

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	<input checked="" type="checkbox"/> 강의 <input checked="" type="checkbox"/> Project <input type="checkbox"/> 토론 <input type="checkbox"/> 실험 <input type="checkbox"/> 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
출판사	한빛미디어	출판년도	2017(3쇄)
부교재		Line +	
서명	Object-Oriented Software Engineering Using UML, Patterns, and Java™	저자	Bernd Bruegge & Allen H
출판사	Prentice Hall	출판년도	Third Edition
서명	Java Design Patterns	저자	
출판사	Exelixis Media P.C	출판년도	2015
서명	Object-Oriented Software Development Using Java,	저자	Xiaoping Jia
출판사	Addison Wesley	출판년도	Second Edition
기자재			



● 평가

출석관리	<ul style="list-style-type: none"> - 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 팀프로젝트를 수행함 - 팀프로젝트는 iterative development방식을 통하여 단계별로 개발결과를 제시함 - 단계별로 리팩토링 또는 디자인패턴의 적용을 시도함 - 디자인 패턴을 정확히, 그리고 다수 적용할때 좋은 평가를 받게 됨



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) ICT 융합기초 과목군 - 파이썬 프로그래밍, 파이썬으로 배우는 기계학습, R을 이용한 빅데이터 분석
(주2) 과학 과목군 - 물리학1, 물리학 실험1, 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 시점과 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 3분후부터 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, 예비군훈련은 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 2번**의 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



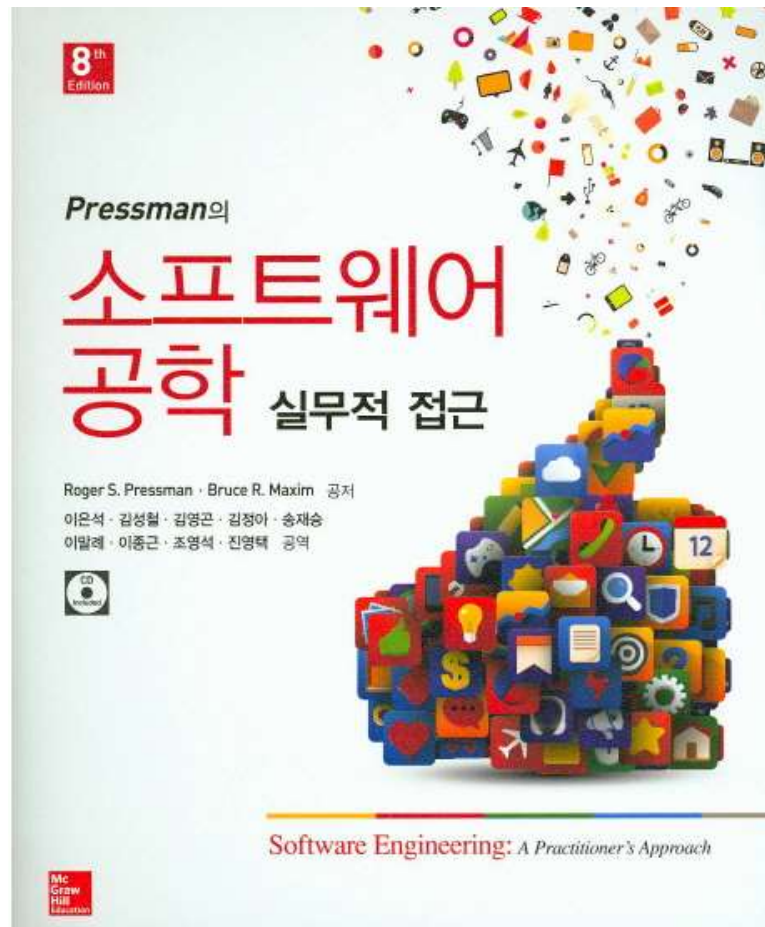
강의

방향



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

“소프트웨어공학”과의 비교



- CHAPTER 01 소프트웨어의 본질
CHAPTER 02 소프트웨어공학

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 테스트 전략
CHAPTER 23 전통적인 애플리케이션 테스트
CHAPTER 24 객체지향 응용프로그램 테스트

- CHAPTER 25 웹앱 테스트
CHAPTER 26 모바일 앱 테스트
CHAPTER 27 보안성 공학
CHAPTER 28 정형 모델링과 검증
CHAPTER 29 소프트웨어 형상관리
CHAPTER 30 제품 척도

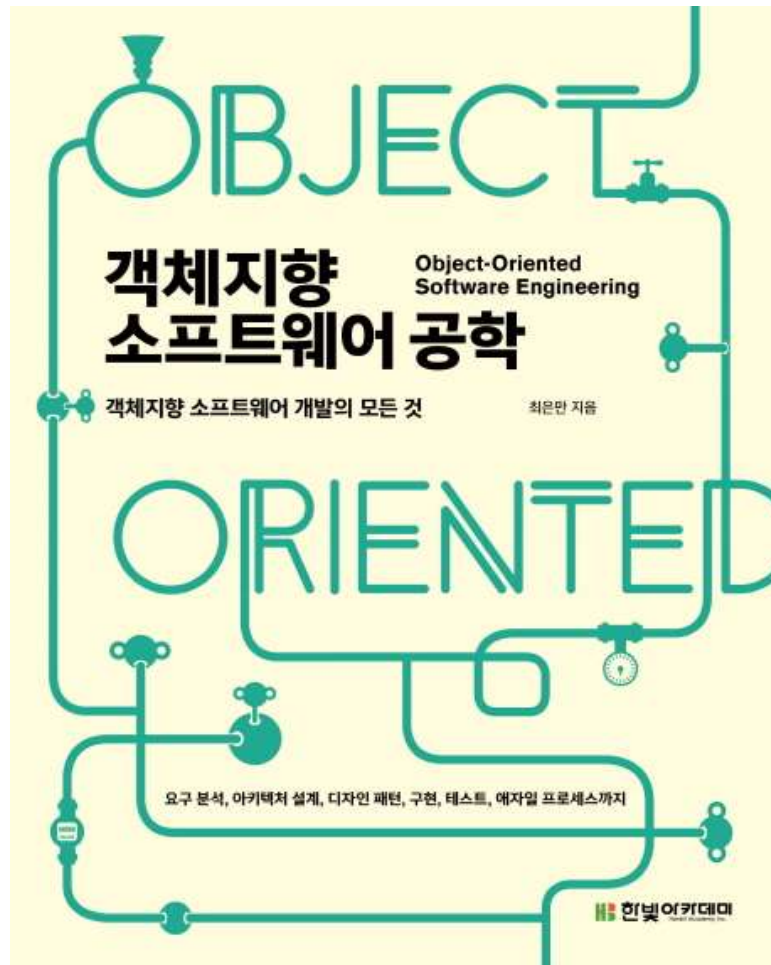
PART 4 소프트웨어 프로젝트 관리

CHAPTER 31 프로젝트 관리의 개념
CHAPTER 32 프로세스와 프로젝트 척도
CHAPTER 33 소프트웨어 프로젝트의 산정
CHAPTER 34 프로젝트 스케줄링
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, 오버레이션
- 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 시점에서 팀 구성
 - 임의 할당 방식
 - 4명 또는 3명
- 팀 활동 예
 - 디자인패턴 1~2개씩 발표 (java design patterns)
 - 기여도 상호 평가 (기말고사)

리팩터링(refactoring)

- 리팩터링(refactoring)은 소프트웨어 공학에서 '결과 **의 변경 없이** 코드의 구조를 재조정함'을 뜻한다. 주로 **가독성**을 높이고 **유지보수**를 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. 리팩터링은 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)



CHAPTER 1 객체지향 모델링

__1.1 모델링

__1.2 UML

__1.3 클래스 다이어그램

___1.3.1 클래스

___1.3.2 관계

__체크포인트 해설

__연습문제

CHAPTER 2 객체지향 원리

__2.1 추상화

__2.2 캡슐화

__2.3 일반화 관계

___2.3.1 일반화는 또 다른 캡슐화

___2.3.2 일반화 관계와 위임

___2.3.3 집합론 관점으로 본 일반화 관계

__2.4 다형성

__2.5 피터 코드의 상속 규칙

__체크포인트 해설

__연습문제

CHAPTER 3 SOLID 원칙

__3.1 단일 책임 원칙

___3.1.1 책임의 의미

___3.1.2 변경

___3.1.3 책임 분리

___3.1.4 산탄총 수술

___3.1.5 관심지향 프로그래밍과 횡단 관심 문제

__3.2 개방-폐쇄 원칙

__3.3 리스코프 치환 원칙

__3.4 의존 역전 원칙

__3.5 인터페이스 분리 원칙

__체크포인트 해설

__연습문제

CHAPTER 4 디자인 패턴

__4.1 디자인 패턴의 이해

__4.2 GoF 디자인 패턴

__4.3 UML과 디자인 패턴

___4.3.1 컬레보레이션

___4.3.2 순차 다이어그램

___4.3.3 순차 다이어그램과 클래스 다이어그램의 관계

__체크포인트 해설

__연습문제

CHAPTER 5 스트래티지 패턴

— 5.1 로봇 만들기

— 5.2 문제점

— 5.2.1 기존 로봇의 공격과 이동 방법을 수정하는 경우

— 5.2.2 새로운 로봇에 공격/이동 방법을 추가/수정하는 경우

— 5.3 해결책

— 5.4 스트래티지 패턴

— 연습문제

CHAPTER 6 싱글턴 패턴

CHAPTER 7 스테이트 패턴

CHAPTER 8 커맨드 패턴

CHAPTER 9 옵서버 패턴(-J)

CHAPTER 10 데커레이터 패턴

CHAPTER 11 템플릿 메서드 패턴

CHAPTER 12 팩토리 메서드 패턴

CHAPTER 13 추상 팩토리 패턴

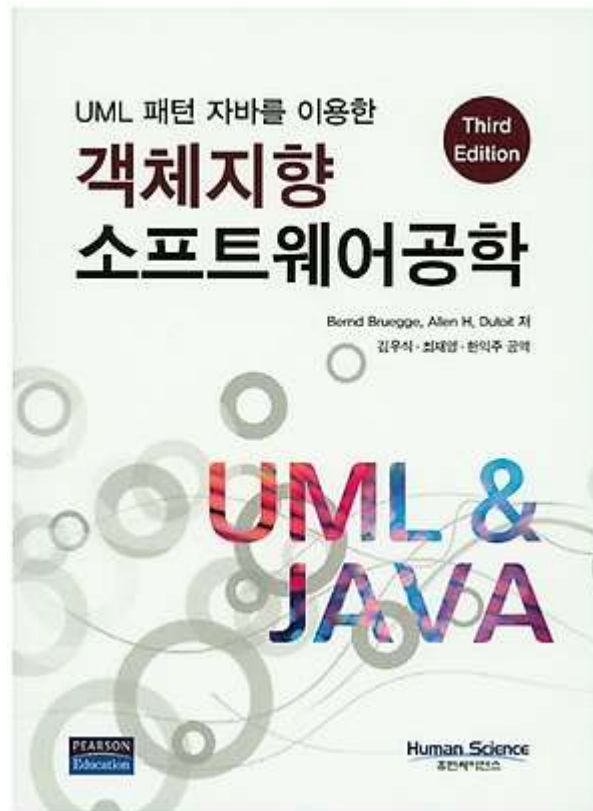
CHAPTER 14 컴퍼지트 패턴

가능한한 개인별 구매 추천

주교재



UML 패턴 자바를 이용한 객체지향 소프트웨어공학(2014)



