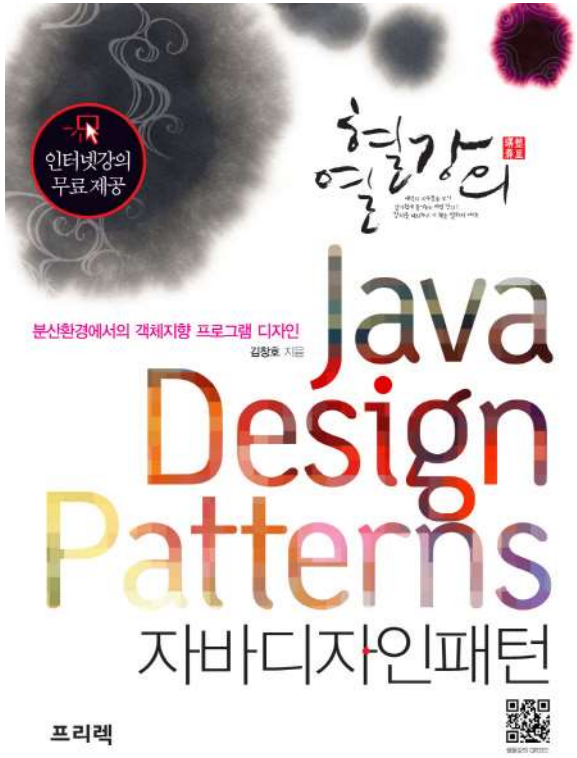


- (앞의 책과 8장까진 같음)

9장 : 계산기 프로젝트
1. 계산기 설명
2. 책임 연쇄 패턴 적용하기
3. 수학함수 피연산자 정의하기
4. 계산기 화면 만들기
5. 모델 만들기
6. Command 패턴 적용하기
7. 사칙연산 처리하기
8. State 패턴 적용하기
9. 계산 초기화
10. 수학함수 처리하기
11. +-부호 지정하기
12. 전체 소스

10장 : 페인터프로젝트
1. 페인터 설명
2. 페인터 화면 만들기
3. 모델 만들기
4. Factory Method 패턴 적용하기
5. MVC 패턴 적용하기
6. Observer 패턴 적용하기
7. Strategy 패턴 적용하기
8. State 패턴 적용하기
9. Memento 패턴 적용하기
10. 전체 소스

열혈강의 자바 디자인 패턴 분산환경에서의 객체지향 프로그램 디자인 (#)



