

<1 과목 소프트웨어 설계>

1. 꼭 알아야 할 키워드 = ____ (맞출)

2. # = 두음 암기 or 한 칸 띄어 쓴 건 산출물

3. 시나공 + 수제비 정리 (페이지 참고)

4. "Ctrl+F" 탐색 → 제목 활용하기

[1] 소프트웨어 생명 주기, SDLC(Software Development Life Cycle) ★★

p.22, 5-3, 20 년 1, 2, 3, 4 회 기출문제

1) 폭포수 모형(Waterfall Model) ★

- 가장 오래되고 가장 폭넓게 사용된 고전적 생명 주기 모형
- 한 단계가 끝나야만 다음 단계로 넘어가는 선형 순차적 모형
- 단계별 정의 및 산출물이 명확
- 개발 중간에 요구사항의 변경이 용이하지 않음
- 타당성검토 → 계획 → 요구 분석 → 설계 → 구현(코딩) → 테스트(검사) → 유지보수

#분설구테유

© 2021. 함께 공부해요 All rights reserved.

2) 프로토타입 모형(Prototype Model, 원형 모형) ★

- 견본(시제)품을 만들어 최종 결과물을 예측하는 모형
- 인터페이스 중점을 두어 개발
- 개발 중간에 요구사항의 변경이 용이

3) 나선형 모형(Spiral Model, 점진적 모형) ★ _ 20 년 1, 2, 3 회 기출문제

- 폭포수 모형과 프로토타입 모형의 장점에 위험 분석 기능을 추가한 모형
- 점진적 개발 과정 반복으로 요구사항 추가 가능
- 정밀하고 유지보수 과정 필요 없음
- 계획 및 정의 → 위험 분석 → 공학적 개발 → 고객 평가

#계위개고

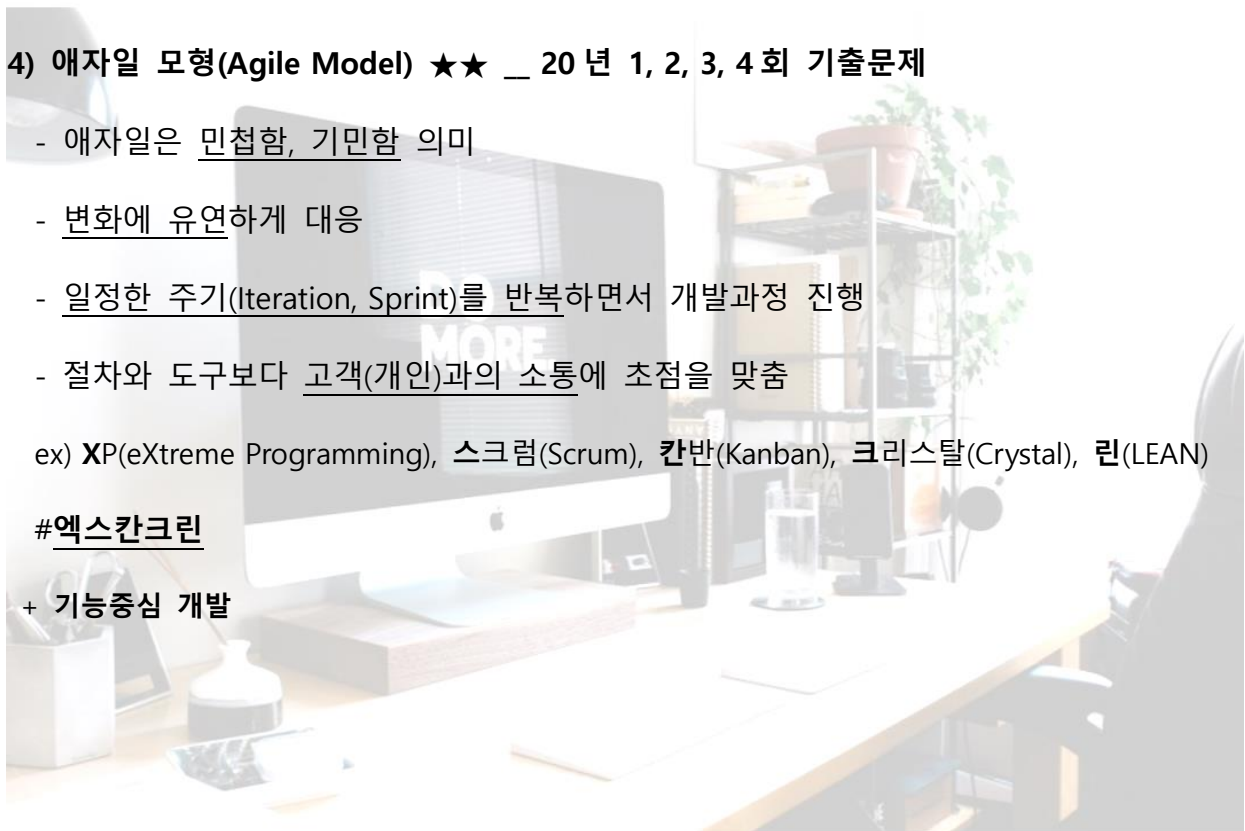
4) 애자일 모형(Agile Model) ★★ _ 20 년 1, 2, 3, 4 회 기출문제

- 애자일은 민첩함, 기민함 의미
- 변화에 유연하게 대응
- 일정한 주기(Iteration, Sprint)를 반복하면서 개발과정 진행
- 절차와 도구보다 고객(개인)과의 소통에 초점을 맞춤

ex) XP(eXtreme Programming), 스크럼(Scrum), 칸반(Kanban), 크리스탈(Crystal), 린(LEAN)

#엑스칸크린

+ 기능중심 개발



[2] 스크럼(Scrum) 기법 ★

p.26, 1-14

- 팀원 스스로가 스크럼 팀 구성
- 개발 작업에 관한 모든 것을 스스로 해결해야 함
- 스프린트는 2 ~ 4 주 정도의 기간으로 진행

1) 제품 책임자(PO; Product Owner) ★

- 요구사항이 담긴 백로그(Backlog)를 작성하는 주체
- 백로그에 대한 우선순위를 지정, 이해관계자들의 의견을 종합

2) 스크럼 마스터(SM; Scrum Master)

- 일일 스크럼 회의 주관
- 팀원들을 통제하는 것이 목표가 아님

3) 개발팀(DT; Development Team)

- 제품 책임자와 스크럼 마스터를 제외한 모든 팀원
- 최대 인원 7~8 명

4) 스크럼 개발 프로세스

- 스프린트 계획 회의 → 스프린트 → 일일 스크럼 → 스크럼 검토 회의 → 스프린트 회고

#계스일검회

3 XP 기법 ★★

p.31, 1-14

1) XP(eXtreme Programming)의 핵심 가치 ★

- 용기(Courage), 단순성(Simplicity), 의사소통(Communication), 피드백(Feedback), 존중(Respect)

#용단의피존

2) XP의 기본원리 _ 20년 4회 기출문제

- Whole Team(전체 팀), Small Releases(소규모 릴리즈)
Test-Driven Development(테스트 주도 개발), Continuous Integration(계속적인 통합)
Collective Ownership(공동 소유권), Pair Programming(짝 프로그래밍)
Design Improvement(디자인 개선) 또는 Refactoring(리팩토링)

#전소테 계공짜디

4 개발 기술 환경 파악 ★

p.37, 1-3

1) 운영체제(OS; Operating System)

- 하드웨어가 아닌 소프트웨어

Windows, UNIX, Linux, Mac OS | iOS, Android 등등

- 가용성, 성능 | 기술 지원, 구축 비용, 주변 기기 (고려사항)

#가성기구주

2) 미들웨어(Middleware)

- 운영체제와 응용 프로그램 사이에서 추가적인 서비스를 제공하는 소프트웨어

3) 데이터베이스 관리 시스템(DBMS; Database Management System)

- 사용자와 데이터베이스(DB) 사이에서 정보를 생성하고 DB 를 관리하는 소프트웨어
- 데이터베이스(DB)의 구성, 접근 방법, 유지관리에 대한 모든 책임을 짐
- JDBC(Java Database Connectivity, 자바), ODBC(Open Database Connectivity, 응용 프로그램)
- Oracle, MySQL, SQLite, MongoDB, Redis 등등
- **가용성, 성능 | 기술 지원, 구축 비용, 상호 호환성 (고려사항) ★ _ 1, 2 회 기출문제**

#가성기구호

4) 웹 어플리케이션 서버(WAS; Web Application Server) ★

- 정적인 콘텐츠를 처리하는 웹 서버(Web Server)와 반대됨
- 동적인 콘텐츠를 처리하기 위해 사용되는 미들웨어(=소프트웨어)
- 데이터 접근, 세션 관리, 트랜잭션 관리 등을 위한 라이브러리를 제공
- Tomcat, JEUS, WebLogic, JBoss, Jetty, Resin 등등
- **가용성, 성능 | 기술 지원, 구축 비용 (고려사항)**

#가성기구

5) 오픈 소스(Open Source)

- 누구나 별다른 제한 없이 사용할 수 있도록 소스 코드를 무료로 사용할 수 있게 공개한 것
- 라이선스의 종류, 사용자 수, 기술의 지속 가능성 (고려사항)

#라사지 © 2021. 함께 공부해요 All rights reserved.