Part 1 시작하기

Chapter 1 소프트웨어공학의 소개

Chapter 2 UML을 이용한 모델링

Chapter 3 프로젝트 조직과 소통

Part 2 복잡성 다루기

Chapter 4 요구사항 도출

Chapter 5 분석

Chapter 6 시스템 설계: 시스템의 분해

Chapter 7 시스템 설계: 설계 목표 다루기

Chapter 8 객체 설계: 패턴 해법의 재사용

Chapter 9 객체 설계: 인터페이스의 지정

Chapter 10 모델을 코드로 매핑

Chapter 11 테스트

Part 3 변경 관리

Chapter 12 근거 관리

Chapter 13 형상 관리

Chapter 14 프로젝트 관리

Chapter 15 소프트웨어 생명주기

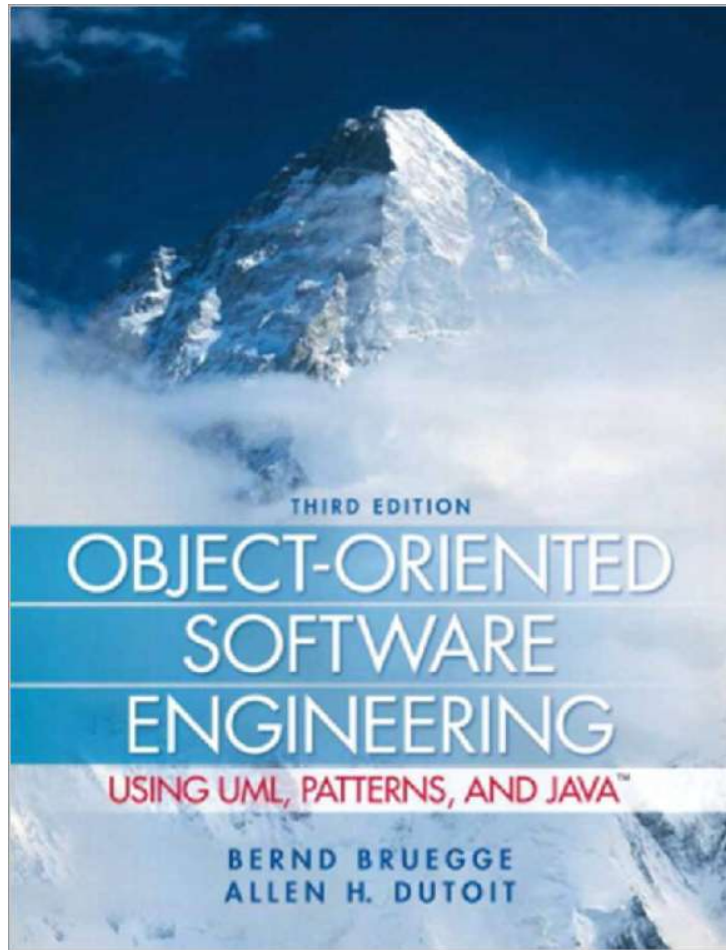
Chapter 16 방법론: 모든 것들의 종합

Part 4 부록

Appendix A 설계 패턴

Appendix B 용어 정리

Appendix C 참고문헌 / 찾아보기

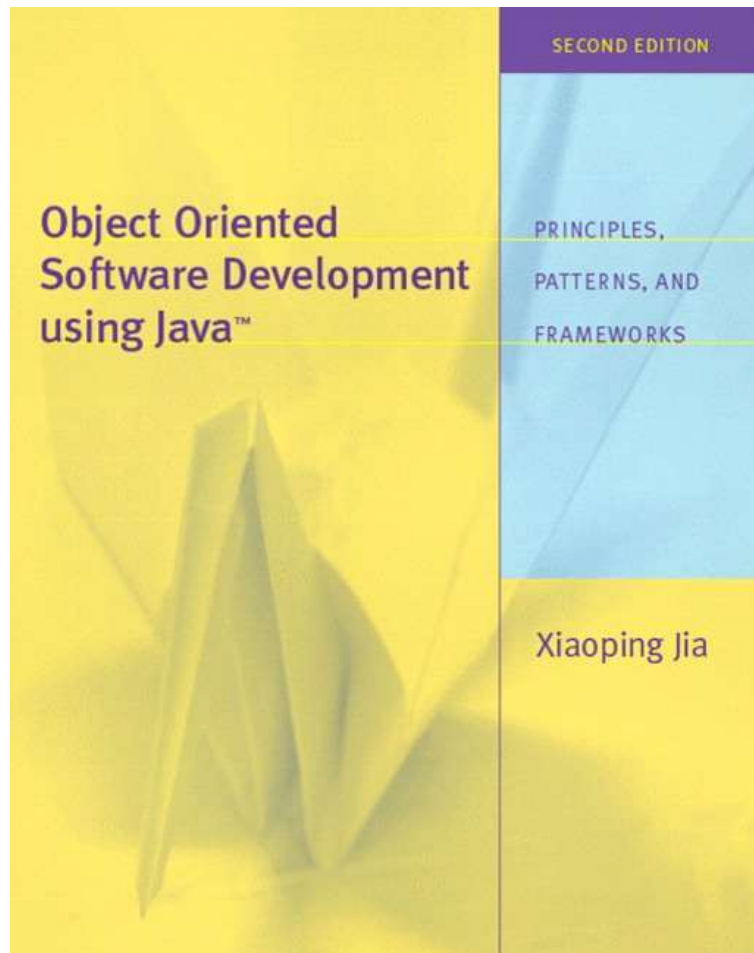


구매 불필요

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

Chapter 3: Introduction to Java

3.1: An Overview of Java 2 Platform

3.2: The Java Run-Time Architecture

3.3: Getting Started with Java

Chapter 4: Elements of Java

4.1: Lexical Elements

4.2: Variables and Types

4.3: Statements

4.4: Class Declarations

4.5: Packages

4.6: Exceptions

4.7: A Simple Animation Applet

Chapter 5: Classes and Inheritance

5.1: Overloading Methods and Constructors

5.2: Extending Classes

5.3: Extending and Implementing
Interfaces

5.4: Hiding Fields and Class Methods

5.5: Applications – Animation Applets

Chapter 6: From Building Blocks to Projects

6.1: Design and Implementation of
Classes

6.2: Contracts and Invariants

6.3: The Canonical Form of Classes

6.4: Unit Testing

6.5: Project Build

Chapter 7: Design by Abstraction

7.1: Design Patterns

7.2: Designing Generic Components

7.3: Abstract Coupling

7.4: Design Case Study – Animation of Sorting Algorithms

Chapter 8: Object-Oriented Application Frameworks

8.1: Application Frameworks

8.2: The Collections Framework

8.3: The Graphical User Interface Framework

8.4: The Input/Output Framework

Chapter 9: Design Case Study: A Drawing Pad

9.1: Planning

9.2: Iteration 1: A Simple Scribble Pad

9.3: Iteration 2: Menus, Options, and Files

9.4: Iteration 3: Refactoring

9.5: Iteration 4: Adding Shapes and Tools

9.6: Iteration 5: More Drawing Tools

9.7: Iteration 6: The Text Tool

Chapter 10: More Design Patterns

10.1: Type-Safe Enumeration Type

10.2: Creational Patterns

10.3: Behavioral Patterns

10.4: Structural Patterns

Chapter 11: Concurrent Programming

11.1: Threads

11.2: Thread Safety and Liveness

11.3: Design Case Study

Chapter 12: Distributed Computing

12.1: Socket-Based Communication

12.2: Remote Method Invocation

12.3: Java Database Connectivity (JDBC)

12.4: Common Object Request Broker Architecture (CORBA)

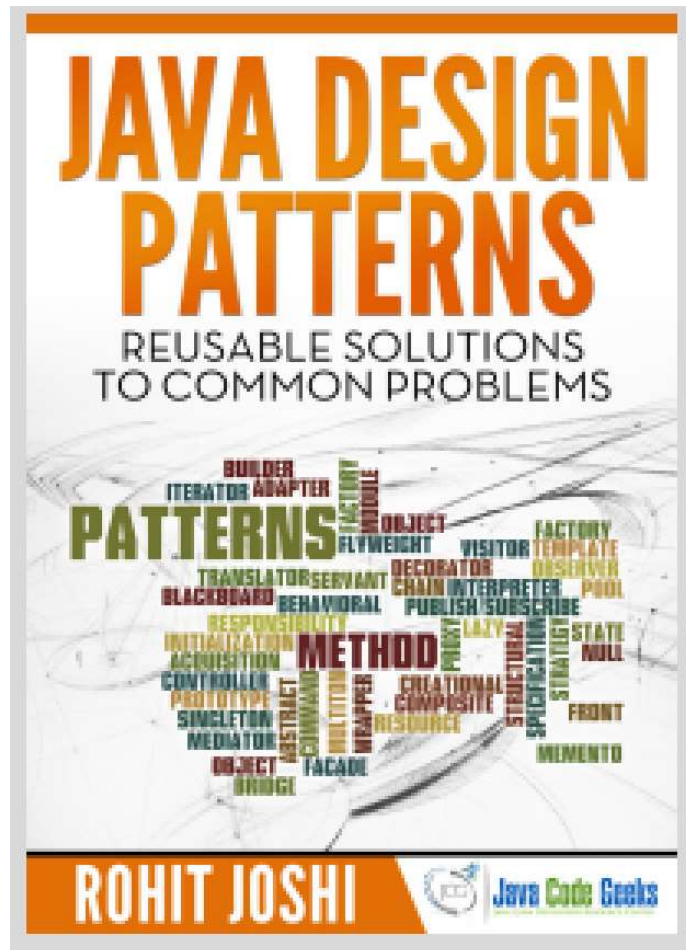
Appendix

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 - 콤포지트 (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 - 팩토리 메소드(10.2.3) 프로토타입(10.2.4) 빌더 (10.2.5)
- Chap10B1 - 커맨드(10.3.1) 어댑터(10.4.1)
- Chap10B2 - 콤포지트(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 development)

참고

도서



실전 개발을 위한 자바 디자인 패턴(###\$\$)



1부 : 디자인패턴 (생성패턴)

1. 싱글톤(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 Responsibility)
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. 싱글톤(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. 데코레이터(Decorator)
5. 퍼사드(Facade)
6. 플라이웨이트(Flyweight)
7. 프록시(Proxy)

6장 : 행위 패턴

1. 책임 연쇄(Chain of Responsibility)
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

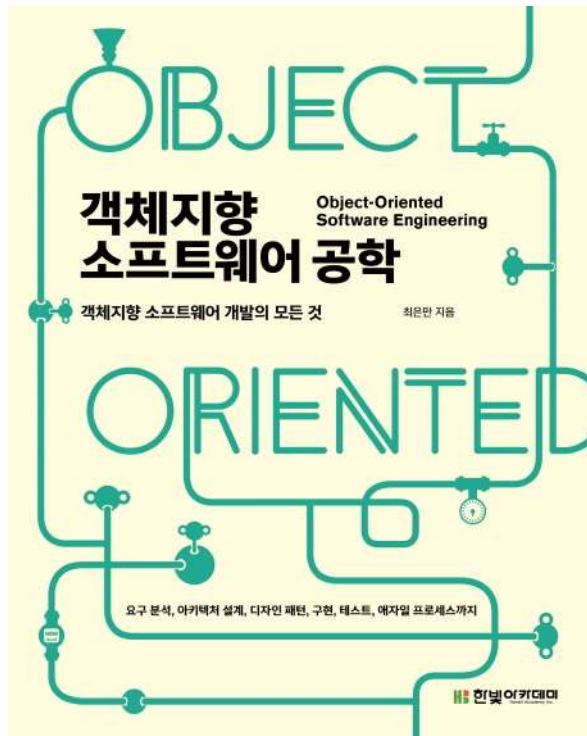
(객체지향 일반)

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, 오버레이션

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

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

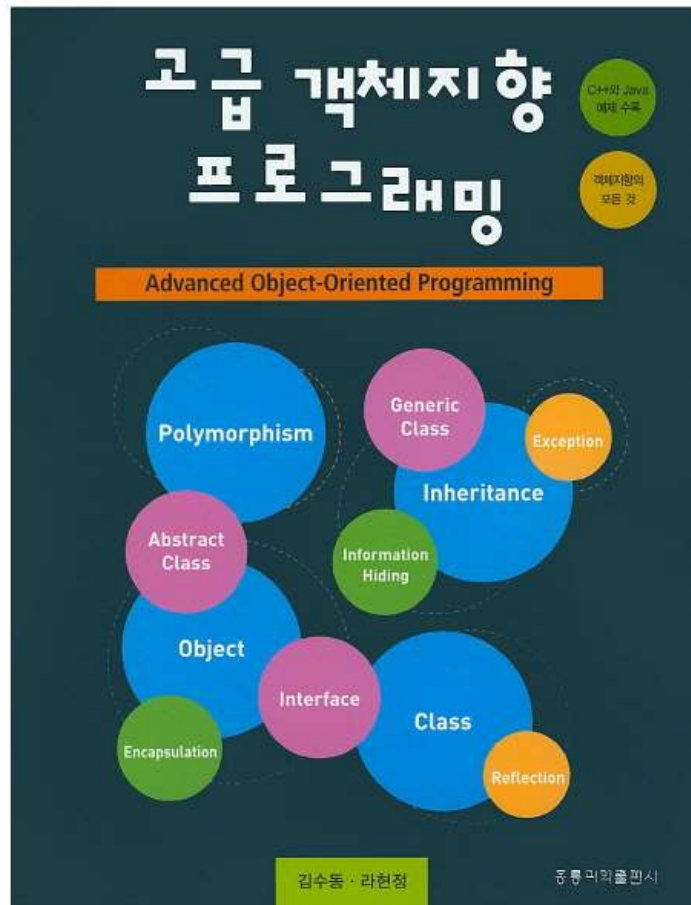
Chapter 10 인터페이스

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)