Chapter 01 MVC: Model-View-Controller - 역할과 책임 나누기

1.1 MVC의 일반적 예

1.2 MVC란?

1.3 웹에서의 MVC

1.4 Explorer 비교

Chapter 2 Factory Method 패턴 - 객체 생성은 의뢰하자

2.1 객체 생성

2.2 Factory Method 패턴이란?

Chapter 3 객체지향 - 패턴 이해를 위한 객체지향 개념

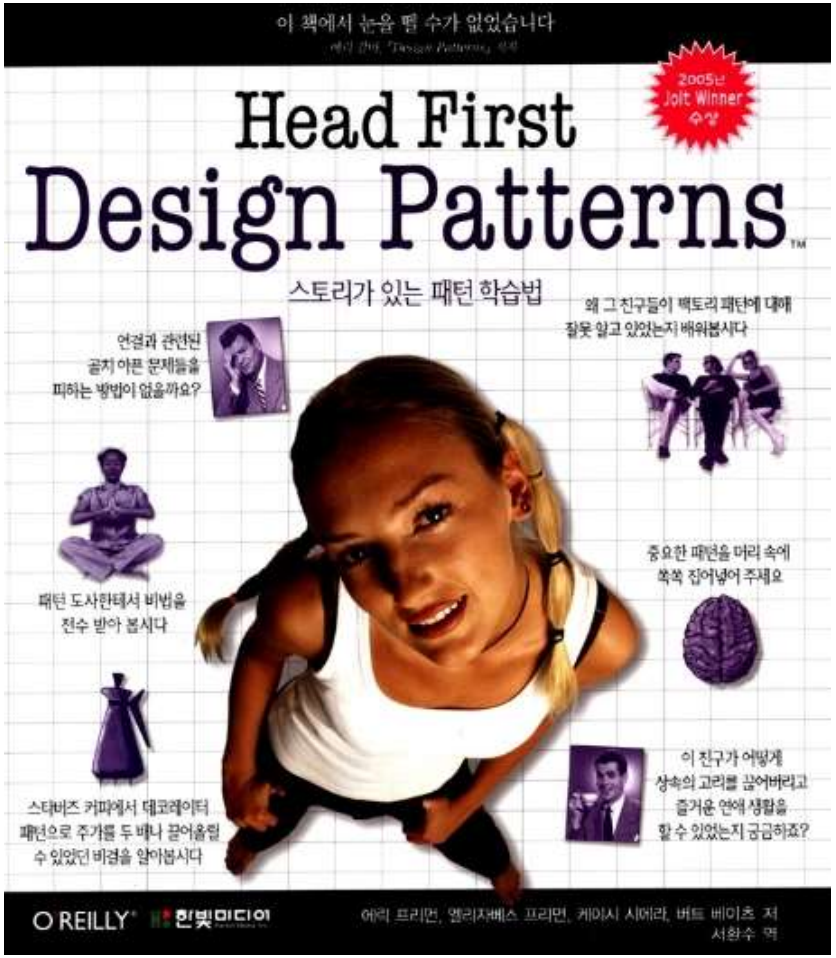
3.1 추상화란?

3.2 추상화의 보편적 예들

3.3 추상화의 관점에서 본 패턴

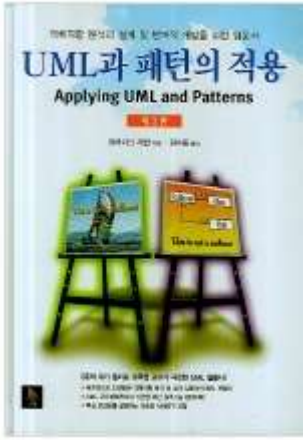
(#)

Head First Design Patterns (2005)



1. 디자인 패턴 소개
SimUDuck
조는 상속에 대해서 생각을 해 봅니다...
인터페이스는 어떨까요?
소프트웨어 개발에 있어서 바뀌지 않는 것
바뀌는 부분과 그렇지 않은 부분 분리하기
오리의 행동 디자인
Duck 코드 테스트
동적으로 행동을 지정하는 방법
캡슐화된 행동을 큰 그림으로 바라봅시다
“A는 B이다”보다“A에는 B가 있다”가 나을 수 있습니다
스트래티지 패턴
전문 용어의 위력
디자인 패턴을 어떻게 사용하나요?
디자인 도구상자
연습문제 정답

UML과 디자인 패턴의적용 (2005)



Part 1 소개 (INTRODUCTION)

Part 2 도입 (INCEPTION)