5) 요구사항 정의 ★

p.41, 1-11

1) 기능 요구사항

- 기능, 입력, 출력, 저장, 수행 등등

2) 비기능 요구사항

- 성능, 품질, 제약사항, 호환성, 보안 등등

3) 요구사항 개발 프로세스 ★ _ 5-5

- 도출(Elicitation)/추출 → 분석(Analysis) → 명세(Specification) → 확인(Validation)/검증(Valification)

#도분명확 #추분명검

4) 요구사항 분석 기법 ★

- 요구사항 분류, 개념 모델링(UML), 요구사항 할당, 요구사항 협상, 정형 분석

#분개할협정

5) 요구사항 확인 기법 ★★ _ 20 년 1, 2, 3 회 기출문제

- 요구사항 검토, 프로토타이핑, 모델 검증, 인수 테스트(알파 테스트, 베타 테스트)

#검프모인

© 2021. 함께 공부해요 All rights reserved.

[6] UML ★★★

p.51, 20 년 1, 2, 3 회 기출문제

1) UML(Unified Modeling Language)의 구성 요소 ★

- 사물, 관계, 다이어그램

#사관다

2) 사물(Things)

- 구조, 행동, 그룹, 주해 {사물}

#구행그주

3) 관계(Relationships) ★★ _ 20 년 3 회 기출문제

- 연관(—), 집합(◇), 포함(◆), 일반화(—▷), 의존(-->), 실체화(--▷) {관계}

#연집포 일의실

4) 구조적, 정적 다이어그램(Diagram) ★★ _ 20 년 1, 2, 3 회 기출문제

- 클래스(Class), 객체(Object), 컴포넌트(Component), 배치(Deployment),
복합체 구조(Composite Structure), 패키지(Package) {다이어그램(Diagram)}
- 컴포넌트, 배치 다이어그램은 구현 단계에서 사용되는 다이어그램임 ★

#클객컴 배복패

5) 행위, 동적 다이어그램(Diagram) ★★ _ 20 년 1, 2, 3 회 기출문제

- 유스케이스(Use Case, 사용사례), 시퀀스(Sequence, 순차),
커뮤니케이션(Communication, 협업), 상태(State), 활동(Activity),
상호작용 개요(Interaction Overview), 타이밍(Timing) {다이어그램(Diagram)}

#유시커 상황호타

7 사용자 인터페이스(UI; User Interface) ★

p. 62, 1-23

1) UI의 구분 ★

- CLI(Command Line Interface): 텍스트 형태로 이뤄진 인터페이스
- GUI(Graphical User Interface): 마우스로 선택해 작업을 하는 그래픽 환경의 인터페이스
- NUI(Natural User Interface): 사용자의 말이나 행동으로 기기를 조작하는 인터페이스
- VUI(Voice User Interface): 사람의 음성으로 기기를 조작하는 인터페이스
- OUI(Organic User Interface): 모든 사물과 사용자 간의 상호작용을 위한 인터페이스

2) UI의 기본 원칙 ★★ _ 20년 1, 2회 기출문제

- 직관성: 누구나 쉽게 이해하고 사용할 수 있어야 함
- 유효성: 사용자의 목적을 정확하고 완벽하게 달성해야 함
- 학습성: 누구나 쉽게 배우고 익힐 수 있어야 함
- 유연성: 사용자의 요구사항을 최대한 수용하고 실수를 최소화해야 함

#직유학연

3) 웹의 3 요소

- 웹 표준(Web Standards), 웹 접근성(Web Accessibility), 웹 호환성(Cross Browsing)

#표점호

© 2021. 함께 공부해요 All rights reserved.

4) UI 설계 도구 ★

- 와이어프레임(Wireframe): 레이아웃을 협의하거나 공유하기 위해 사용
- 스토리보드(Story Board): 최종적으로 참고하는 작업 지침서, 작업 산출물 (디스크립션)
- 프로토타입(Prototype): 인터랙션을 적용해 실제 구현된 것처럼 테스트가 가능한 동적인 모형
- 목업(Mockup): 실제 화면과 유사한 정적인 모형
- 유스케이스(Use Case): 사용자 측면 요구사항을 다이어그램 형식으로 묘사 (유스케이스 명세서)

#와스프목유

5) UI 프로토타입

- ▶ **장점:** 사용자를 설득하고 이해시키기 쉬움 | 개발 시간을 줄일 수 있음 | 사전 오류 발견 가능
- ▶ **단점:** 반복적인 개선 및 보완 작업으로 인한 작업 시간 증가 및 자원 소모 | 부분적인 프로토타이핑으로 인한 중요한 작업 생략 가능성

페이퍼 프로토타입, 디지털 프로토타입, HTML/CSS

6) UI 시나리오 문서 요건

- 이해성(Understandable): 누구나 쉽게 이해할 수 있도록 설명
- 완전성(Complete): 최대한 상세하게 기술
- 일관성(Consistent): 일관성 유지
- 가독성(Readable): 표준화된 템플릿 등을 활용하여 문서를 쉽게 읽을 수 있도록 해야함
- 수정 용이성(Modifiable): 수정 및 개선이 쉬워야 함
- 추적 용이성(Traceable): 변경사항에 대해서 쉽게 추적할 수 있어야 함

#이완일 가수추

7) 기타

- **HCI(Human Computer Interaction or Interface)**: {사람}과 {컴퓨터}의 {상호작용}을 연구해서 사람이 컴퓨터를 편리하게 사용하도록 만드는 학문
- **UX(User Experience)**: 사용자가 시스템이나 서비스를 이용하면서 느끼고 생각하는 총체적인 경험
주관성(Subjecctivity), 정황성(Contextuality), 총체성(Holistic)
- 감성공학: 1 류; 인간의 감성 / 2 류; 심리적 기능 / 3 류; 공학적 및 수학적 모델, 객관적

8) 품질 요구사항 ★

p.76, 2-60, 20 년 3 회 기출문제

1) 국제 제품 품질 표준 (수제비) ★

- ISO/IEC 9126
- ISO/IEC 12119
- ISO/IEC 14598
- ISO/IEC 25000: SW 품질 평가 통합 모델, SQaRE 로도 불리며 위 3 개 표준을 통합 품질 관리(2500n), 품질 모델(2501n), 품질 측정(2502n), 품질 요구(2503n), 품질 평가(2504n)
#관모측요평

2) ISO/IEC 9126 ★★ — 20 년 1, 2, 3 회 기출문제

- 기능성(Functionality): 요구사항을 정확하게 만족하는 기능을 제공하는가?
적절성(적합성), 정확성, 상호 운용성, 보안성, 호환성
- 신뢰성(Reliability): 요구된 기능을 정확하고 일관되게 오류 없이 수행하는가?
성숙성, 결함 허용성, 회복성

- **사용성(Usability)**: 사용자가 정확하게 이해하고 사용하는가?
이해성, 학습성, 운용성, 친밀성
- **효율성(Efficiency)**: 할당된 시간 동안 한정된 자원으로 얼마나 빨리 처리하는가?
시간 효율성, 자원 효율성
- **유지 보수성(Maintainability)**: 환경의 변화에 소프트웨어를 쉽게 개선, 확장, 수정할 수 있는가?
분석성, 변경성, 안정성, 시험성
- **이식성(Portability)**: 소프트웨어를 다른 환경에서도 쉽게 적용할 수 있는가?
적용성, 설치성, 대체성, 공존성

#기신사 효유이

3) ISO/IEC 14598 (수제비)

- **반복성(Repeatability)**, **재현성(Reproducibility)**, **공정성(Impartiality)**, **객관성(Objectivity)**

#반재공객

4) 국제 프로세스 품질 표준

- ISO/IEC **9001**
- ISO/IEC **12207**: 기본 프로세스, 조직 프로세스, 지원 프로세스

#기조지

- ISO/IEC **15504(SPICE)**: 불완전 → 수행 → 관리 → 확립 → 예측 → 최적화

#불수관 확예최

- **CMMI**(Capability Maturity Model Integration): 조직차원의 성숙도를 평가하는 단계별 표현과 프로세스 영역별 능력도를 평가하는 연속적 표현이 있음

©2021 한계공학 All rights reserved.