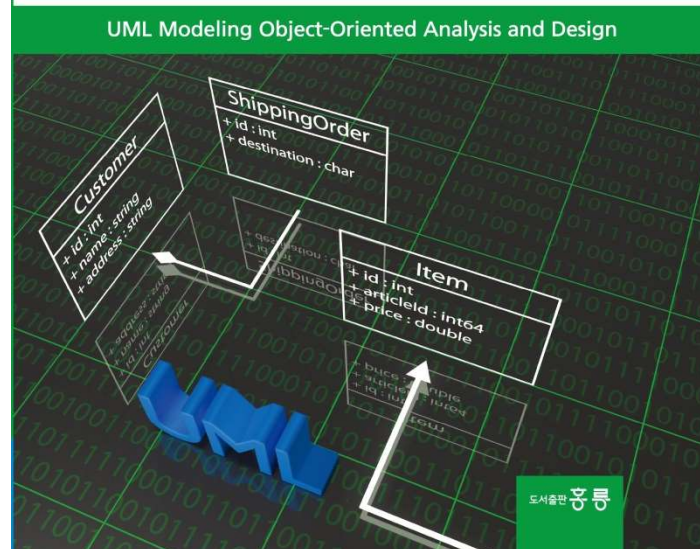
제 11장 객체지향 예외 처리
(Exception Handling)

제 12장 리플렉션(Reflection)

UML 모델링 객체지향 분석 및 설계 (2020) (X)

UML 모델링 객체지향 분석 및 설계

이성구 지음



PART 01 소개

CHAPTER 01 소프트웨어 공학 정의

CHAPTER 02 소프트웨어 개발 프로세스

CHAPTER 03 객체지향과 UML

PART 02 UML 다이어그램

CHAPTER 04 구조 다이어그램

CHAPTER 05 행위 다이어그램

CHAPTER 06 상호작용 다이어그램

CHAPTER 07 UML 확장

PART 03 객체지향 분석/설계

CHAPTER 08 요구 공학

CHAPTER 09 객체지향 분석 방법

CHAPTER 10 설계 개요

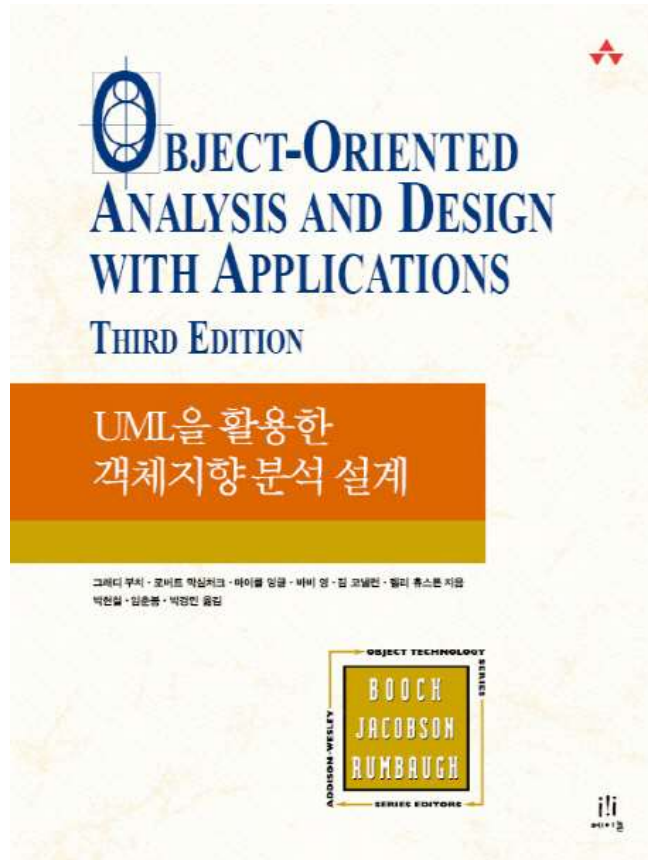
CHAPTER 11 시스템 아키텍처 설계

CHAPTER 12 객체지향 설계 방법

CHAPTER 13 설계 패턴

CHAPTER 14 모듈 설계

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

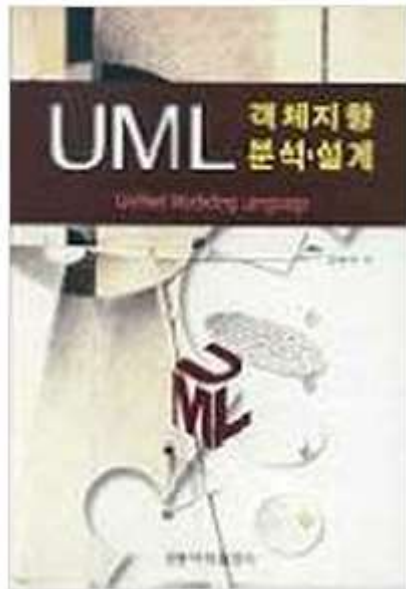
___A.2 스몰토크

___A.3 C++

___A.4 자바

부록B 권장도서 안내

UML 객체지향 분석.설계 (2000)

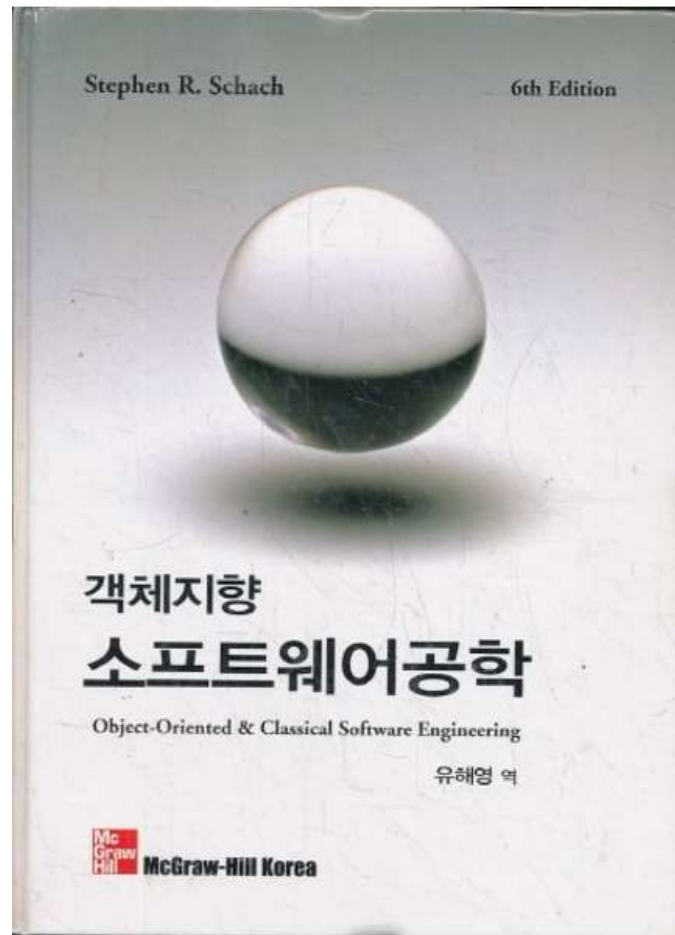


- 001. [UML 개요]....(13)
- 002. UML의 등장....(15)
- 003. UML의 뷰 및 다이어그램....(19)
- 004. 클래스....(27)
- 005. 관계....(35)
- 006. 확장 메커니즘....(43)
- 007. 고급 클래스....(49)
- 008. 고급 관계....(59)
- 009. 인터페이스 및 타입....(69)
- 010. 패키지....(75)

011. 인스턴스....(83)
012. 유스 케이스....(91)
013. 상호작용....(101)
014. 상호작용 다이어그램....(109)
015. 협동 및 패턴....(115)
016. 활동 다이어그램....(123)
017. 이벤트 및 신호....(133)
018. 프로세스 및 스레드....(141)
019. 상태 머신....(147)
020. 컴포넌트....(157)
021. 배치 및 배치 다이어그램....(167)
022. [객체지향 분석. 설계]....(175)
023. UML과 프로세스....(177)
024. 요구사항 분석....(187)
025. 개념 모델 구축....(199)

026. 시스템 행위....(211)
027. 시스템 설계와 상호작용....(221)
028. 책임 할당을 위한 기본 설계 패턴
....(227)
029. 설계 클래스 다이어그램....(257)
030. 시스템 설계 패턴....(267)
031. 시스템 구현....(277)
032. [반복적 객체지향 분석. 설계]....(289)
033. 사용 관계 및 타입 관계....(291)
034. 패키지와 개념 모델의 확장....(301)
035. 행위 모델링....(313)
036. 책임 할당을 위한 고급 설계 패턴
....(319)
037. 기타의 설계 패턴....(329)
038. 프레임워크 및 패턴....(347)
039. [용어 정의 및 참고문헌]....(363)
040. 찾아보기....(371)

객체지향 소프트웨어공학 (2005)(@@@) (X)



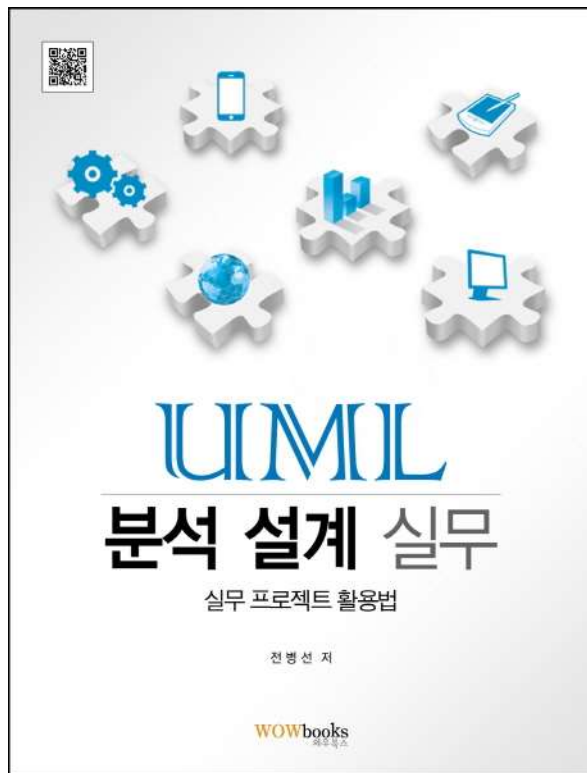
1부 소프트웨어 공학 개요

1. 소프트웨어 공학의 영역
2. 소프트웨어 생명주기 모델
3. 소프트웨어 프로세스
4. 팀
5. 툴의 선택
6. 테스트
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)(@@@)



들어가기

제1장. UML 분석 설계 개요

1.1 소프트웨어 개발의 어려움

1.2 분석 설계 방법으로서의 모델링의 필요성

1.3 모델, 소프트웨어 시스템의 추상화

1.4 UML, 모델링 언어

1.5 UML 분석 설계 과정

제2장. 구조 다이어그램

2.1 클래스 다이어그램

2.2 패키지 다이어그램

2.3 컴포넌트 다이어그램

2.4 배포 다이어그램

2.5 확장 메커니즘

제3장. 행위 다이어그램

3.1 활동 다이어그램

3.2 시퀀스 다이어그램

3.3 상태 다이어그램

3.4 유스케이스 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

Part 1 디자인 패턴과 친해지기

Chapter 01 Iterator – 순서대로
지정해서 처리하기

1_Iterator 패턴

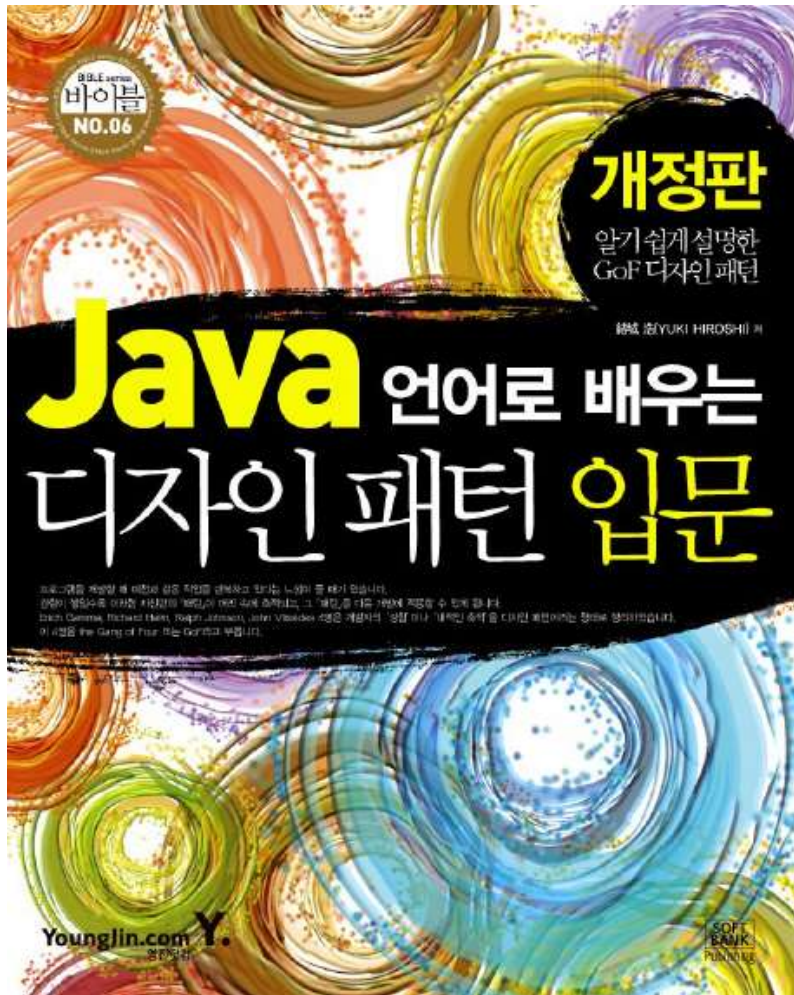
2_예제 프로그램

3_Iterator 패턴의 등장인물

4_독자의 사고를 넓히기 위한 힌
트

5_관련 패턴

Chapter 02 Adapter – 바꿔서 재
이용하기



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 Abstract 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