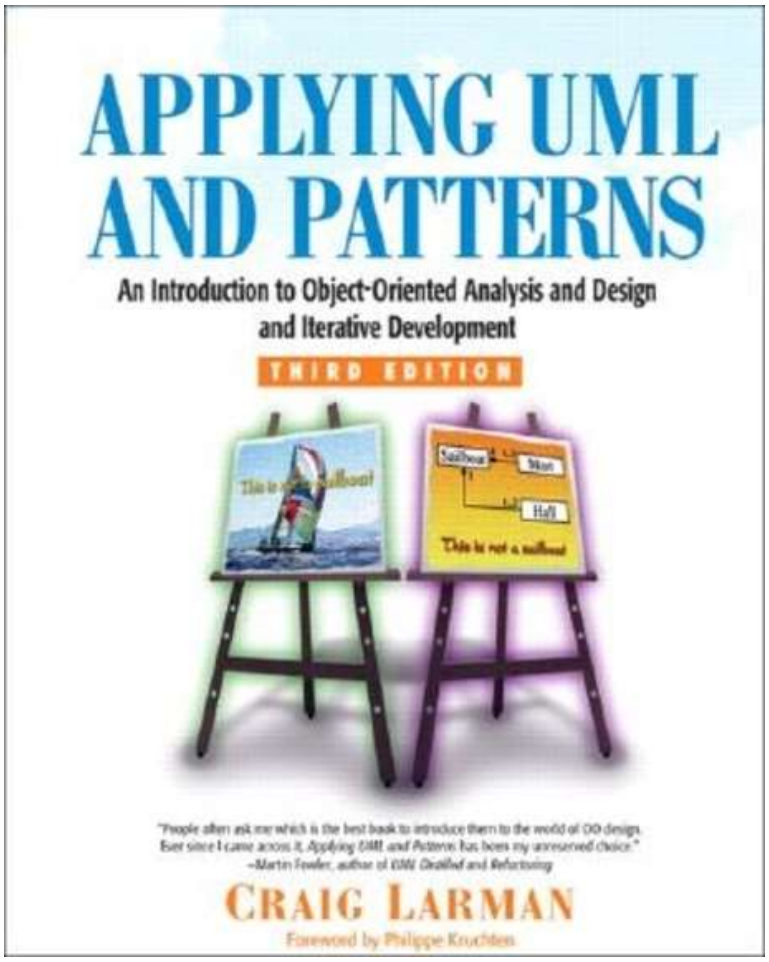
Part 3 반복단계 1

Part 4 반복단계 2

Part 5 반복단계 3

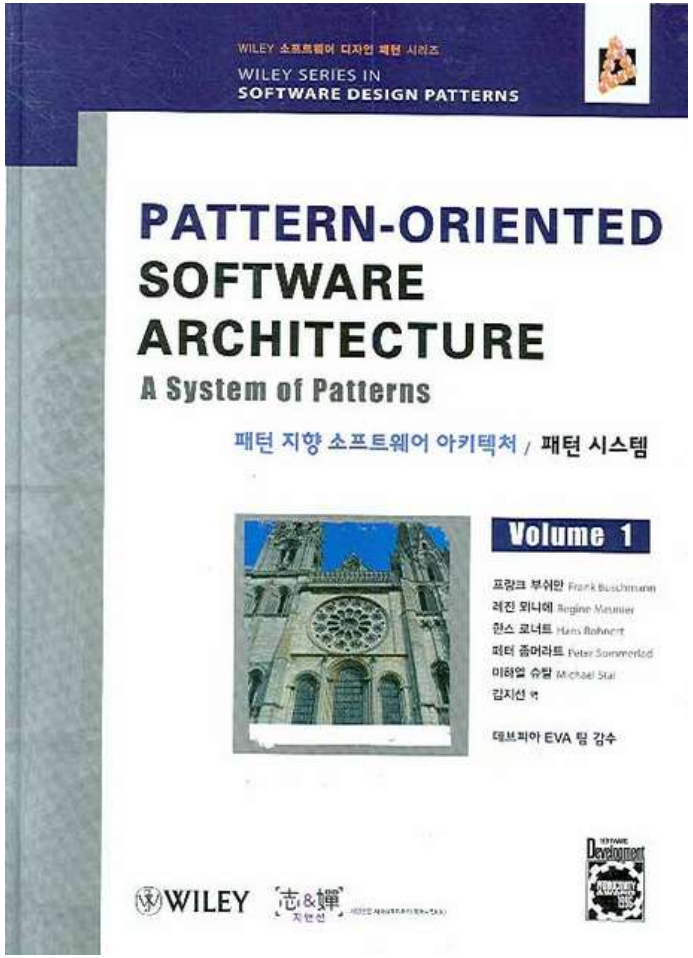
Part 6 특별한 논제들

Applying UML and Patterns 3/E



- Larman, Craig 지음 | Prentice Hall PTR | 2005년 01월 01일

패턴 지향 소프트웨어 아키텍처 패턴시스템



- VOLUME. 1 WILEY 소프트웨어 디자인 패턴 시리즈 | 양장
- 프랑크 부쉬만 , 레진 뫼니에 , 한스 로너트 , 페터 좀머라트 , 미하엘 슈탈 지음 | 김지선 옮김 | 데브피아 EVA 팀 감수 | 지앤선 | 2008년 01월 18일 출간