9] 소프트웨어 아키텍처 ★

p.98

- 사용자의 비기능적 요구사항으로 나타난 제약 반영
- 기능적 요구사항을 구현하는 방법을 찾는 해결 과정

#모추단정

1) 모듈화(Modularity)

- 시스템 기능들을 모듈 단위로 나눠 소프트웨어의 성능 및 재사용성을 향상시키는 것
- 모듈의 크기 多: 모듈 개수 적음 | 모듈 간 통합 비용 적음 | 모듈 하나의 개발 비용 큼
- 모듈의 크기 小: 모듈 개수 많음 | 모듈 간 통합 비용 큼

2) 추상화(Abstraction)

- 전체적이고 포괄적인 개념을 설계한 후 차례로 세분화하여 구체화 시키는 것
- 과정 추상화: 자세한 수행 과정을 정의하지 않고, 전반적인 흐름만 파악
- 데이터 추상화: 데이터의 세부적인 속성이나 용도를 정의하지 않고, 데이터 구조를 대표하는 표현으로 대체
- 제어 추상화: 이벤트 발생의 정확한 절차나 방법을 정의하지 않고, 대표하는 표현으로 대체

#과데제

3) 단계적 분해(Stepwise Refinement)

- Niklaus Wirth 에 의해 제안된 하향식 설계 전략
- 추상화의 반복에 의해 세분화
- 소프트웨어 기능에서부터 시작해 점차적으로 구체화
- 상세한 내역은 가능한 한 뒤로 미루어 진행

4) 정보 은닉(Information Hiding)

- 한 모듈 내부에 포함된 절차와 자료들의 정보가 감추어져 다른 모듈이 접근하거나 변경하지 못하도록 하는 기법
- 정보 은닉을 통한 독립적 모듈 수행 가능
- 모듈 변경 시 영향을 받지 않아 수정, 시험, 유지보수 용이

10 아키텍처 패턴 ★

p.102

1) 레이어 패턴(Layers Pattern)

- 시스템을 계층(Layer)으로 구분하여 구성하는 고전적 방법
- # OSI 참조 모델 ★

2) 클라이언트-서버 패턴(Client-Server Pattern)

- 하나의 서버 컴포넌트와 다수 클라이언트 컴포넌트로 구성되는 패턴
 - 클라이언트나 서버는 요청과 응답을 받기 위해 동기화 되는 경우를 제외하고는 서로 독립적
- * 컴포넌트(Component): 독립적인 업무 또는 기능을 수행하는 실행코드 기반으로 작성된 모듈

3) 파이프-필터 패턴(Pipe-Filter Pattern) ★

- 데이터 스트림 절차의 각 단계를 필터 컴포넌트로 캡슐화해 파이프를 통해 데이터를 전송하는 패턴
- 필터 컴포넌트는 재사용성이 좋고, 추가가 쉬워 확장 용이
- 필터 컴포넌트들을 재배치하여 다양한 파이프라인 구축 가능

UNIX의 셸(Shell)

4) 모델-뷰-컨트롤러 패턴(Model-View-Controller Pattern) ★★

- 서브시스템을 3 개의 부분으로 구조화하는 패턴
- 모델(**M**odel): 서브시스템의 핵심 기능과 데이터를 보관
- 뷰(**V**iew): 사용자에게 정보를 표시
- 컨트롤러(**C**ontroller): 사용자로부터 받은 입력 처리 / 뷰 제어 / UI 담당
- 각 부분은 별도의 컴포넌트로 분리되어 있으므로 서로 영향을 받지 않고 개발 작업 수행
- 한 개의 모델에 대해 여러 개의 뷰를 만들 수 있으므로 대화형 애플리케이션에 적합

5) 마스터-슬레이브 패턴(Master-Slave Pattern)

- 마스터 컴포넌트에서 슬레이브 컴포넌트로 분할한 후, 슬레이브 컴포넌트에서 처리된 결과물을 다시 돌려받는 방식으로 작업을 수행하는 패턴

장애 허용 시스템(Fault Tolerance System), 병렬 컴퓨팅 시스템 ★

6) 브로커 패턴(Broker Pattern)

- 컴포넌트와 사용자를 연결해주는 패턴

분산 환경 시스템

7) 피어-투-피어 패턴(Peer-To-Peer Pattern)

- 피어를 하나의 컴포넌트로 간주하며, 각 피어는 서비스를 호출하는 클라이언트가 될 수도, 서비스를 제공하는 서버가 될 수도 있는 패턴

멀티스레딩(Multi Threading) 방식 사용



© 2021. 함께 공부해요 All rights reserved.

8) 이벤트-버스 패턴(Event-Bus Pattern)

- 소스가 특정 채널에 이벤트 메시지를 발행하면, 해당 채널을 구독한 리스너들이 메시지를 받아 이벤트를 처리하는 방식
- 이벤트를 생성하는 소스(Source), 이벤트를 수행하는 리스너(Listener), 이벤트의 통로인 채널(Channel), 채널들을 관리하는 버스(Bus)

#소리채버

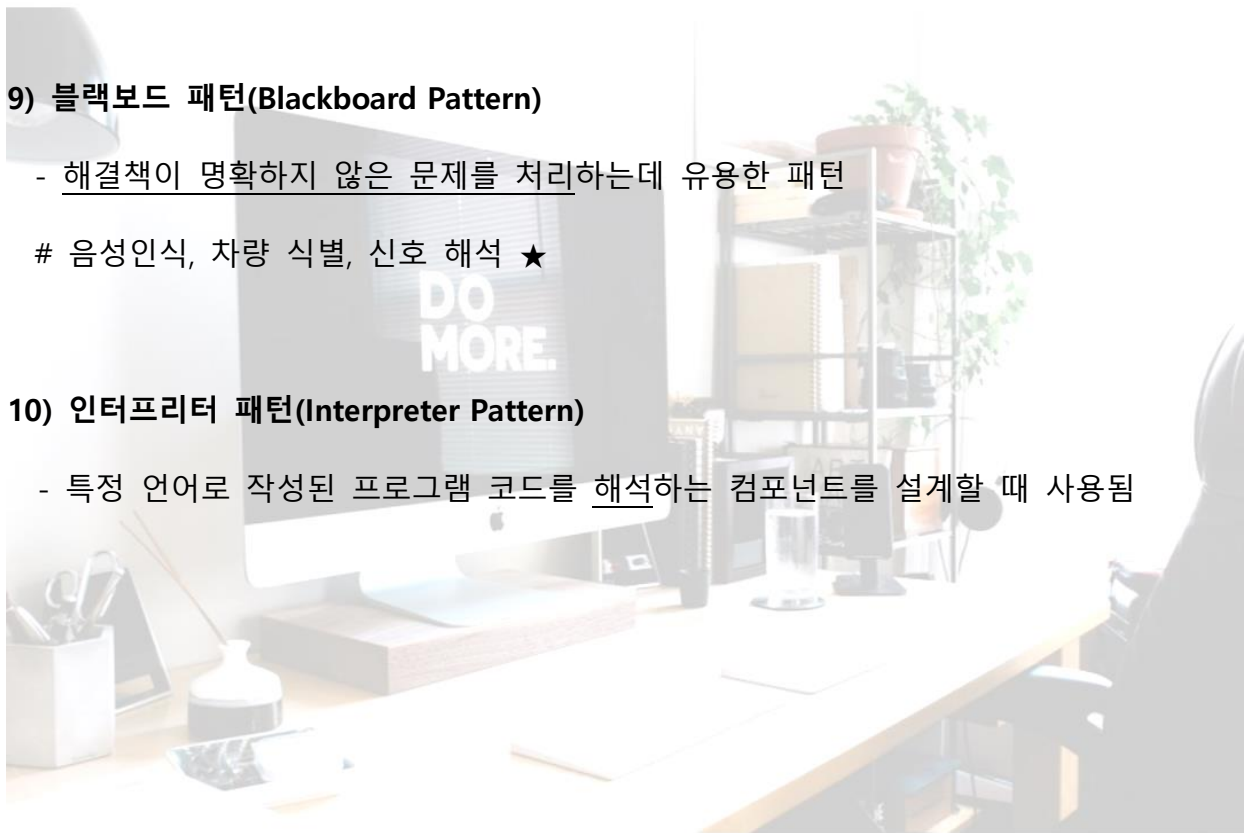
9) 블랙보드 패턴(Blackboard Pattern)