Part 8 상태를 관리하기

Chapter 17 Observer - 상태의 변화를 알려 주기

Chapter 18 Memento - 상태를 저장하기

Chapter 19 State - 상태를 클래스로 표현하기

Part 9 낭비 없애기

Chapter 20 Flyweight - 동일한 것을 공유해서 낭비 없애기

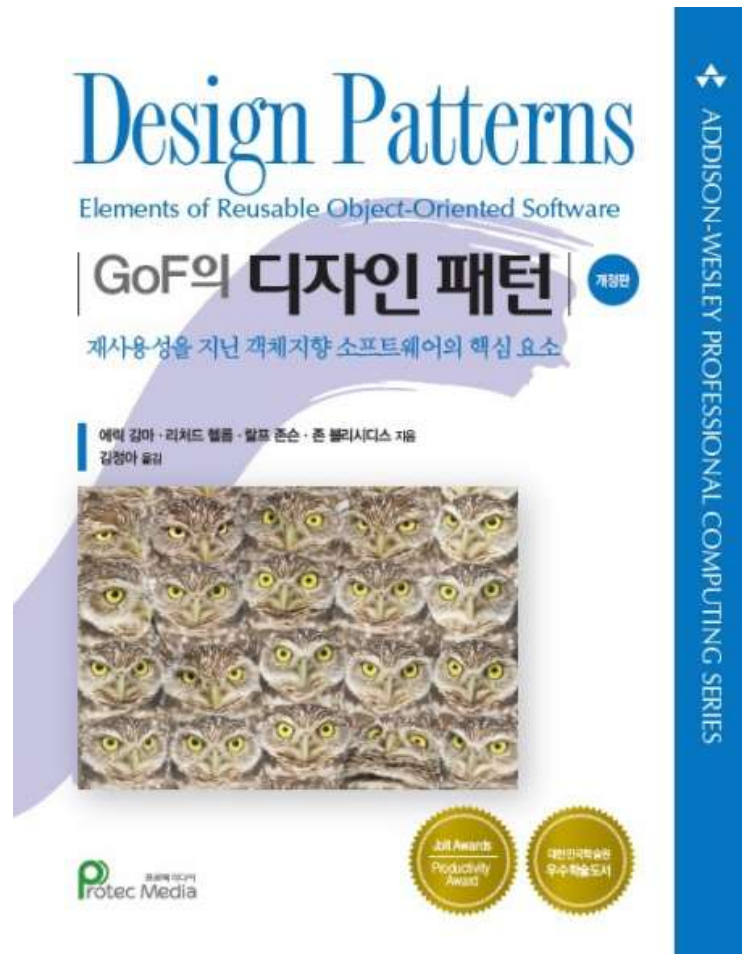
Chapter 21 Proxy - 필요해지면 만들기

Part 10 클래스로 표현하기

Chapter 22 Command - 명령을 클래스로 하기

Chapter 23 Interpreter - 문법규칙을 클래스로 표현하기

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 사
용된 기본 클래스

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



Part 01 객체지향

Chapter 01 들어가기

1. 지저분해지는 코드
2. 수정하기 좋은 구조를 가진 코드
3. 소프트웨어의 가치

Chapter 02 객체 지향

1. 절차 지향과 객체 지향
 - 1.1 절차 지향
 - 1.2 객체 지향
2. 객체(Object)
 - 2.1 객체의 핵심은 기능을 제공하는 것
 - 2.2 인터페이스와 클래스
 - 2.3 메시지
3. 객체의 책임과 크기
4. 의존
 - 4.1 의존의 양면성
5. 캡슐화
 - 5.1 절차 지향 방식 코드
 - 5.2 캡슐화 된 기능 구현
 - 5.3 캡슐화의 결과는 내부 구현 변경의 유연성 획득
 - 5.4 캡슐화를 위한 두 개의 규칙
6. 객체 지향 설계 과정

Chapter 03 다형성과 추상 타입

1. 상속 개요
2. 다형성과 상속
 - 2.1 인터페이스 상속과 구현 상속
3. 추상 타입과 유연함
 - 3.1 추상 타입과 실제 구현의 연결
 - 3.2 추상 타입을 이용한 구현 교체의 유연함
 - 3.3 변화되는 부분을 추상화하기
 - 3.4 인터페이스에 대고 프로그래밍하기
 - 3.5 인터페이스는 인터페이스 사용자 입장에서 만들기
 - 3.6 인터페이스와 테스트

Chapter 04 재사용: 상속보단 조립

1. 상속과 재사용
 - 1.1 상속을 통한 재사용의 단점 1, 상위 클래스 변경의 어려움
 - 1.2 상속을 통한 재사용의 단점 2, 클래스의 불필요한 증가
 - 1.3 상속을 통한 재사용의 단점 3, 상속의 오용
2. 조립을 이용한 재사용
 - 2.1 위임
 - 2.2 상속은 언제 사용하나?

Part 02 설계 원칙 / DI와 서비스 로케이터

Chapter 05 설계 원칙: SOLID

1. 단일 책임 원칙(Single responsibility principle)
 - 1.1 단일 책임 원칙 위반이 불러오는 문제점
 - 1.2 책임이란 변화에 대한 것
2. 개방 폐쇄 원칙(Open-closed principle)
 - 2.1 개방 폐쇄 원칙이 깨질 때의 주요 증상
 - 2.2 개방 폐쇄 원칙은 유연함에 대한 것
3. 리스코프 치환 원칙(Liskov substitution principle)
 - 3.1 리스코프 치환 원칙을 지키지 않을 때의 문제
 - 3.2 리스코프 치환 원칙은 계약과 확장에 대한 것
4. 인터페이스 분리 원칙(Interface segregation principle)
 - 4.1 인터페이스 변경과 그 영향
 - 4.2 인터페이스 분리 원칙
 - 4.3 인터페이스 분리 원칙은 클라이언트에 대한 것
5. 의존 역전 원칙(Dependency inversion principle)
 - 5.1 고수준 모듈이 저수준 모듈에 의존할 때의 문제
 - 5.2 의존 역전 원칙을 통한 변경의 유연함 확보
 - 5.3 소스 코드 의존과 런타임 의존
 - 5.4 의존 역전 원칙과 패키지
6. SOLID 정리

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

Part 03 07 주요 디자인 패턴

Chapter 07 주요 디자인 패턴

1. 디자인 패턴이란?

2. 전략(Strategy) 패턴

3. 템플릿 메서드(Template Method) 패턴

3.1 상위 클래스가 흐름 제어 주체

3.2 템플릿 메서드와 전략 패턴의 조합

4. 상태(State) 패턴

4.1 상태 변경은 누가?

5. 데코레이터(Decorator) 패턴

5.1 데코레이터 패턴을 적용할 때 고려할 점

6. 프록시(proxy) 패턴

6.1 프록시 패턴을 적용할 때 고려할 점

7. 어댑터(Adapter) 패턴

8. 옵저버(Observer) 패턴

8.1 옵저버 객체에게 상태 전달 방법

8.2 옵저버에서 주제 객체 구분

8.3 옵저버 패턴 구현의 고려 사항

9. 미디에이터(Mediator) 패턴

9.1 추상 미디에이터 클래스의 재사용

10. 파사드(Facade) 패턴

10.1 파사드 패턴의 장점과 특징

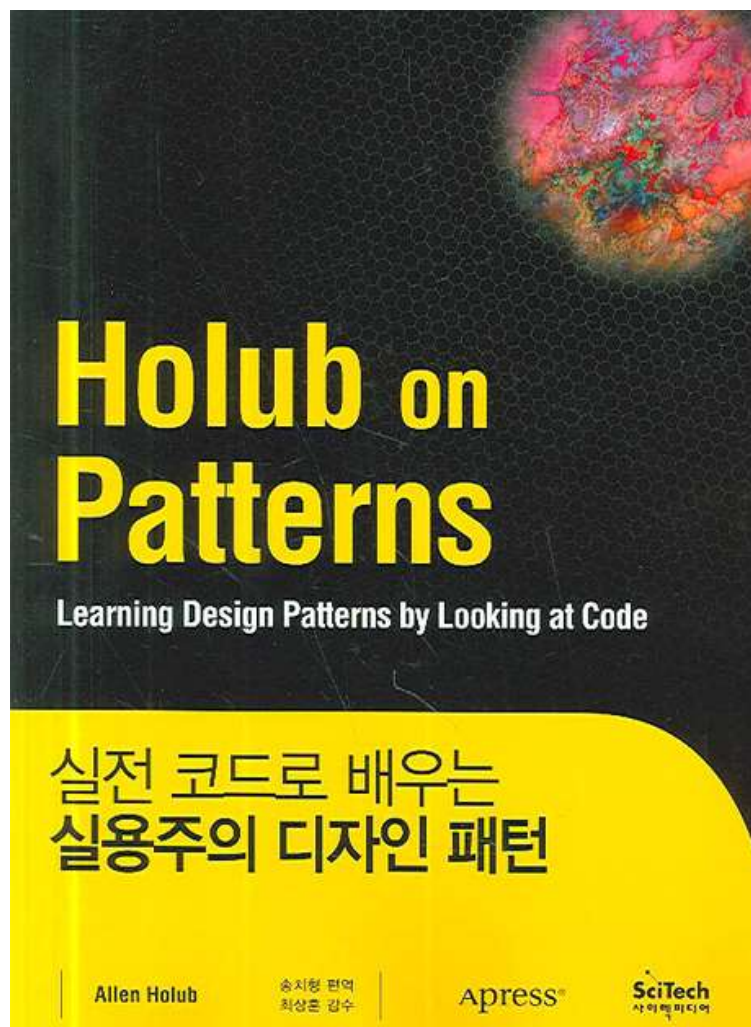
11. 추상 팩토리(Abstract Factory) 패턴

12. 컴포지트(Composite) 패턴

12.1 컴포지트 패턴 구현의 고려 사항

13. 널(Null) 객체 패턴

실전 코드로 배우는 실용주의 디자인 패턴(X)