(리팩토링)

참고

도서

자바로 배우는 리팩토링 입문 건강한 코드로 소프트웨어 체질을 개선하자 (2017) (@@@-%%)

0장 리팩토링이란

0.1 리팩토링이란

0.2 리팩토링과 악취

0.3 리팩토링 카탈로그

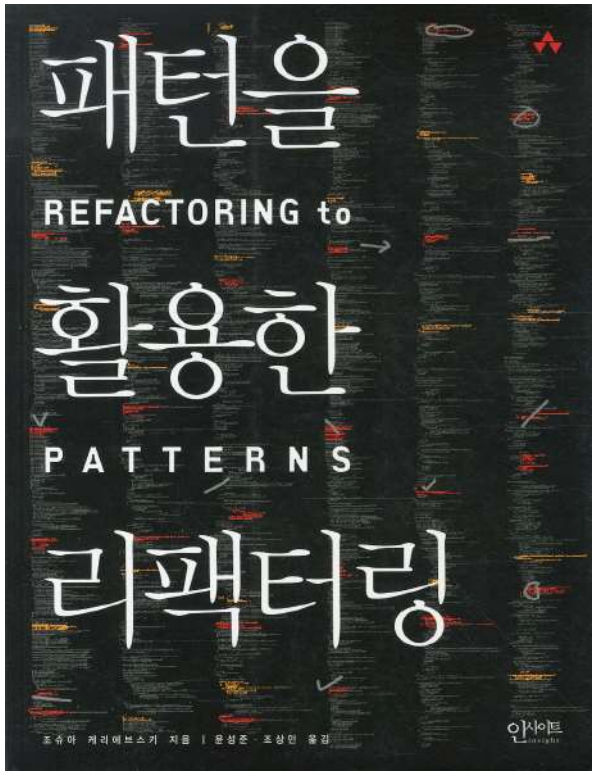
0.4 리팩토링 에센스

0.5 리팩토링 Q&A

0.6 연습 문제

0.7 연습 문제 해답

패턴을 활용한 리팩터링 (2011-도서관)



8장 일반화

Form Template Method

Extract Composite

Replace One/Many Distinction with Composite

Replace Hard-Coded Notifications with Observer

Unify Interfaces with Adapter

Extract Adapter

Replace Implicit Language with Interpreter

9장 보호

Replace Type Code with Class

Limit Instantiation with Singleton

Introduce Null Object

10장 축적

Move Accumulation to Collecting Parameter

Move Accumulation to Visitor

11장. 유틸리티

Chain Constructor

Unify Interfaces

Extract Parameter

표 4.1 12개의 냄새와 관련 리팩토링

| 표 4.1 |

냄새[1]	리팩터링
중복된 코드 (Duplicated Code, 78쪽) [F]	Form Template Method (281) Introduce Polymorphic Creation with Factory Method (134) Chain Constructors (448) Replace One/Many Distinctions with Composite (303) Extract Composite (291) Unify Interfaces with Adapter (333) Introduce Null Object (402)

냄새	리팩터링
긴 메서드 (Long Method, 79쪽) [F]	Compose Method (179) Move Accumulation to Collecting Parameter (415) Replace Conditional Dispatcher with Command (265) Move Accumulation to Visitor (423) Replace Conditional Logic with Strategy (187)
복잡한 조건문 (Conditional Complexity, 80쪽)	Replace Conditional Logic with Strategy (187) Move Embellishment to Decorator (206) Replace State-Altering Conditionals with State (234) Introduce Null Object (402)
기본 타입에 대한 강박관념 (Primitive Obsession, 81쪽) [F]	Replace Type Code with Class (383) Replace State-Altering Conditionals with State (234) Replace Conditional Logic with Strategy (187) Replace Implicit Tree with Composite (249) Replace Implicit Language with Interpreter (360) Move Embellishment to Decorator (206) Encapsulate Composite with Builder (145)