- 해결책이 명확하지 않은 문제를 처리하는데 유용한 패턴

음성인식, 차량 식별, 신호 해석 ★

10) 인터프리터 패턴(Interpreter Pattern)

- 특정 언어로 작성된 프로그램 코드를 해석하는 컴포넌트를 설계할 때 사용됨



11 객체지향(Object-Oriented) ★★

p.106, 1-58

1) 객체(Object)

- 독립적으로 식별 가능한 이름을 갖고 있음
- 객체가 가질 수 있는 조건인 상태(State)는 일반적으로 시간에 따라 변함
- 객체와 객체는 상호 연관성에 의한 관계가 형성됨
- 객체가 반응할 수 있는 메시지의 집합을 행위(연산, Method)라고 하며, 객체는 행위의 특징을 나타냄
- 객체는 일정한 기억장소를 갖고 있음

2) 클래스(Class) ★★ _ 20 년 1, 2, 3 회 기출문제

- 하나 이상의 유사한 객체들을 묶어서 하나의 공통된 특성을 표현한 것 ★
- 공통된 속성과 연산(행위)를 갖는 객체의 집합
- 객체지향 프로그램에서 데이터를 추상화하는 단위 ★
- 각각의 객체들이 갖는 속성과 연산(Method)을 정의하고 있는 틀
- 슈퍼 클래스(Super Class)는 특정 클래스의 상위(부모) 클래스
- 서브 클래스(Sub Class)는 특정 클래스의 하위(자식) 클래스

3) 인스턴스(Instance)

- 클래스에 속한 각각의 객체
- 클래스로부터 새로운 객체를 생성하는 것을 인스턴스화(Instantiation)라고 함

4) 메서드(Method)

- 클래스로부터 생성된 객체를 사용하는 방법
- 전통적 시스템의 함수(Function) 또는 프로시저(Procedure)에 해당하는 연산

5) 메시지(Message)

- 객체에게 어떤 행위를 하도록 지시하기 위한 방법

#캡상다형 ★

6) 캡슐화(encapsulation) _ 20 년 3 회 기출문제

- 데이터(속성)와 데이터를 처리하는 함수를 하나로 묶는 것
- 인터페이스를 제외한 세부 내용이 은폐(정보 은닉)되어 외부 접근이 제한됨
- 정보 은닉 측면과 가장 밀접한 관계가 있음
- 외부 모듈의 변경으로 인한 파급 효과가 적음
- 재사용 용이, 인터페이스 단순해짐
- 결합도 Down / 응집도 Up

7) 상속(Inheritance)

- 이미 정의된 상위(부모) 클래스의 모든 속성과 연산을 하위(자식) 클래스가 물려받는 것
- 소프트웨어의 재사용(Reuse)을 높이는 중요한 개념

8) 다중 상속(Multiple Inheritance)

- 한 개의 클래스가 두 개 이상의 상위(부모) 클래스로부터 속성과 연산을 상속받는 것

9) 다형성(Polymorphism)

- 하나의 메시지에 대해 각각의 객체(클래스)가 가지고 있는 고유한 방법(특성)으로 응답할 수 있는 능력

ex) '+' 연산자의 경우 숫자 클래스에서는 덧셈, 문자 클래스에서는 문자열의 연결 기능

12 결합도(Coupling) ★★

p.110, 1-49, 20 년 1, 2, 3 회 기출문제

- 모듈 간에 상호 의존하는 정도 또는 두 모듈 사이의 연관 관계를 의미
- 결합도는 낮을수록(↓) Good = 독립적인 모듈

#내공외제스자 (Bad → Good) ★★

1) 내용 결합도(Content Coupling)

- 한 모듈이 다른 모듈의 내부 기능 및 그 내부 자료를 직접 참조하거나 수정할 때의 결합도

2) 공통 결합도(Common Coupling)

- 공유되는 공통 데이터 영역을 여러 모듈이 사용할 때의 결합도 (전역 변수)

3) 외부 결합도(External Coupling)

- 어떤 모듈에서 선언한 데이터(변수)를 외부의 다른 모듈에서 참조할 때의 결합도 (순차적)

4) 제어 결합도(Control Coupling)

- 어떤 모듈이 다른 모듈 내부의 논리적인 흐름을 제어하기 위해 제어 신호를 이용하여 통신하거나 제어 요소를 전달하는 결합도

5) 스탬프 결합도(Stamp Coupling)

- 모듈 간의 인터페이스로 배열이나 레코드 등의 자료 구조가 전달될 때의 결합도

6) 자료 결합도(Data Coupling)

- 어떤 모듈이 다른 모듈을 호출하면서 매개 변수(파라미터)나 인수로 데이터를 넘겨주고, 호출 받은 모듈은 받은 데이터에 대한 처리 결과를 다시 돌려주는 결합도

© 2021. All rights reserved.

13 응집도(Cohesion) ★★

p.111, 1-48, 20 년 1, 2, 3 회 기출문제

- 모듈의 내부 요소들의 서로 관련되어 있는 정도
- 응집도는 높을수록(↑) Good = 독립적인 모듈

#우논시절통순기 (Bad → Good) ★★

1) 우연적 응집도(Coincidental Cohesion)

- 모듈 내부의 각 구성 요소들이 서로 관련 없는 요소로만 구성된 경우의 응집도

2) 논리적 응집도(Logical Cohesion)

- 유사한 성격을 갖거나 특정 형태로 분류되는 처리 요소들로 하나의 모듈이 형성되는 경우의 응집도

3) 시간적 응집도(Temporal Cohesion)

- 특정 시간에 처리되는 몇 개의 기능을 모아 하나의 모듈로 작성할 경우의 응집도

4) 절차적 응집도(Procedural Cohesion)

- 모듈이 다수의 관련 기능을 가질 때 모듈 안의 구성 요소들이 그 기능을 순차적으로 수행할 경우의 응집도

5) 통신적(교환적) 응집도(Communication Cohesion)

- 동일한 입력과 출력을 사용하여 서로 다른 기능을 수행하는 구성 요소들이 모였을 경우의 응집도

6) 순차적 응집도(Sequential Cohesion)

- 모듈 내 하나의 활동으로부터 나온 출력 데이터(출력값)를 그 다음 활동의 입력 데이터로 사용할 경우의 응집도

7) 기능적 응집도(Functional Cohesion)

- 모듈 내부의 모든 기능 요소들이 단일 문제와 연관되어 수행될 경우의 응집도

14 공통 모듈 ★

p.114, 1-47, 20 년 1, 2 회 기출문제

#정명완일추

1) 정확성(Correctness)

- 시스템 구현 시 해당 기능이 필요하다는 것을 알 수 있도록 정확히 작성

2) 명확성(Clarity) ★

- 해당 기능에 대해 일관되게 이해되고, 한 가지로 해석될 수 있도록 즉, 중의적으로 해석되지 않도록 명확하게 작성

3) 완전성(Completeness)

- 시스템 구현을 위해 필요한 모든 것을 기술

4) 일관성(Consistency)