목차

0장 소프트웨어 설계의 고고학

1장 OO와 디자인 패턴 기초 다지기

2장 인터페이스로 프로그래밍하기 그리고 몇 개의 생성 패턴

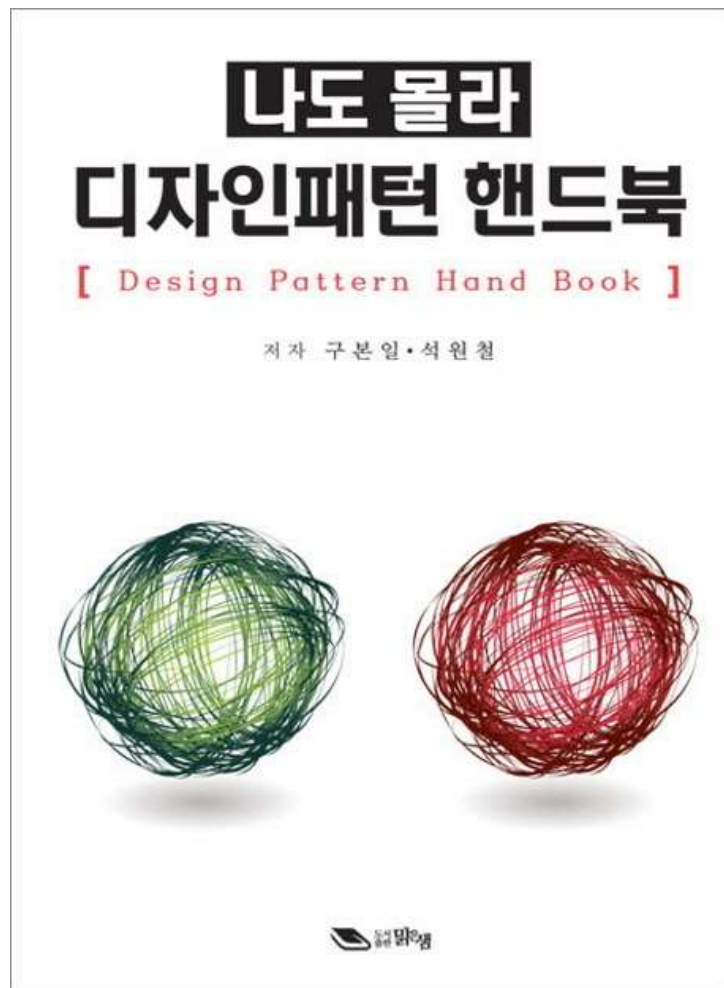
3장 라이프 게임

4장 소형 데이터베이스 SQL 구현하기

부록 디자인 패턴 쿼크 레퍼런스

찾아보기

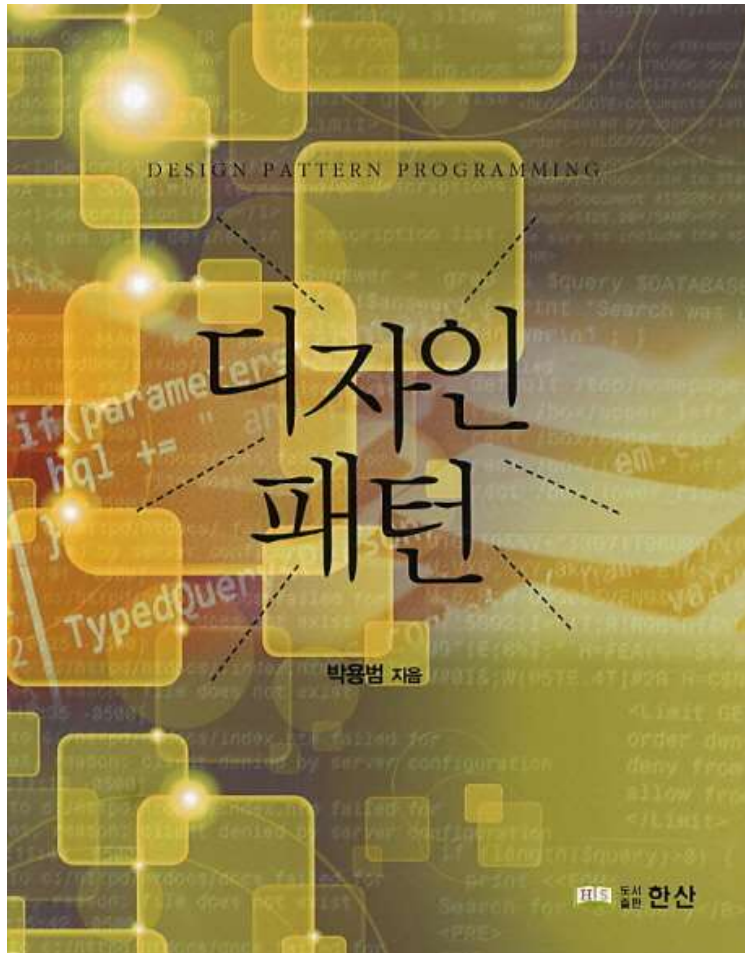
나도 몰라 디자인 패턴 핸드북



머리말

1. Why Design Pattern _ 9
2. Template Method Pattern _ 13
3. Factory Pattern _ 21
4. Adapter Pattern _ 37
5. Chain Of Responsibility Pattern _ 45
6. Decorator Pattern _ 55
7. Prototype Pattern _ 75
8. State Pattern _ 85

디자인 패턴(박용범, 한산)

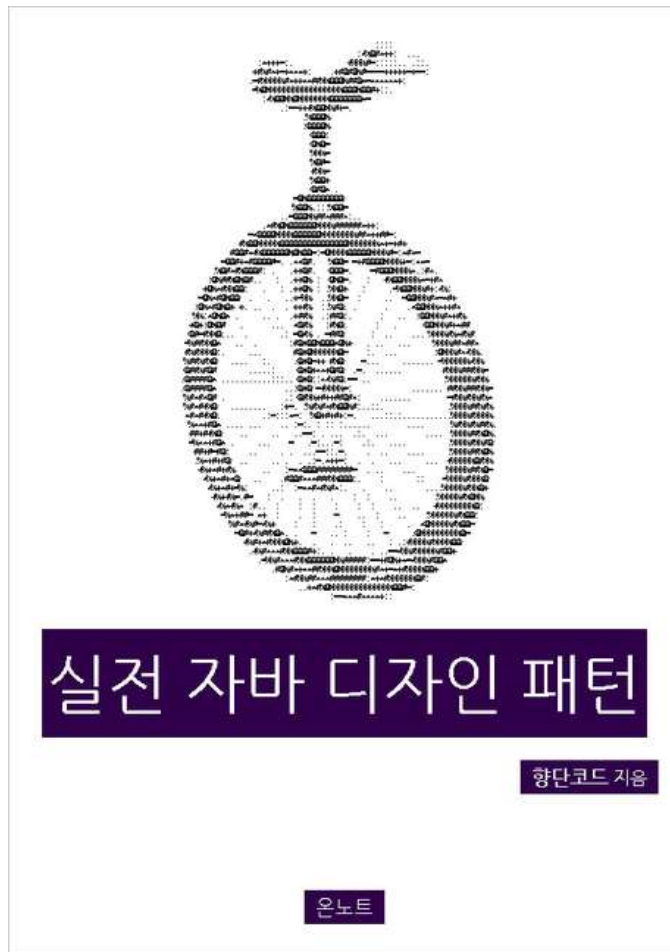


목차

- 제1장 생성자의 한계
- 제2장 Abstract Factory Pattern
- 제3장 Factory Method Pattern
- 제4장 Singleton Pattern
- 제5장 Prototype Pattern
- 제6장 Adapter Pattern
- 제7장 Composition Pattern
- 제8장 Decorator Pattern
- 제9장 Proxy Pattern
- 제10장 Observer Pattern
- 제11장 Strategy Pattern
- 제12장 State Pattern
- 제13장 Command Pattern
- 제14장 Iterator Pattern
- 제15장 Template Method Pattern
- 제16장 MVC Pattern

실전 자바 디자인 패턴(###)

- (앞의 책과 8장까진
같음)



9장 : 계산기 프로젝트

1. 계산기 설명
2. 책임 연쇄 패턴 적용하기
3. 수학적 함수 피연산자 정의하기
4. 계산기 화면 만들기
5. 모델 만들기
6. Command 패턴 적용하기
7. 사칙연산 처리하기
8. State 패턴 적용하기
9. 계산 초기화
10. 수학적 함수 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

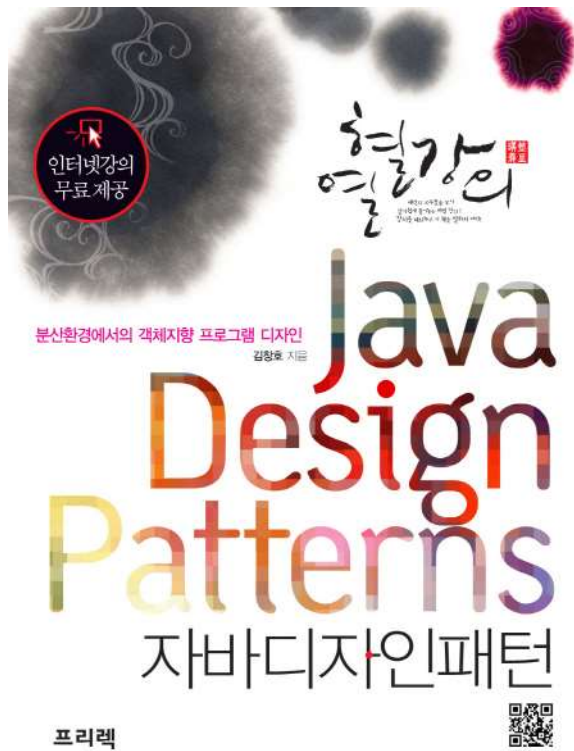
11장 : 게임 프로젝트

1. 게임 설명
2. 게임 화면 만들기
3. 모델 만들기
4. MVC 패턴 적용하기
5. 타원형 스프라이트 만들기
6. Singleton 패턴의 이미지 저장소 만들기
7. 이미지 스프라이트를 위한 Flyweight 패턴 적용하기
8. 텍스트 스프라이트 만들기
9. Factory를 이용한 객체 생성하기
10. 컴포지트 패턴의 복합 스프라이트 만들기
11. Abstract Factory 패턴 적용하기
12. Strategy 패턴 적용하기
13. State 패턴 적용하기
14. 마우스 이벤트 처리하기
15. 키 이벤트 처리하기
16. 게임로직 구현하기
17. 전체 소스

12장 : 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

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 생성
은 의뢰하자

- 2.1 객체 생성
- 2.2 Factory Method 패턴이란?

Chapter 3 객체지향 – 패턴 이해를 위한 객체
지향 개념

- 3.1 추상화란?
- 3.2 추상화의 보편적 예들
- 3.3 추상화의 관점에서 본 패턴

(#)

Chapter 4 Prototype 패턴 - 객체 생성은 복사를 통해서

Chapter 5 Facade 패턴 - 제대로 곁잡기

Chapter 6 Singleton 패턴과 Object Pool 패턴 - 다수를 위한 하나

Chapter 7 Immutable 패턴과 Flyweight 패턴 - 변하지 않는 가벼움

Chapter 8 Command 패턴과 Mediator 패턴 - 중개인을 통해 명령하기

Chapter 9 Observer 패턴 - 상태 변화 알려주기

Chapter 10 Composite 패턴 - 조직 구조 표현하기

Chapter 11 Builder 패턴과 Chain of Responsibility 패턴 - 건축 방식은 하나, 책임감은 연계하기

Chapter 12 Iterator 패턴 - 집합체의 구성물 보여주기

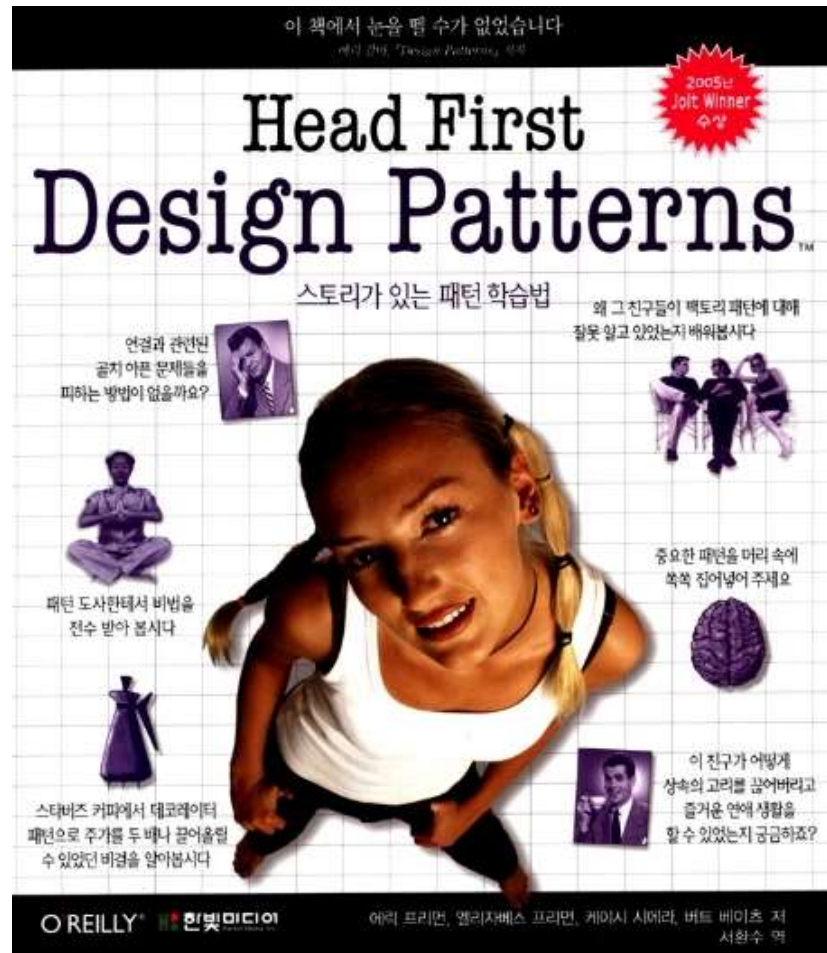
Chapter 13 Strategy 패턴 - 전략은 하나, 전술은 다양하게