추잡한 노출 (Indecent Exposure, 82쪽)	Encapsulate Classes with Factory (124)
문어발 솔루션 (Solution Sprawl, 83쪽)	Move Creation Knowledge to Factory (110)
인터페이스가 서로 다른 대체 클래스 (Alternative Classes with Different Interfaces, 83쪽) [F]	Unify Interfaces with Adapter (333)
게으른 클래스 (Lazy Class, 84쪽) [F]	Inline Singleton (168)
거대한 클래스 (Large Class, 84쪽) [F]	Replace Conditional Dispatcher with Command (265) Replace State-Altering Conditionals with State (234) Replace Implicit Language with Interpreter (360)

냄새	리팩터링
Switch 문 (Switch Statements, 85쪽) [F]	Replace Conditional Dispatcher with Command (265) Move Accumulation to Visitor (423)
조합의 폭발적 증가 (Combinatorial Explosion, 85쪽)	Replace Implicit Language with Interpreter (360)
괴짜 솔루션 (Oddball Solution, 86쪽)	Unify Interfaces with Adapter (333)

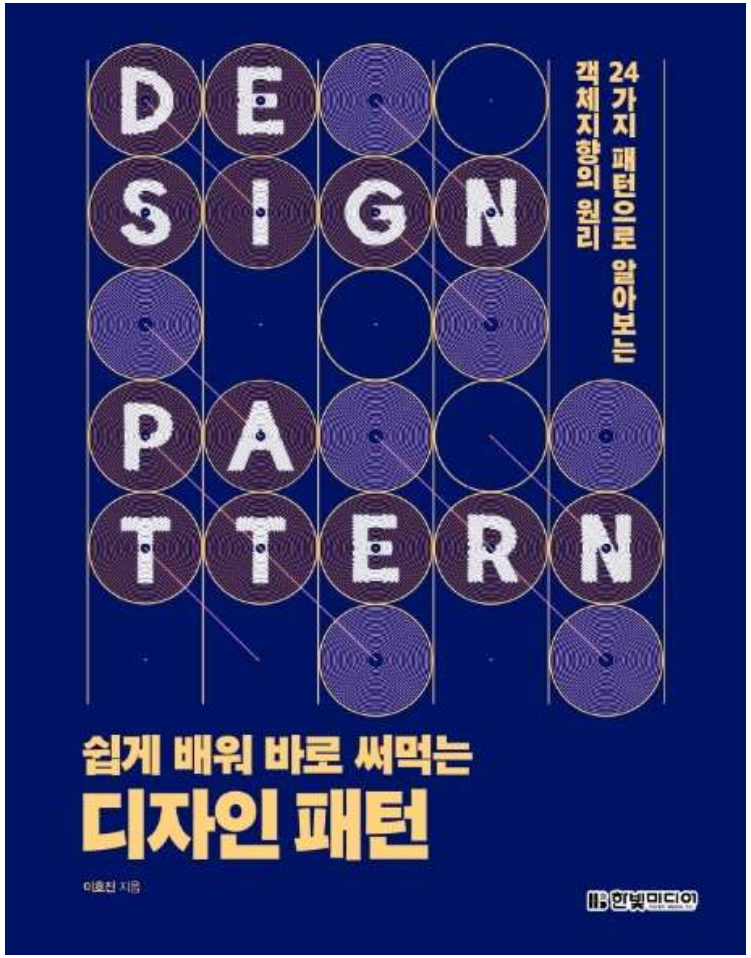
표 5.1

세션	리팩터링
1	Replace Constructors with Creation Methods (97) Chain Constructors (448)
2	Encapsulate Classes with Factory (124)
3	Introduce Polymorphic Creation with Factory Method (134)
4	Replace Conditional Logic with Strategy (187)
5	Form Template Method (281)
6	Compose Method (179)