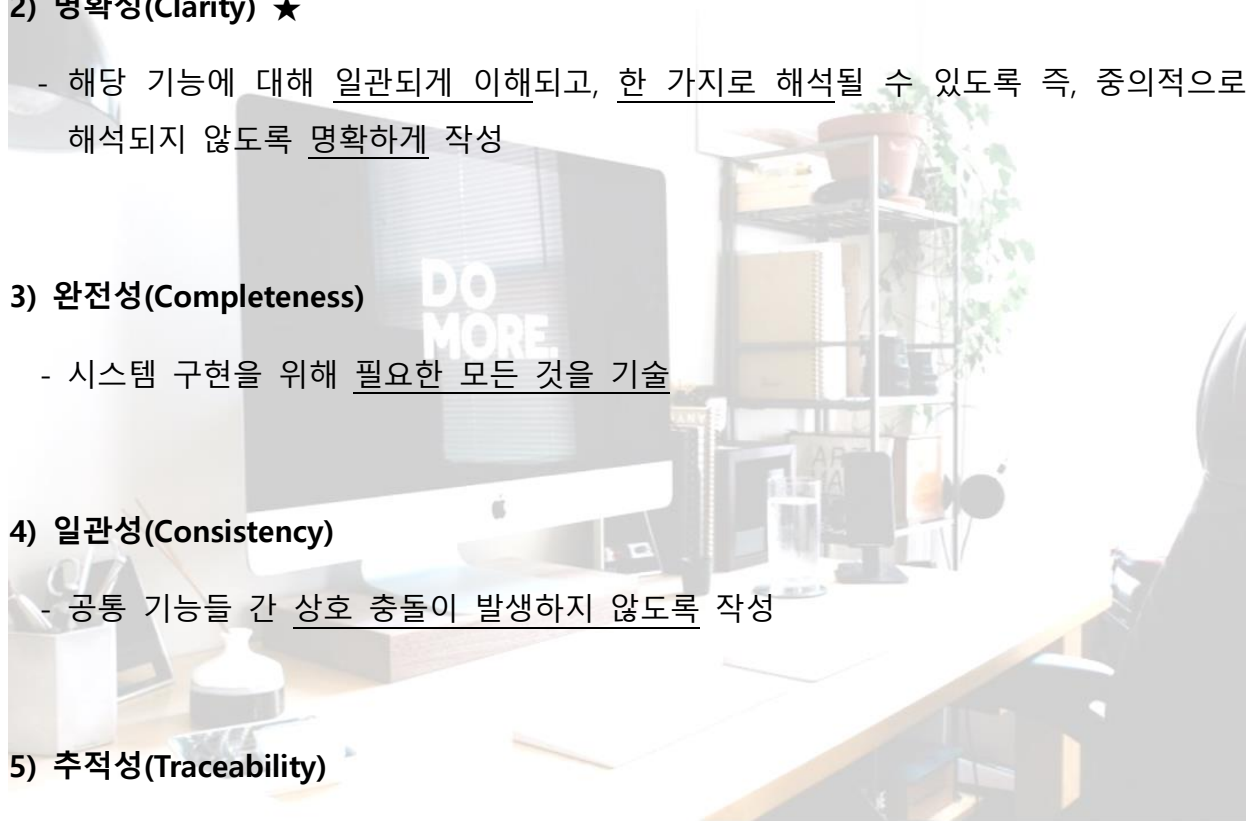
- 공통 기능들 간 상호 충돌이 발생하지 않도록 작성

5) 추적성(Traceability)

- 기능에 대한 요구사항의 출처, 관련 시스템 등의 관계를 파악할 수 있도록 작성

6) 재사용(Reuse) 규모에 따른 분류 _ 20 년 4 회 기출문제

- 함수와 객체, 컴포넌트, 애플리케이션



© 2021 함께 공부해요 All rights reserved.

15 코드 ★★

p.116, 20 년 1, 2, 3 회 기출문제

- 식별, 분류, 배열, 간소화, 표준화, 연상, 암호화, 오류 검출 {기능}

#식분배간 표연암오

1) 순차(순서) 코드(Sequence Code, 일련 번호 코드) ★

- 일정 기준에 따라서 최초의 자료부터 차례로 일련번호를 부여하는 방법

ex) 1, 2, 3, 4, ...

2) 블록 코드(Block Code, 구분 코드) ★

- 공통성이 있는 것끼리 블록으로 구분하고, 각 블록 내에서 일련번호를 부여하는 방법

ex) 1001~1100: 총무부, 1101~1200: 영업부

3) 10 진 코드(Decimal Code, 도서 분류식 코드) ★

- 0~9 까지 10 진 분할하고, 다시 각각에 대해 10 진 분할하는 방법을 필요한 만큼 반복하는 방법

ex) 1000: 공학, 1100: 소프트웨어 공학, 1110: 소프트웨어 설계

4) 그룹 분류 코드(Group Classification Code)

- 일정 기준에 따라 대분류, 중분류, 소분류 등으로 구분하고, 각 그룹 안에서 일련번호를 부여하는 방법

ex) 1-01-001: 본사-총무부-인사계, 2-01-001: 지사-총무부-인사계

5) 연상 코드(Mnemonic Code, 기호 코드)

- 명칭이나 약호와 관계있는 숫자나 문자, 기호를 이용하여 코드를 부여하는 방법

ex) TV-40: 40 인치 TV, L-15-220: 15W 220V 램프

6) 표의 숫자 코드(Significant Digit Code, 유효 숫자 코드) ★

- 길이, 넓이, 부피, 지름, 높이 등의 물리적 수치를 그대로 코드에 적용시키는 방법

ex) 120-720-1500: 두께 X 폭 X 길이가 120X720X1500 인 강판

7) 합성 코드(Combined Code)

- 2 개 이상의 코드를 조합하여 만드는 방법

ex) 연상 코드+순차 코드 → KE-711: 대한항공 711 기, AC-253: 에어캐나다 253 기

8) 코드 부여 체계

- 이름만으로 개체의 용도와 적용 범위를 알 수 있도록 코드를 부여하는 방식
- 각 개체에 유일한 코드 부여하여 개체들의 식별 및 추출을 용이하게 함
- 코드를 부여하기 전 각 단위 시스템의 고유한 코드와 개체를 나타내는 코드가 정의되어야 함

ex) PJC-COM-003: 전체 시스템 단위의 3 번째 공통 모듈

ex) PY3-MOD-010: PY3 라는 단위 시스템의 10 번째 모듈

16 디자인 패턴 ★★

p.119, 1-60, 20 년 1, 2, 3 회 기출문제

- 아키텍처 패턴이 디자인 패턴보다 상위 수준의 설계에 사용됨
- 서브시스템에 속하는 컴포넌트들과 그 관계를 설계하기 위한 참조 모델
- cf) 아키텍처 패턴은 전체 시스템의 구조를 설계하기 위한 참조 모델

#생구행

1) 생성 패턴(Creational Pattern) _ 20 년 3 회 기출문제 ★

- 추상 팩토리(Abstract Factory): 서로 연관, 의존하는 객체들을 그룹으로 생성해 추상적으로 표현
- 빌더(Builder): 객체의 생성 과정과 표현 방법 분리 → 동일한 객체 생성에도 서로 다른 결과
- 팩토리 메소드(Factory Method): 객체를 생성하기 위한 인터페이스를 정의하여, 어떤 클래스가 인스턴스화 될 것인지는 서브클래스가 결정하도록 하는 것(Virtual-Constructor 패턴)
- 프로토타입(Prototype): 원본 객체를 복제하는 방법
- 싱글톤(Singleton): 하나의 객체를 여러 프로세스가 동시에 참조할 수 없음