Chapter 14 Bridge 패턴 - 험한 세상 다리가 되어

Chapter 15 부록 A RMI - 원격지의 것은 내 것이다

Chapter 16 부록 B Swing - 스윙은 아름답게

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

2. 옵저버 패턴
3. 데코레이터 패턴
4. 팩토리 패턴
5. 싱글턴 패턴
6. 커맨드 패턴
7. 어댑터 패턴과 퍼사드 패턴
8. 템플릿 메소드 패턴
9. 이터레이터와 컴포지트 패턴
10. 스테이트 패턴
11. 프록시 패턴
12. 컴파운드 패턴
13. 패턴과 함께 하는 행복한 삶

14. 부록: 기타 패턴
- 브리지 패턴
- 빌더 패턴
- 역할 사슬 패턴
- 플라이웨이트 패턴
- 인터프리터 패턴
- 미디어이터 패턴
- 메멘토 패턴
- 프로토타입 패턴
- 비지터 패턴

UML과 디자인 패턴의 적용 (2005)

Part 1 소개
(INTRODUCTION)

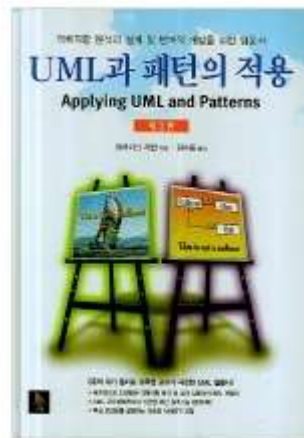
Part 2 도입
(INCEPTION)

Part 3 반복단계 1

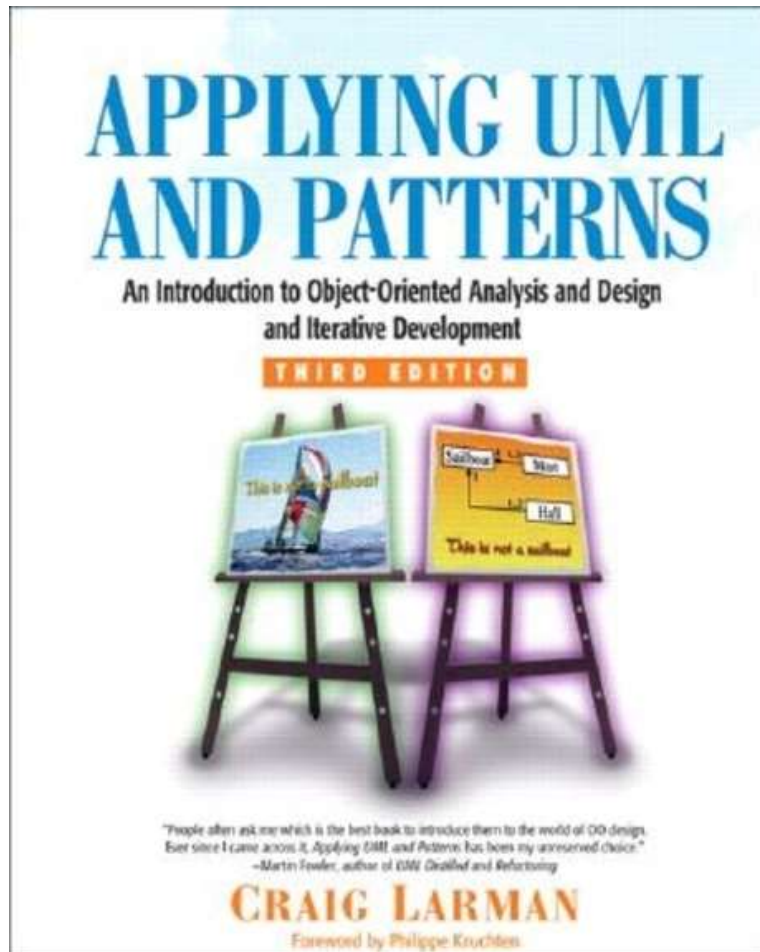
Part 4 반복단계 2

Part 5 반복단계 3

Part 6 특별한 논제들

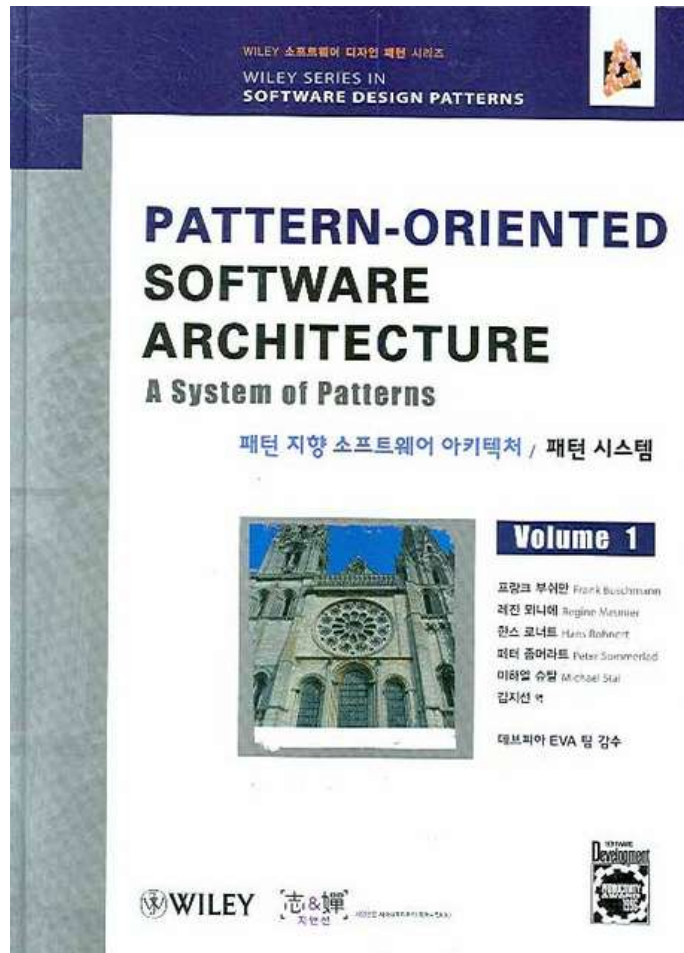


Applying UML and Patterns 3/E



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

패턴 지향 소프트웨어 아키텍처 패턴 시스템



- VOLUME. 1 WILEY 소프트웨어 디자인 패턴 시리즈 | 양장
- 프랑크 부쉬만, 레진 뢰니에, 한스 로너트, 페터 쏜머라트, 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

1장 매직 넘버를 기호 상수로 치환 _소스 코드에 100이라고 적힌 경우

1.1 리팩토링

1.1.1 매직 넘버를 기호 상수로 치환

1.2 예제 프로그램

1.2.1 리팩토링 전

1.2.2 리팩토링 실행

1.2.3 리팩토링 후

1.3 한 걸음 더 나아가기

1.3.1 분류 코드를 클래스로 치환하기

1.3.2 enum

1.3.3 기호 상수가 적합하지 않은 경우

1.3.4 바이트 코드에 내장된 상수에 주의하기

2장 제어 플래그 삭제 _제어 플래그 때문에 코드가 읽기 어려운 경우

2.1 리팩토링

2.1.1 제어 플래그 삭제

2.2 예제 프로그램(FindInt)

2.2.1 리팩토링 전

2.2.2 리팩토링 실행(break 사용)

2.2.3 리팩토링 실행(return 사용)

2.3 예제 프로그램(SimpleDatabase)

2.3.1 리팩토링 전

2.3.2 리팩토링 실행

2.4 한 걸음 더 나아가기

2.4.1 break나 return을 쓰면 가독성이 좋아지는 이유

2.4.2 인스턴스 필드로 만든 제어 플래그의 위험성

2.4.3 플래그명

2.4.4 boolean 이외의 플래그

2.4.5 정규 표현식 패키지 사용

3장 어서션 도입 _‘이렇게 될 것이다’라는 주석이 있는 경우

3.1 리팩토링

3.1.1 어서션 도입

3.2 예제 프로그램

3.2.1 리팩토링 전

3.2.2 리팩토링 실행

3.2.3 리팩토링 후

3.2.4 컴파일과 실행

3.3 어서션 동작 확인

3.4 한 걸음 더 나아가기

3.4.1 자바 어서션 문법

3.4.2 어서션은 예외 처리를 대신할 수 없음

3.4.3 자바 어서션은 클래스 라이브러리가 아님

3.4.4 어서션 완전 삭제

3.4.5 다른 언어 환경의 어서션

4장 널 객체 도입 _null 확인이 너무 많은 경우

4.1 리팩토링

4.1.1 널 객체 도입

4.2 예제 프로그램

4.2.1 리팩토링 전

4.2.2 리팩토링 실행

4.2.3 리팩토링 후

4.3 한 걸음 더 나아가기

4.3.1 팩토리 메서드 패턴

4.3.2 싱글톤 패턴

4.3.3 널 객체로 중첩 클래스 사용

4.3.4 null 확인은 나쁜가

4.3.5 패턴 중독에 빠지지 않기

4.3.6 상수와 널 객체

4.3.7 isNull 메서드는 필요한가

4.3.8 기존 클래스를 수정할 수 없다면

5장 메서드 추출 _코드가 너무 길어서 읽기 어려운 경우

5.1 리팩토링

5.1.1 메서드 추출

5.2 예제 프로그램

5.2.1 리팩토링 전

5.2.2 리팩토링 실행

5.2.3 리팩토링 후

5.3 한 걸음 더 나아가기

5.3.1 역 리팩토링

5.3.2 메서드 추출은 당연한가?

5.3.3 메서드가 길어지는 경우

6장 클래스 추출 _클래스의 책임이 너무 많은 경우

6.1 리팩토링

6.1.1 클래스 추출

6.2 예제 프로그램

6.2.1 리팩토링 전

6.2.2 리팩토링 실행

6.2.3 리팩토링 후

6.3 한 걸음 더 나아가기

6.3.1 양방향 링크는 피한다

6.3.2 기능 추가와 리팩토링

6.3.3 불변 인터페이스

6.3.4 역 리팩토링: 클래스 인라인화

7장 분류 코드를 클래스로 치환 _int로 객체를 구분하는 경우

7.1 리팩토링

7.1.1 분류 코드를 클래스로 치환

7.2 예제 프로그램

7.2.1 리팩토링 전

7.2.2 기본 타입을 사용한 분류 코드의 문제점

7.2.3 리팩토링 실행

7.2.4 리팩토링 후

7.3 한 걸음 더 나아가기

7.3.1 기본 타입을 사용한 분류 코드의 문제점

7.3.2 enum

8장 분류 코드를 하위 클래스로 치환 _분류 코드마다 동작이 다른 경우(1)

8.1 리팩토링

8.1.1 분류 코드를 하위 클래스로 치환

8.1.2 구조와 동작

8.2 예제 프로그램

8.2.1 리팩토링 전

8.2.2 리팩토링 실행

8.2.3 리팩토링 후

8.3 한 걸음 더 나아가기

8.3.1 switch 문과 instanceof 연산자가 풍기는 악취

8.3.2 객체 생성 switch 문 삭제

8.3.3 팩토리 메서드 여러 개 준비하기

8.3.4 어디까지 리팩토링해야 하나

9장 분류 코드를 상태/전략 패턴으로 치환 _ 분류 코드마다 동작이 다른 경우(2)