7	Replace Implicit Tree with Composite (249)
8	Encapsulate Composite with Builder (145)
9	Move Accumulation to Collecting Parameter (415)
10	Extract Composite (291) Replace One/Many Distinctions with Composite (303)
11	Replace Conditional Dispatcher with Command (265)
12	Extract Adapter (347) Unify Interfaces with Adapter (333)
13	Replace Type Code with Class (383)
14	Replace State-Altering Conditionals with State (234)
15	Introduce Null Object (402)
16	Inline Singleton (168) Limit Instantiation with Singleton (396)

세션	리팩터링
17	Replace Hard-Coded Notifications with Observer (319)
18	Move Embellishment to Decorator (206) Unify Interfaces (453) Extract Parameter (456)
19	Move Creation Knowledge to Factory (110)
20	Move Accumulation to Visitor (423)
21	Replace Implicit Language with Interpreter (360)

쉽게 배워 바로 써먹는 디자인 패턴 24가지 패턴으로 알아보는 객체지향의 원리(이호진, 한빛미디어, PHP-도서관)

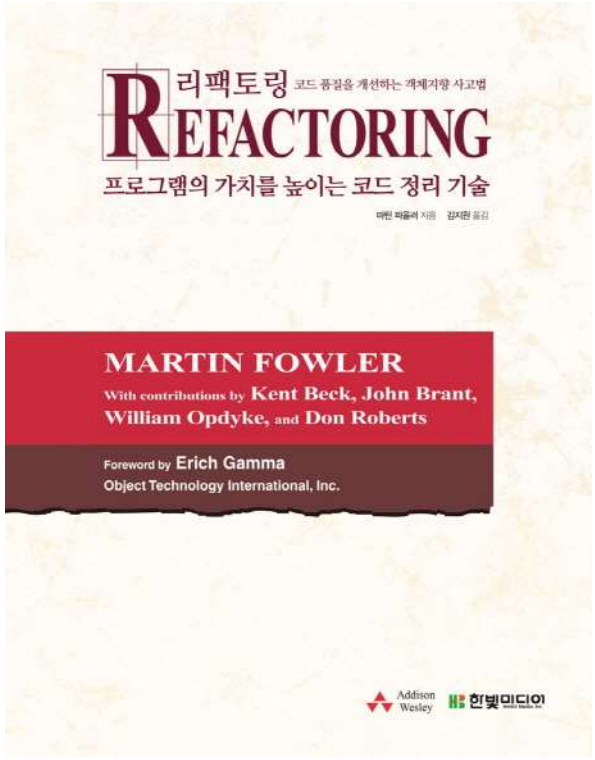


CHAPTER 0 디자인 패턴
0.1 패턴
0.2 소프트웨어 공학
0.3 설계 원칙
0.4 GoF
0.5 패턴의 요소
0.6 유지 보수
0.7 정리

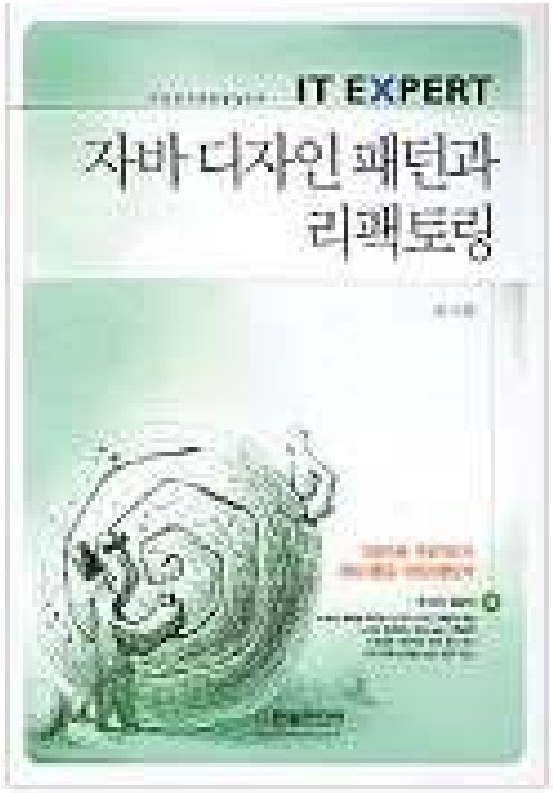
[PART 1 생성 패턴 - 추상화를 통해 객체 생성하기]