#추빌팩프싱

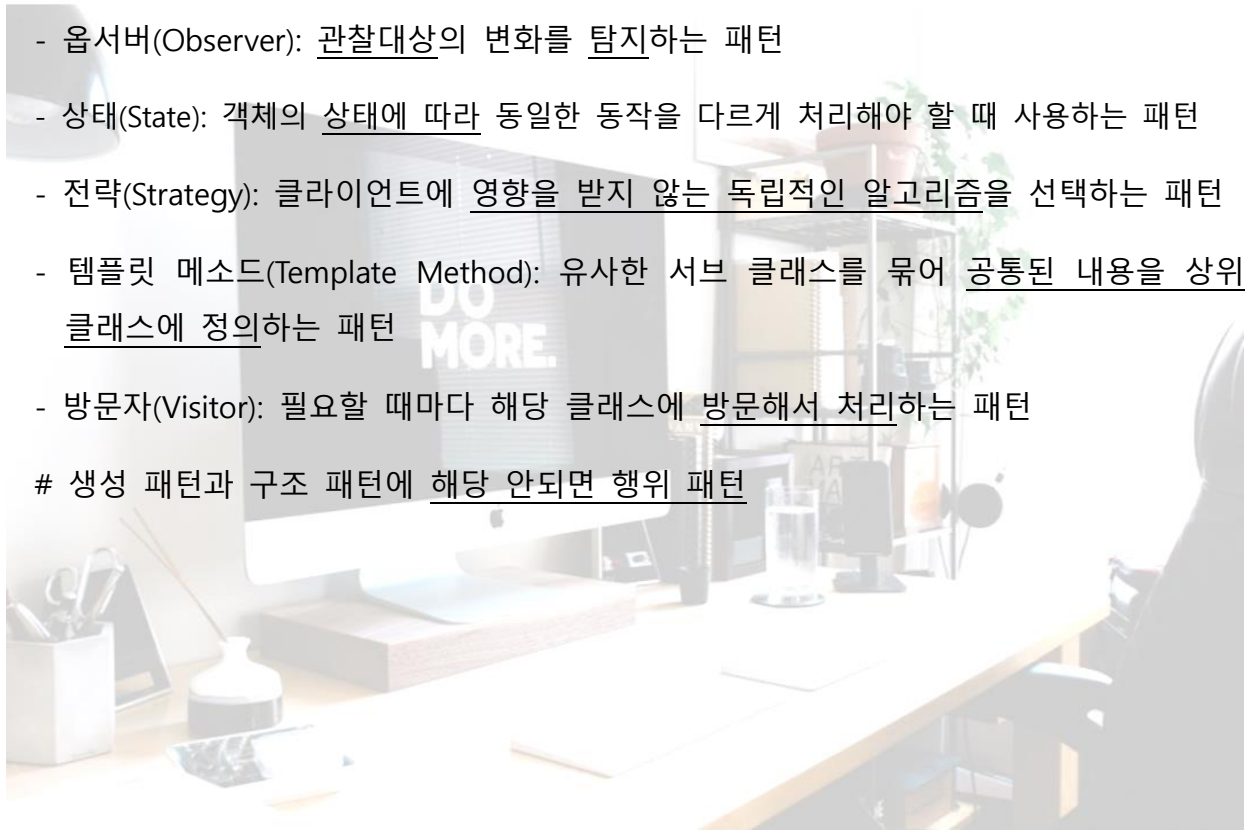
2) 구조 패턴(Structural Pattern) ★

- 어댑터(Adapter): 호환성이 없는 클래스 인터페이스를 이용할 수 있도록 변환해주는 패턴
- 브리지(Bridge): 구현부에서 추상층을 분리하여, 독립적으로 확장 및 다양성을 가지는 패턴
- 컴포지트(Composite): 여러 객체를 가진 복합, 단일 객체를 구분 없이 다룰 때 사용하는 패턴
- 데코레이터(Decorator): 상속을 사용하지 않고도 객체의 기능을 동적으로 확장해주는 패턴
- 퍼싸드(Façade): 서브 클래스들의 기능을 간편하게 사용할 수 있도록 하는 패턴; ex) 리모컨
- 플라이웨이트(Flyweight): 공유해서 사용함으로써 메모리를 절약하는 패턴
- 프록시(Proxy): 접근이 어려운 객체를 연결해주는 인터페이스 역할을 수행하는 패턴

#어브컴데 퍼플프

3) 행위 패턴(Behavioral Pattern)

- 책임 연쇄(Chain of Responsibility): 한 객체가 처리하지 못하면 다음 객체로 넘어가는 패턴
 - 커맨드(Command): 요청에 사용되는 각종 명령어들을 추상, 구체 클래스로 분리하여 단순화함
 - 인터프리터(Interpreter): 언어에 문법 표현을 정의하는 패턴
 - 반복자(Iterator): 동일한 인터페이스를 사용하도록 하는 패턴
 - 중재자(Mediator): 서로의 존재를 모르는 상태에서도 협력할 수 있게 하는 패턴
 - 메멘토(Memento): 요청에 따라 객체를 해당 시점의 상태로 돌릴 수 있는 기능을 제공하는 패턴
 - 옵서버(Observer): 관찰대상의 변화를 탐지하는 패턴
 - 상태(State): 객체의 상태에 따라 동일한 동작을 다르게 처리해야 할 때 사용하는 패턴
 - 전략(Strategy): 클라이언트에 영향을 받지 않는 독립적인 알고리즘을 선택하는 패턴
 - 템플릿 메소드(Template Method): 유사한 서브 클래스를 묶어 공통된 내용을 상위 클래스에 정의하는 패턴
 - 방문자(Visitor): 필요할 때마다 해당 클래스에 방문해서 처리하는 패턴
- # 생성 패턴과 구조 패턴에 해당 안되면 행위 패턴



17 인터페이스 요구사항 검증 ★

p.135, 1-75

1) 요구사항 검증(Requirements Verification)

- 인터페이스 요구사항 검토 계획 수립 → 검토 및 오류 수정 → 베이스라인 설정

2) 요구사항 검증 방법 ★★ _ 20 년 1, 2, 3 회 기출문제

- 동료 검토(Peer Review): 요구사항 명세서 작성자가 내용을 직접 설명하고 동료들이 이를 들으면서 결함을 발견하는 검토 방법
- 워크 스루(Walk Through): 검토회의 전에 요구사항 명세서를 미리 배포하여 사전 검토한 후, 짧은 검토 회의를 통해 결함을 발견하는 검토 방법
- 인스펙션(Inspection): 요구사항 명세서 작성자를 제외한 다른 검토 전문가들이 확인하면서 결함을 발견하는 검토 방법

#동위원

3) 인터페이스 요구사항 검증 주요 항목

- 기능성(Functionality), 완전성(Completeness), 일관성(Consistency), 명확성(Unambiguity), 검증 가능성(Verifiability), 추적 가능성(Traceability), 변경 용이성(Easily Changeable)

#기완일 명검추변

18 인터페이스 ★

p.140, 1-81

1) 인터페이스 식별

- 인터페이스 요구사항 명세서와 인터페이스 요구사항 목록을 기반으로 개발할 시스템과 이와 연계할 내, 외부 시스템 사이의 인터페이스를 식별하고 인터페이스 목록을 작성하는 것

2) 인터페이스 시스템 식별

- 인터페이스별로 인터페이스에 참여하는 시스템들을 송신 시스템과 수신 시스템으로 구분하여 작성하는 것

3) 인터페이스 표준 항목 ★

- 시스템 공통부: 시스템 간 연동 시 필요한 공통 정보
 - # 인터페이스 ID, 전송 시스템 정보, 서비스 코드 정보, 응답 결과 정보, 장애 정보
- 거래 공통부: 시스템들이 연동된 후 송, 수신 되는 데이터를 처리할 때 필요한 정보
 - # 직원 정보, 승인자 정보, 기기 정보, 매체 정보

19 인터페이스 방법 명세화 ★★

p.145, 1-98

1) 시스템 연계 기술 ★★

▶ 직접 연계 방식

- DB 링크(DB link): 수신 시스템에서 DB Link 를 생성하고 송신 시스템에서 해당 DB 링크를 직접 참조하는 방식
ex) 테이블명@DB Link 명
- DB 연결(DB Connection): 수신 시스템의 WAS 에서 송신 시스템 DB 로 연결하는 DB 커넥션 풀(DB Connection Pool)을 생성하고 연계 프로그램에서 해당 DB 커넥션 풀명을 이용
ex) 송신 시스템의 Data Source = DB Connection Pool 이름
- API/Open API: 송신 시스템의 DB 에서 데이터를 읽어와 제공하는 애플리케이션 프로그래밍 인터페이스 프로그램
Open API 는 이런 기능을 누구나 무료로 사용할 수 있도록 공개된 API
- JDBC: 수신 시스템의 프로그램에서 JDBC 드라이버를 이용하여 송신 시스템 DB 와 연결
- 하이퍼 링크(Hyper Link): 웹 애플리케이션에서 하이퍼링크 이용
ex) 구글