9.1 리팩토링

9.1.1 분류 코드를 상태/전략 패턴으로 치환

9.2 예제 프로그램

9.2.1 리팩토링 전

9.2.2 리팩토링 실행

9.2.3 리팩토링 후

9.3 코드 추가 수정

9.3.1 enum 사용

9.3.2 상태 의존 코드를 상태 객체로 이동

9.3.3 코드 추가 수정

9.4 한 걸음 더 나아가기

9.4.1 분류 코드를 치환하는 세 가지 방법 비교

9.4.2 상태 패턴과 전략 패턴의 차이

9.4.3 다형적 해결로 default 제거

10장 에러 코드를 예외로 치환 _에러 처리가 흩어져 있는 경우

10.1 리팩토링

10.1.1 에러 코드를 예외로 치환

10.2 예제 프로그램

10.2.1 리팩토링 전

10.2.2 리팩토링 실행

10.2.3 리팩토링 후

10.3 코드 추가 수정

10.3.1 분류 코드를 상태/전략 패턴으로 치환

10.4 한 걸음 더 나아가기

10.4.1 검사 예외와 비검사 예외

10.4.2 예외 계층

10.4.3 java.io.EOFException에 대해

10.4.4 비검사 예외와 사전 확인용 메서드

11장 생성자를 팩토리 메서드로 치환 _클래스 이름이 new로 하드 코딩된 경우

11.1 리팩토링

11.1.1 생성자를 팩토리 메서드로 치환

11.2 예제 프로그램

11.2.1 리팩토링 전

11.2.2 리팩토링 실행

11.2.3 리팩토링 후

11.2.4 분류 코드를 하위 클래스로 치환 리팩토링 실행

11.2.5 리팩토링 후

11.3 한 걸음 더 나아가기

11.3.1 프로바이더

11.3.2 매개변수 문제

11.3.3 팩토리 메서드와 생성 메서드

12장 관측 데이터 복제 _모델과 뷰가 뒤섞여 있는 경우

12.1 리팩토링

12.1.1 관측 데이터 복제

12.2 예제 프로그램

12.2.1 리팩토링 전

12.2.2 리팩토링 실행

12.2.3 리팩토링 후

12.3 한 걸음 더 나아가기

12.3.1 이벤트에 포함된 정보

12.3.2 무엇을 이벤트 리스너로 할 것인가

12.3.3 무한 반복에 주의

12.3.4 다른 이벤트 리스너 추가하기

13장 상속을 위임으로 치환 _IS-A 관계가 아닌데 상속하고 있는 경우

13.1 상속과 위임

13.1.1 상속

13.1.2 위임

13.1.3 상속과 위임 비교

13.2 리팩토링

13.2.1 상속을 위임으로 치환

13.3 예제 프로그램

13.3.1 리팩토링 전

13.3.2 새로운 의문

13.3.3 리팩토링 실행

13.3.4 리팩토링 후

13.4 생성자 연쇄

13.5 한 걸음 더 나아가기

13.5.1 상속은 최후의 무기

13.5.2 리스코프 치환 원칙

13.5.3 IS-A 관계와 HAS-A 관계

14장 대리자 은폐 _위임 대상까지 노출되어 있는 경우

14.1 리팩토링

14.1.1 대리자 은폐

14.2 예제 프로그램

14.2.1 리팩토링 전

14.2.2 1회째 리팩토링 실행

14.2.3 1회째 리팩토링 후

14.2.4 2회째 리팩토링 실행

14.2.5 2회째 리팩토링 후

14.3 한 걸음 더 나아가기

14.3.1 '숨기기'의 중요성

14.3.2 다양한 은폐

14.3.3 중개자 제거

14.3.4 클래스 인라인화

14.3 한 걸음 더 나아가기

14.3.1 '숨기기'의 중요성

14.3.2 다양한 은폐

14.3.3 중개자 제거

14.3.4 클래스 인라인화

15장 상속 구조 정리 _상속이 엉켜 있는 경우

15.1 리팩토링

15.1.1 상속 구조 정리

15.2 예제 프로그램

15.2.1 리팩토링 전

15.2.2 의문점

15.2.3 리팩토링 실행

15.2.4 리팩토링 후

15.3 한 걸음 더 나아가기

15.3.1 직교성

15.3.2 상속과 @Override

15.4 정리

15.5 연습 문제

15.6 연습 문제 해답

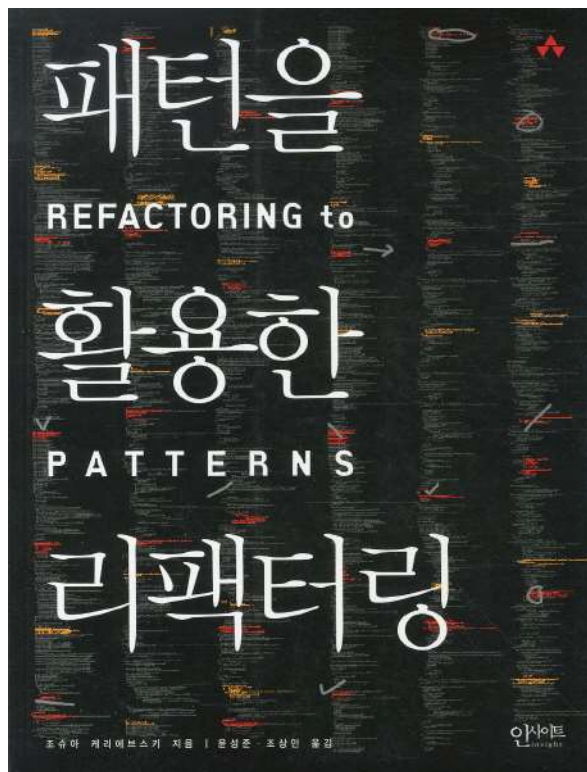
부록

부록 A 리팩토링 목록

부록 B JUnit

부록 C 참고 문헌과 웹 사이트

패턴을 활용한 리팩터링 (2011- 도서관)



01장_ 이책을 쓴 이유

02장_ 리팩터링

03장_ 패턴

04장_ 코드 속의 냄새

05장_ 패턴을 고려한 리팩터링 카탈로그

06장_ 생성

07장_ 단순화

08장_ 일반화

09장_ 보호

10장_ 축적

11장_ 유틸리티

후기 by John Brant, Don Roberts
refactoring to patterns 칭찬의 말

참고문헌

인덱스

이 책을 읽어야 하는 사람
필요한 배경지식
이 책의 사용법
이 책의 역사
거인들의 어깨에 기대어
감사의 글

1장. 이 책을 쓴 이유
과도한 설계
패턴 만능주의
미진한 설계
테스트 주도 개발과 지속적인 리팩터링
리팩터링과 패턴
발전적 설계

2장. 리팩터링
리팩터링이란?
리팩터링을 하는 이유
많은 눈
사람이 읽기 쉬운 코드
깔끔하게 유지하기
작은 단계
설계 부채
새로운 아키텍처 발전시키기
복합 리팩터링과 테스트 주도 리팩터링
복합 리팩터링의 장점
리팩터링 도구

3장 패턴
패턴이란?
패턴 중독
패턴을 구현하는 다양한 방법
패턴 목표, 패턴 지향, 패턴 제거 리팩터링
패턴은 코드를 더 복잡하게 만드는가?
패턴 지식
패턴을 이용한 사전 설계

4장. 코드 속의 냄새

중복된 코드

긴 메서드

복잡한 조건문

기본 타입에 대한 강박관념

추잡한 노출

문어발 솔루션

인터페이스가 서로 다른 대체 클래스

게으른 클래스

거대한 클래스

Switch 문

조합의 폭발적 증가

괴짜 솔루션

5장. 패턴을 고려한 리팩터링 카탈로그

리팩터링 형식

카탈로그에서 참조한 프로젝트

시작점

학습 순서

6장 생성

Replace Constructors with Creation Methods

Move Creation Knowledge to Factory

Encapsulate Classes with Factory

Introduce Polymorphic Creation with Factory Method

Encapsulate Composite with Builder

Inline Singleton

7장 단순화

Compose Method

Replace Conditional Logic with Strategy

Move Embellishment to Decorator

Replace State-Altering Conditionals with State

Replace Implicit Tree with Composite

Replace Conditional Dispatcher with Command

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 Com- plexity, 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 Obses- sion, 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 다
른 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 (254)

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)

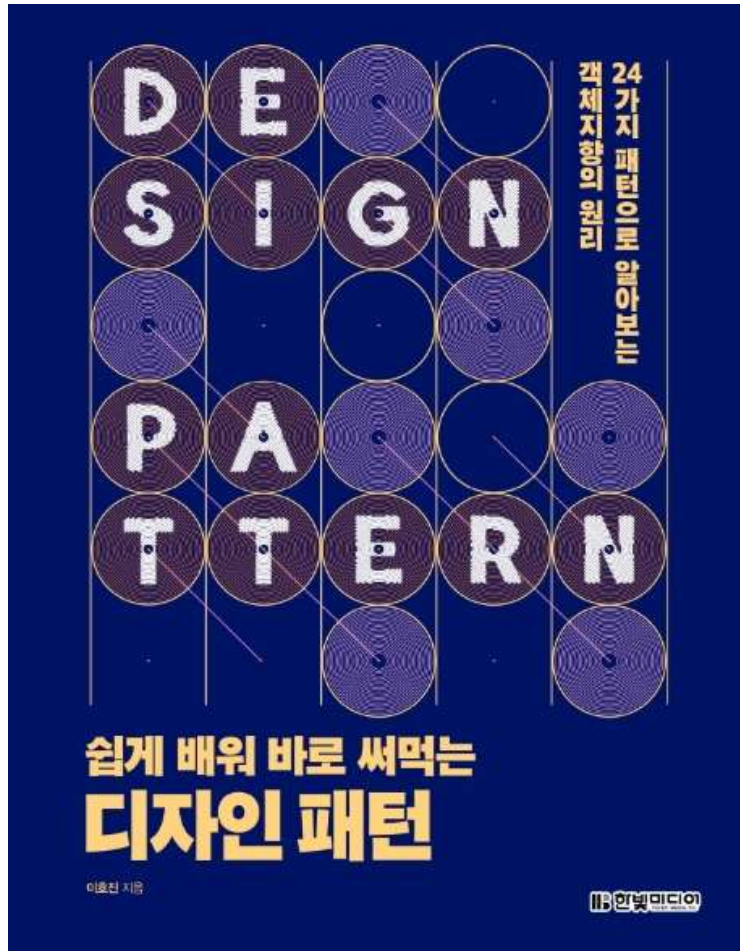
표 5.1

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

CHAPTER 2 싱글턴 패턴

CHAPTER 3 팩토리 메서드 패턴

CHAPTER 4 추상 팩토리 패턴

CHAPTER 5 빌더 패턴

CHAPTER 6 프로토타입 패턴

[PART 2 구조 패턴 - 상속과 합성을 사용해 객체 확장하기]

CHAPTER 7 어댑터 패턴

CHAPTER 8 브리지 패턴

CHAPTER 9 복합체 패턴

CHAPTER 10 장식자 패턴

CHAPTER 11 파사드 패턴

CHAPTER 12 플라이웨이트 패턴

CHAPTER 13 프록시 패턴

[PART 3 행동 패턴 - 복합 구조로 객체의 책임 분산하기]

CHAPTER 14 반복자 패턴

CHAPTER 15 명령 패턴

CHAPTER 16 방문자 패턴

CHAPTER 17 체인 패턴

CHAPTER 18 감시자 패턴

CHAPTER 19 중재자 패턴

CHAPTER 20 상태 패턴

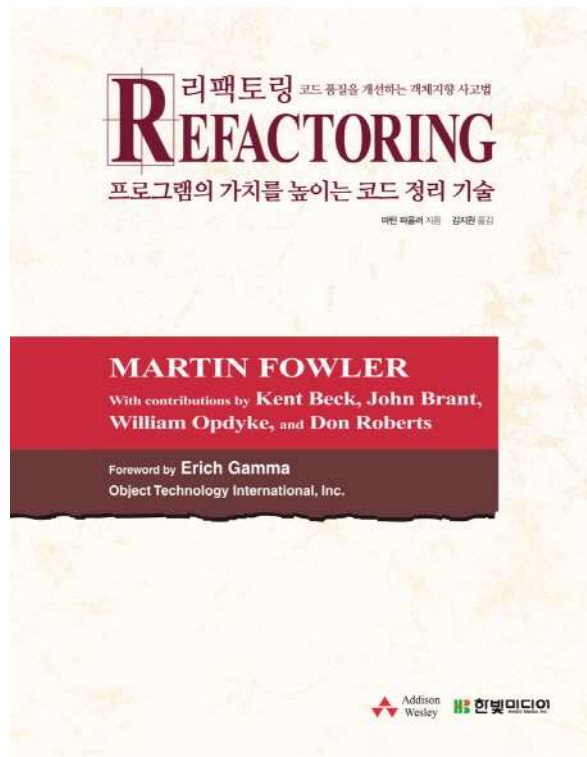
CHAPTER 21 메멘토 패턴

CHAPTER 22 템플릿 메서드 패턴

CHAPTER 23 전략 패턴

CHAPTER 24 인터프리터 패턴

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



CHAPTER 01 맛보기 예제

원래의 프로그램
리팩토링 첫 단계
statement 에서드 분해와 기능 재분배
가격 책정 부분의 조건문을 재정의로 교체
고찰

CHAPTER 02 리팩토링 개론

리팩토링은 무엇인가
리팩토링은 왜 해야 하나
리팩토링은 어떨 때 필요한가
팀장에게 어떻게 말을 꺼내나
리팩토링 관련 문제들
리팩토링과 설계
리팩토링과 성능
리팩토링의 유래