CHAPTER 1 팩토리 패턴
1.1 클래스와 객체지향
1.2 의존성
1.3 의존성 주입
1.4 의존 관계의 문제점
1.5 팩토리 패턴
1.6 단순 팩토리
1.7 장점과 단점
1.8 관련 패턴
1.9 정리

리팩토링: 코드 품질을 개선하는 객체 지향 사고법(2012) (@@@-도서관)



자바 디자인 패턴과 리팩토링 (2003) (@@@)



1부 객체지향이란 무엇인가?

- 1장 객체지향의 밑바닥
- 2장 UML 입문

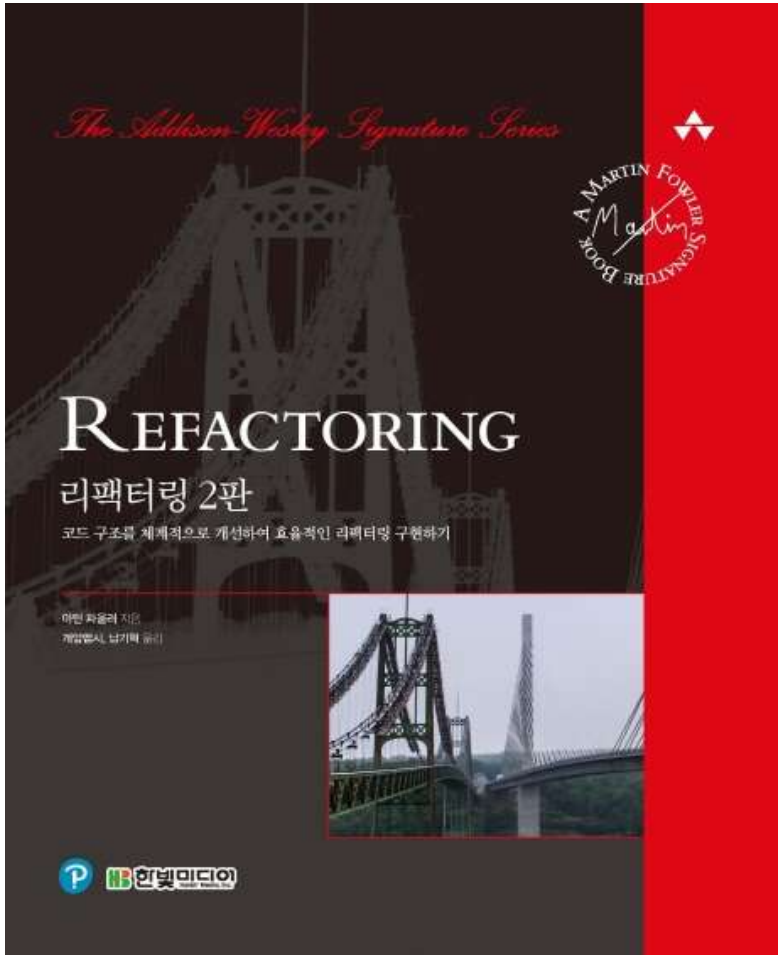
2부 자바의 객체지향 특성 100% 활용

- 3장 클래스와 상속, 그리고...
- 4장 확장성 있고 유연한 자바 프로그램 만들기

3부 객체지향 소프트웨어 설계의 기본 원리

- 5장 소프트웨어의 변화와 포용
- 6장 변화를 수용하는 객체지향 설계의 원리

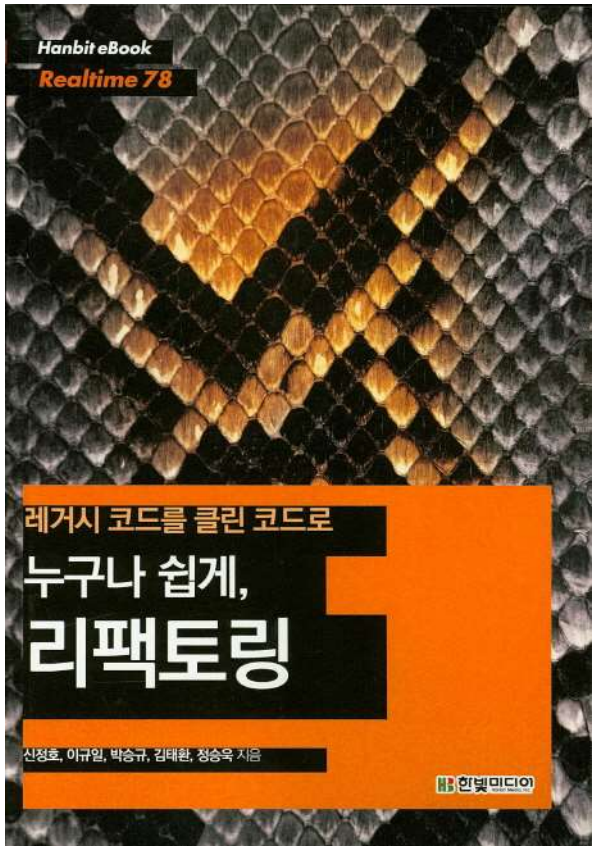
리팩터링 - 코드 구조를 체계적으로 개선하여 효율적인 리팩터링 구현하기 2판 (자바스크립트, 2020-도서관)



CHAPTER 01 리팩터링: 첫 번째 예시
1.1 자, 시작해보자!
1.2 예시 프로그램을 본 소감
1.3 리팩터링의 첫 단계
1.4 statement() 함수 쪼개기
1.5 중간 점검: 난무하는 중첩 함수
1.6 계산 단계와 포맷팅 단계 분리하기
1.7 중간 점검: 두 파일(과 두 단계)로 분리됨