#링크연제하

- 연계 솔루션: EAI 서버와 송, 수신 시스템에 설치되는 클라이언트를 이용하는 방식

▶ 간접 연계 방식

- 소켓(Socket): 서버는 통신을 위한 소켓을 생성하여 포트를 할당하고 클라이언트의 통신 요청 시 클라이언트와 연결하는 네트워크 기술
- 웹 서비스(Web Service): 웹 서비스에서 WSDL, UDDI, SOAP 프로토콜을 이용해 연계 하는 서비스
WSDL(Web Services Description Language): 웹 서비스와 관련된 서식이나 프로토콜 등을 표준적인 방법으로 기술하고 게시하기 위한 언어
UDDI(Universal Description, Discovery and Integration): 인터넷에서 전 세계의 비즈니스

업체 목록에 자신의 목록을 등록하기 위한 확장성 생성 언어(XML)기반의 규격
SOAP(Simple Object Access Protocol): 웹 서비스를 실제로 이용하기 위한 객체 간의 통신 규약

- ESB(Enterprise Service **Bus**): 개방형 표준인 웹 서비스를 이용하며, 메시징과 웹 서비스, 데이터 변형, 인텔리전트 라우팅을 결합하여 다양한 애플리케이션 간의 연결과 상호작용을 지원하는 표준기반의 미들웨어 플랫폼

#소웹버

2) 인터페이스 통신 유형

- 단방향: 시스템에서 거래 요청만 하고 응답은 없는 방식
- 동기(Sync): 시스템에서 거래 요청 후 응답이 올 때까지 대기(Request-Reply)하는 방식
ex) 은행 업무: 송금 버튼을 누르면 그 즉시 버튼에 대한 응답으로 돈이 송금됨
- 비동기(Async): 시스템에서 거래 요청 후 다른 작업을 수행하다 응답이 오면 처리하는 방식
ex) 채점하는 교수님: 시험지를 받고 채점하는 건 그 날 즉시해도, 다음 날 채점해도 상관없음

동기, 비동기는 양방향

3) 인터페이스 처리 유형

- 실시간 방식: 사용자가 요청한 내용을 바로 처리해야 할 때 사용하는 방식
- 지연 처리 방식: 매건 단위 처리로 비용이 많이 발생할 때 사용하는 방식
- 배치 방식: 대량의 데이터를 처리할 때 사용하는 방식

#실지배

© 2021. 함께 공부해요 All rights reserved.

4) 인터페이스 발생 주기

- 매일, 수시, 주 1 회 등

[20] 미들웨어 솔루션 명세 ★★

p.152 / 1-97, 20 년 1, 2, 3 회 기출문제

- 운영체제(OS)와 해당 운영체제에서 실행되는 응용 프로그램 사이에서 운영체제가 제공하는 서비스 이외에 추가적인 서비스를 제공하는 소프트웨어

#디원메트 레객와

1) DB(Database)

- 클라이언트에서 원격의 데이터베이스와 연결하기 위한 미들웨어, 2-Tier 아키텍처
ODBC(마이크로소프트), IDAPI(볼랜드), Glue(오라클)

2) RPC(Remote Procedure Call, 원격 프로시저 호출)

- 응용 프로그램의 프로시저를 사용해 원격 프로시저를 로컬 프로시저처럼 호출하는 방식의 미들웨어
Entera(이큐브시스템스), ONC/RPC(OSF)

3) MOM(Message Oriented Middleware, 메시지 지향 미들웨어) ★

- 메시지 기반의 비동기형 메시지를 전달하는 방식의 미들웨어
MQ(IBM), Message Q(오라클), JMS(JCP)

4) TP-Monitor(Transaction Processing Monitor, 트랜잭션 처리 모니터) ★

- 항공기나 철도 예약 업무 등과 같은 온라인 트랜잭션 업무에서 트랜잭션을 처리 및 감시하는 미들웨어
- 사용자 수가 증가해도 빠른 응답 속도를 유지해야 하는 업무에 주로 사용됨
tuxedo(오라클), tmax(티맥스소프트)

5) Legacyware(레거시웨어)

- 기존 애플리케이션에 새로운 업데이트된 기능을 덧붙이고자 할 때 사용되는 미들웨어

6) ORB(Object Request Broker, 객체 요청 브로커) ★

- 객체 지향 미들웨어로 코바(CORBA) 표준 스펙을 구현한 미들웨어

* 코바(CORBA; Common Object Request Broker Architecture):

네트워크에서 분산 프로그램 객체를 생성, 배포, 관리하기 위한 규격을 의미

Orbix(Micro Focus), CORBA(OMG)

7) WAS(Web Application Server, 웹 애플리케이션 서버) ★

- 사용자의 요구에 따라 변하는 동적인 콘텐츠를 처리하기 위해 사용되는 미들웨어

cf) 웹 서버: 정적인 콘텐츠를 처리

- 클라이언트/서버 환경보다는 웹 환경을 구현하기 위한 미들웨어

- HTTP 세션 처리를 위한 웹 서버 기능뿐만 아니라 미션-크리티컬한 기업 업무까지 JAVA, EJB 컴포넌트 기반으로 구현이 가능

#Web Logic(오라클), WebSphere(IBM), JEUS, Tomcat

21 추가 정리, 수제비 및 기출문제 ★★★

1) 플랫폼의 유형 _ 1-2

- **싱글** 사이트 플랫폼: 제휴 관계를 통해 소비자와 공급자를 연결하는 형태
ex) 아이튠즈, 안드로이드 마켓
- **투** 사이트 플랫폼: 두 그룹을 중개하고 모두에게 개방하는 형태
ex) 소개팅 앱
- **멀티** 사이트 플랫폼: 다양한 이해관계 그룹을 연결하여 중개하는 형태
ex) 페이스북, 인스타그램

#싱투멀

2) 플랫폼 성능 특성 분석 기법 _ 1-3

- 사용자 **인터뷰**, 성능 테스트, **산출물** 점검

#인성산

3) OSI 7 계층(Layer) ★★ _ 1-5

- 응용 계층(Application Layer, 7): 사용자와 네트워크 간 응용서비스 연결, 데이터 생성
HTTP, FTP, TELNET, SMTP/SNTP, DNS
- 표현 계층(Presentation Layer, 6): 데이터 형식 설정, 코드변환, 암/복호화
JPEG, MPEG
- 세션 계층(Session Layer, 5): 연결 접속(유지), 동기제어, 동기점(대화)
SSH, TLS
- 전송 계층(Transport Layer, 4): 종단간(End to End) 신뢰성 있고 효율적인 데이터 전송, 데이터 분할, 재조립, 흐름 제어(슬라이딩 윈도우), 오류 제어, 혼잡 제어
TCP/UDP, RTCP → 세그먼트(Segment)

©2021. 함께 공부해요 All rights reserved.

- 네트워크 계층(**Network Layer**, 3): 단말기 간 데이터 전송을 위한 최적화된 경로(라우팅) 제공
IP, ICMP, IGMP, RIP, OSPF → 패킷(Packet)
- 데이터 링크 계층(**Data Link Layer**, 2): 인접 시스템 간 물리적 연결을 이용해 데이터 전송, 동기화, 오류제어, 흐름 제어, 오류검출 및 재전송
HDLC, PPP, LLC, Ethernet(이더넷) → 프레임(Frame)
- 물리 계층(**Physical Layer**, 1): 매체 간의 전기적, 기능적, 절차적 기능 정의
RS-232C → 비트(Bit)

#아(A)파(P)서(S) 티(T)내(Ne)다(Da) 피(Phy)나다!

4) 린(LEAN) _ 1-14 → 애자일(Agile) 방법론 유형 중 하나

- 낭비제거, 품질 내재화, 지식 창출, 낮은 확정, 빠른 인도, 사람 존중, 전체 최적화

#낭품지 확인사전

5) CASE(Computer-Aided Software Engineering) 도구의 분류 _ 1-19, 1, 2 회 기출문제

- 상위 CASE: 계획수립, 요구분석, 기본설계 단계를 다이어그램으로 표현
모순 검사, 오류 검증, 자료흐름도 작성 지원
- 중위 CASE
상세 설계 작업, 화면 출력 작성 지원
- 하위 CASE
시스템 명세서, 소스 코드 생성 지원

© 2021. 함께 공부해요 All rights reserved.