CHAPTER 03 코드의 구린내

중복 코드 Duplicated Code
장황한 메서드 Long Method
방대한 클래스 Large Class
과다한 매개변수 Long Parameter List
수정의 산발 Divergent Change
기능의 산재 Shotgun Surgery
잘못된 소속 Feature Envy
데이터 뭉치 Data Clumps
강박적 기본 타입 사용 Primitive Obsession
switch 문 Switch Statements
평행 상속 계층 Parallel Inheritance Hierarchies
직무유기 클래스 Lazy Class
막연한 범용 코드 Speculative Generality
임시 필드 Temporary Field
메시지 체인 Message Chains
과잉 중개 메서드 Middle Man
지나친 관여 Inappropriate Intimacy
인터페이스가 다른 대용 클래스 Alternative Classes with Different Interfaces
미흡한 라이브러리 클래스 Incomplete Library Class
데이터 클래스 Data Class
방치된 상속물 Refused Bequest
불필요한 주석 Comments

CHAPTER 04 테스트 작성

CHAPTER 05 리팩토링 기법 카탈로그에 대해

CHAPTER 06 메서드 정리

CHAPTER 07 객체 간의 기능 이동

CHAPTER 08 데이터 체계화

CHAPTER 09 조건문 간결화

CHAPTER 10 메서드 호출 단순화

CHAPTER 11 일반화 처리

CHAPTER 12 복합 리팩토링

CHAPTER 13 리팩토링, 재사용, 현실성

CHAPTER 14 리팩토링 도구

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 프
로그램 만들기

3부 객체지향 소프트웨어 설계
의 기본 원리

5장 소프트웨어의 변화와 포용

6장 변화를 수용하는 객체지향
설계의 원리

4부 GoF의 디자인패턴

7장 디자인패턴 소개

8장 GoF 디자인패턴 리스트

01 Abstract Factory

02 Factory Method

03 State & Strategie

04 Templet Method

05 Decorator

06 Composite

07 Bridge & Adapter

08 Mediator

09 Observer

10 Command

11 그밖의 유용한 패턴

9장 JHotdraw 프레임워크의
디자인패턴

제5부 리팩토링

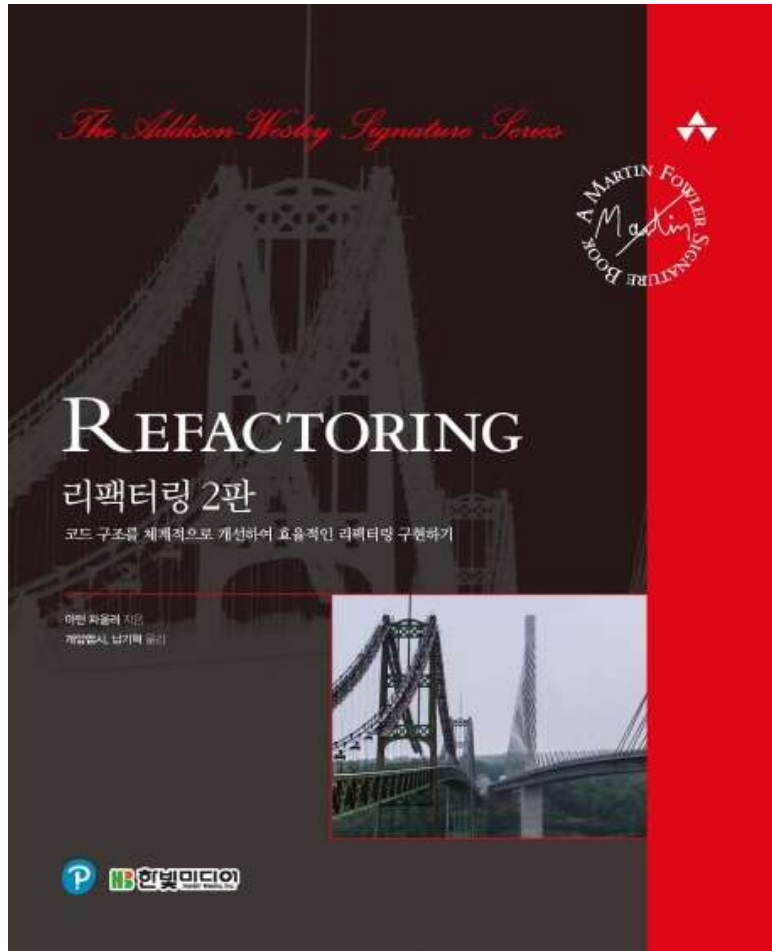
10장 리팩토링 소개

11장 리팩토링 카탈로그

12장 종합 대여 관리 시스템의
리팩토링 실습

부록 : 짝 프로그래밍을 이
용한 객체지향 설계 협력 학
습 방법

리팩터링 - 코드 구조를 체계적으로 개선하여 효율적인 리팩터링 구현하기 2판 (자바 스크립트, 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 포매팅 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

CHAPTER 03 코드에서 나는 악취

- 3.1 기이한 이름
- 3.2 중복 코드
- 3.3 긴 함수
- 3.4 긴 매개변수 목록
- 3.5 전역 데이터
- 3.6 가변 데이터
- 3.7 뒤엎힌 변경
- 3.8 산탄총 수술
- 3.9 기능 편애
- 3.10 데이터 뭉치
- 3.11 기본형 집착
- 3.12 반복되는 switch문
- 3.13 반복문
- 3.14 성의 없는 요소
- 3.15 추측성 일반화
- 3.16 임시 필드
- 3.17 메시지 체인
- 3.18 중개자
- 3.19 내부자 거래
- 3.20 거대한 클래스
- 3.21 서로 다른 인터페이스의 대안 클래스들
- 3.22 데이터 클래스
- 3.23 상속 포기
- 3.24 주석

CHAPTER 04 테스트 구축하기

- 4.1 자가 테스트 코드의 가치
- 4.2 테스트할 샘플 코드
- 4.3 첫 번째 테스트
- 4.4 테스트 추가하기
- 4.5 픽스처 수정하기
- 4.6 경계 조건 검사하기
- 4.7 끝나지 않은 여정

CHAPTER 05 리팩터링 카탈로그 보는 법

- 5.1 리팩터링 설명 형식
- 5.2 리팩터링 기법 선정 기준

CHAPTER 06 기본적인 리팩터링

- 6.1 함수 추출하기
- 6.2 함수 인라인하기
- 6.3 변수 추출하기
- 6.4 변수 인라인하기
- 6.5 함수 선언 바꾸기
- 6.6 변수 캡슐화하기
- 6.7 변수 이름 바꾸기
- 6.8 매개변수 객체 만들기
- 6.9 여러 함수를 클래스로 묶기
- 6.10 여러 함수를 변환 함수로 묶기
- 6.11 단계 쪼개기

CHAPTER 07 캡슐화

- 7.1 레코드 캡슐화하기
- 7.2 컬렉션 캡슐화하기
- 7.3 기본형을 객체로 바꾸기
- 7.4 임시 변수를 질의 함수로 바꾸기
- 7.5 클래스 추출하기
- 7.6 클래스 인라인하기
- 7.7 위임 숨기기
- 7.8 중개자 제거하기
- 7.9 알고리즘 교체하기

CHAPTER 08 기능 이동

- 8.1 함수 옮기기
- 8.2 필드 옮기기
- 8.3 문장을 함수로 옮기기
- 8.4 문장을 호출한 곳으로 옮기기
- 8.5 인라인 코드를 함수 호출로 바꾸기
- 8.6 문장 슬라이드하기
- 8.7 반복문 쪼개기
- 8.8 반복문을 파이프라인으로 바꾸기
- 8.9 죽은 코드 제거하기

CHAPTER 09 데이터 조직화

- 9.1 변수 쪼개기
- 9.2 필드 이름 바꾸기
- 9.3 파생 변수를 질의 함수로 바꾸기
- 9.4 참조를 값으로 바꾸기
- 9.5 값을 참조로 바꾸기
- 9.6 매직 리터럴 바꾸기

CHAPTER 10 조건부 로직 간소화

- 10.1 조건문 분해하기
- 10.2 조건식 통합하기
- 10.3 중첩 조건문을 보호 구문으로 바꾸기
- 10.4 조건부 로직을 다형성으로 바꾸기
- 10.5 특이 케이스 추가하기
- 10.6 어서션 추가하기
- 10.7 제어 플래그를 탈출문으로 바꾸기

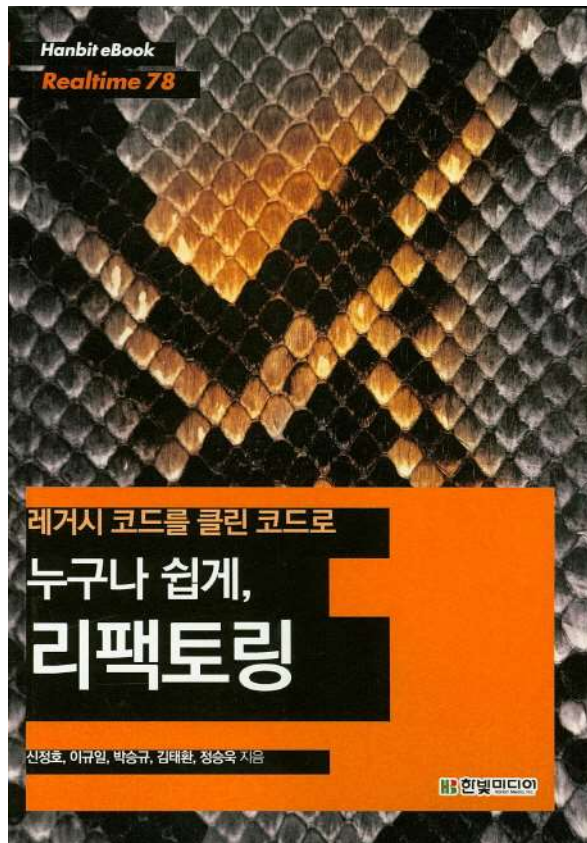
CHAPTER 11 API 리팩터링

- 11.1 질의 함수와 변경 함수 분리하기
- 11.2 함수 매개변수화하기
- 11.3 플래그 인수 제거하기
- 11.4 객체 통째로 넘기기
- 11.5 매개변수를 질의 함수로 바꾸기
- 11.6 질의 함수를 매개변수로 바꾸기
- 11.7 세터 제거하기
- 11.8 생성자를 팩터리 함수로 바꾸기
- 11.9 함수를 명령으로 바꾸기
- 11.10 명령을 함수로 바꾸기
- 11.11 수정된 값 반환하기
- 11.12 오류 코드를 예외로 바꾸기
- 11.13 예외를 사전확인으로 바꾸기

CHAPTER 12 상속 다루기

- 12.1 메서드 올리기
- 12.2 필드 올리기
- 12.3 생성자 본문 올리기
- 12.4 메서드 내리기
- 12.5 필드 내리기
- 12.6 타입 코드를 서브클래스로 바꾸기
- 12.7 서브클래스 제거하기
- 12.8 슈퍼클래스 추출하기
- 12.9 계층 합치기
- 12.10 서브클래스를 위임으로 바꾸기
- 12.11 슈퍼클래스를 위임으로 바꾸기

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 메서드 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)



1장 기술 부채

- 1.1 기술 부채의 개념
- 1.2 기술 부채의 구성 요소
- 1.3 기술 부채가 미치는 영향
- 1.4 기술 부채를 초래하는 원인
- 1.5 기술 부채를 관리하는 방법

2장 설계 악취

- 2.1 악취에 신경 써야 하는 이유
- 2.2 악취를 일으키는 요인
 - 2.2.1 설계 원칙 위반
 - 2.2.2 부적절한 패턴 사용
 - 2.2.3 언어 제약
 - 2.2.4 객체 지향에서 절차적인 사고방식
 - 2.2.5 점성
 - 2.2.6 우수 관례와 우수 프로세스의 미준수
- 2.3 악취를 해소하는 방법
- 2.4 책에서 다루는 악취 범위
- 2.5 설계 악취의 분류
 - 2.5.1 악취 분류를 기반으로 한 설계 원칙
 - 2.5.2 악취를 명명하는 방식
 - 2.5.3 악취를 문서화하는 템플릿

3장 추상화 악취

4장 캡슐화 악취

5장 모듈화 악취

6장 계층 악취

7장 악취 생태계

8장 실전 기술 부채 상
환

부록 A 소프트웨어 설계 원칙

- A.1 계층 원칙
- A.2 다양한 캡슐화 원칙
- A.3 단일 책임 원칙
- A.4 모듈화 원칙
- A.5 비순환 의존성 원칙(ADP)
- A.6 정보 은닉 원칙
- A.7 추상화 원칙
- A.8 캡슐화 원칙
- A.9 DRY 원칙
- A.10 KISS 원칙
- A.11 LSP 원칙
- A.12 OCP 원칙

부록 B 기술 부채를 상환하는 도구

부록 C 그림 표기법

부록 D 추천 도서

- D.1 핵심 도서
- D.2 리팩토링과 리엔지니어링
- D.3 패턴과 안티패턴
- D.4 기술 부채

부록 E 참고문헌

בב
ע