1.8 다형성을 활용해 계산 코드 재구성하기
1.9 상태 점검: 다형성을 활용하여 데이터 생성하기
1.10 마치며

CHAPTER 02 리팩터링 원칙
2.1 리팩터링 정의
2.2 두 개의 모자
2.3 리팩터링하는 이유
2.4 언제 리팩터링해야 할까?
2.5 리팩터링 시 고려할 문제
2.6 리팩터링, 아키텍처, 애그니(YAGNI)
2.7 리팩터링과 소프트웨어 개발 프로세스
2.8 리팩터링과 성능
2.9 리팩터링의 유래
2.10 리팩터링 자동화
2.11 더 알고 싶다면

레거시 코드를 클린 코드로 누구나 쉽게, 리팩토링 (2014)



- final static 필드를 모아 놓아서 뚱뚱해진 클래스 개선하기
- 혼동되는 생성자 초기화 개선하기
- 독립된 중복 메서드를 효율적으로 개선하기
- 매개변수 남용으로 거대해진 매서드 개선하기
- 비즈니스 로직과 기능 호출이 섞여 있는 메서드 개선하기
- 분기문에 복잡하게 꼬여있는 AND 와 OR 연산자 개선하기
- 조건에 따라 분리되는 객체 생성 로직 개선하기
- 응집도가 낮은 멤버 클래스 개선하기
- 잘못된 이해로 생긴 상속 구조 개선하기
- 원래 기능과 다른 Null 예외 처리 개선하기
- 연동 규약에 종속된 구조 개선하기
- 유사한 기능의 인터페이스 다중 상속 구조 개선하기
- 놓치기 쉬운 싱글톤 오류 개선하기

소프트웨어 악취를 제거하는 리팩토링 구조적 설계 문제를 풀어내는 최선의 실천법 (2015-도서관)