6) UI 컨셉션 세부 수행 활동 _ 1-33

- 정보 구조 설계 → 대표 화면 와이어 프레임 스케치 → 페이퍼 프로토타입을 통한 스토리보드 설계

#정와스

7) UI 설계 프로세스 _ 1-39

- 문제 정의 → 사용자 모델 정의 → 작업 분석 → 컴퓨터 오브젝트 및 기능 정의 → 사용자 인터페이스 정의 → 디자인 평가

#문사작업인디

8) 소프트웨어 설계 유형 _ 1-51

- 자료 구조 설계, 아키텍처 설계, 인터페이스 설계, 프로시저 설계

#자아인프

9) 소프트웨어 아키텍처 4+1 뷰 _ 1-53

- 유스케이스 뷰, 논리 뷰, 프로세스 뷰, 구현 뷰, 배포 뷰

#유논프구배

10) 럼바우의 객체 지향 분석 / 객체 모델링 기법(OMT) ★★ _ 1-59,

1, 2, 3, 4 회 기출문제

- 객체 모델링: 객체 다이어그램
- 동적 모델링: 상태도 (상태 다이어그램)
- 기능 모델링: 자료 흐름도

#객동기

© 2021. 함께 공부해요 All rights reserved.

11) 요구사항 관리 프로세스 _ 1-74, 5-6

- 요구사항 협상 → 요구사항 기준선 → 요구사항 변경관리 → 요구사항 확인 및 검증

#협기변확

12) 인터페이스 정의서 작성 _ 1-95

- 인터페이스 ID, 최대 처리 횟수, 데이터 크기(평균/최대), 시스템 정보, 데이터 정보

#인최크시데

13) 자료 흐름도(DFD; Data Flow Diagram) ★★ _ 개정 전 기출문제, 1, 2, 3 회 기출문제

기 호	의 미	표기법
프로세스 (Process)	자료를 변환시키는 시스템의 한 부분(처리 과정을 나타내며, 처리, 기능, 변환, 버블이라고도 함)	○
자료 흐름(Flow)	자료의 이동을 나타냄	→
자료 저장소 (Data Store)	시스템에서의 자료 저장소(파일, 데이터 베이스)를 나타냄	— — —
단말 (Terminator)	시스템과 교신하는 외부 개체로, 입력 데이터가 만들어지고, 출력 데이터를 받음 (정보의 생산자와 소비자)	□

#PTSD → Process, Terminator, Data Store, Data Flow

14) UML 확장 모델의 스테레오 타입 객체 표현 기호 _ 20 년 1, 2 회 기출문제

<< >>

15) 자료 사전 기호 _ 개정 전 기출문제, 20 년 1, 2, 3 회 기출문제

기호	의미
=	자료의 정의: ~로 구성되어 있다(is composed of)
+	자료의 연결: 그리고(and)
()	자료의 생략: 생략 가능한 자료(Optional)
[]	자료의 선택: 또는(or)
{ }	자료의 반복: 자료 반복(Iteration of)
* *	자료의 설명: 주석(Comment)

16) 객체지향 기법에서 클래스들 사이의 '부분-전체(part-whole)'관계 또는 '부분(is-a-part-of)'의 관계로 설명되는 연관성을 나타내는 용어는? _ 20 년 1, 2 회 기출문제

- ① 일반화 ② 추상화 ③ 캡슐화 ④ 집단화

① 일반화(Generalization)

일반화는 객체들에 있어 공통적인 성질들을 상위 객체로 정의하고, 특수화(specialization)된 객체들을 하위의 부분형(subtype) 객체로 정의하는 추상화 방법이다.

② 추상화(Abstraction)

현실 세계의 물체를 객체에 사상할 때에는 현실을 그대로 객체로서 표현하는 것이 아니고, 문제의 중요한 측면, 혹은 주목하고 싶은 문제의 측면을 강조한다. 즉, 불필요한 부분을 생략하고 객체의 가장 중요한 것에만 중점을 두어 간략화 시킨 것을 말한다.

③ 캡슐화(Encapsulation)

객체의 자료와 행위를 하나로 묶고, 실제 구현 내용을 외부에 감추는 것을 캡슐화라고 한다. 이렇게 캡슐화 된 객체의 행위는 외부에서 볼 때는 실체가 아닌 추상적인 것이 되므로 정보 은닉(information hiding)개념이 지켜진다.

+ 정보 은닉은 객체가 캡슐화를 통하여 내부의 데이터나 오퍼레이션의 구현 내용을 감추는 것을 말한다. 즉, 외부에서의 무분별한 접근을 허용하지 않는다는 것이다.

④ 집단화(Aggregation)

집단화는 서로 관련 있는 여러 개의 객체를 묶어 한 개의 상위 객체를 만드는 것이다. 집단화는 여러 개의 속성을 묶어 사용자 정의형의 엔티티를 만드는 수단으로 사용한다. 한 객체에서 하나 이상의 객체는 사용자 표정의 형이 될 때 복합 객체(composite object)라고 불린다. 즉, 집단화는 복합 객체의 종속 성분을 모델링하기 위해 사용되며, 이들 복합 성분 클래스 관계를 통해 복합 속성 계층(composite attribute hierarchy)을 형성한다.

17) HIPO(Hierarchy Input Process Output) _ 개정 전 기출문제, 20 년 1, 2 회 기출문제

- 하향식 소프트웨어 개발을 위한 문서화 도구
- HIPO 차트 종류: 가시적 도표, 총체적 도표, 세부적 도표
- 기능과 자료의 의존 관계를 동시에 표현할 수 있음
- 보기 쉽고 이해하기 쉬움

18) 객체지향 분석 방법론 _ 개정 전 / 20 년 1, 2 회 / 21 년 1 회 기출문제

▶ Coad 와 Yourdon 방법 ★

- E-R 다이어그램을 사용하여 객체의 행위를 모델링하며, 객체 식별, 구조 식별, 주체 정의, 속성 및 관계 정의, 서비스 정의 등의 과정으로 구성되는 것

▶ Booch 방법

- 미시적(Micro) 개발 프로세스와 거시적(Macro) 개발 프로세스를 모두 사용하는 분석 방법

▶ Jacobson 방법

- 유스케이스(Use Case, 사용사례)를 강조하여 사용하는 분석 방법

▶ Wirfs-Brocks 방법

- 분석과 설계 간의 구분이 없고, 고객 명세서를 평가해서 설계 작업까지 연속적으로 수행하는 분석 방법

19) UML 의 시퀀스 다이어그램의 구성 항목 _ 20 년 3 회 기출문제

- 생명선(Life line), 실행(Activation, 활성 박스), 메시지(Message)

#생실메

20) 객체지향 설계 원칙 ★ _ 개정 전 기출문제, 20 년 3 회 기출문제

원칙	설명
SRP , 단일 책임 원칙 (Single Responsibility Principle)	소프트웨어의 설계 부품(클래스, 함수 등)은 <u>단 하나의 책임만을</u> 가져야 함
OCP , 개방-폐쇄 원칙 (Open-Closed Principle)	기존의 코드를 변경하지 않고(<u>Closed</u>), 기능을 수정하거나 추가할 수 있도록(<u>Open</u>) 설계해야 함
LSP , 리스코프 치환 원칙 (Liskov Substitution Principle)	서브타입(하위 클래스, 자식 클래스)은 어디에서나 <u>자신의 기반타입(상위 클래스, 부모클래스)으로 교체</u> 할 수 있어야 함
ISP , 인터페이스 분리 원칙 (Interface Segregation Principle)	한 클래스는 <u>자신이 사용하지 않는 인터페이스는 구현하지 않아야 함</u> → 자신이 사용하지 않는 기능(인터페이스)에는 영향을 받지 않아야 함
DIP , 의존 역전 원칙 (Dependency Inversion Principle)	의존 관계를 맺을 때, 변화하기 쉬운 것보단 <u>변화하기 어려운 것에 의존해야 한다</u> 는 원칙

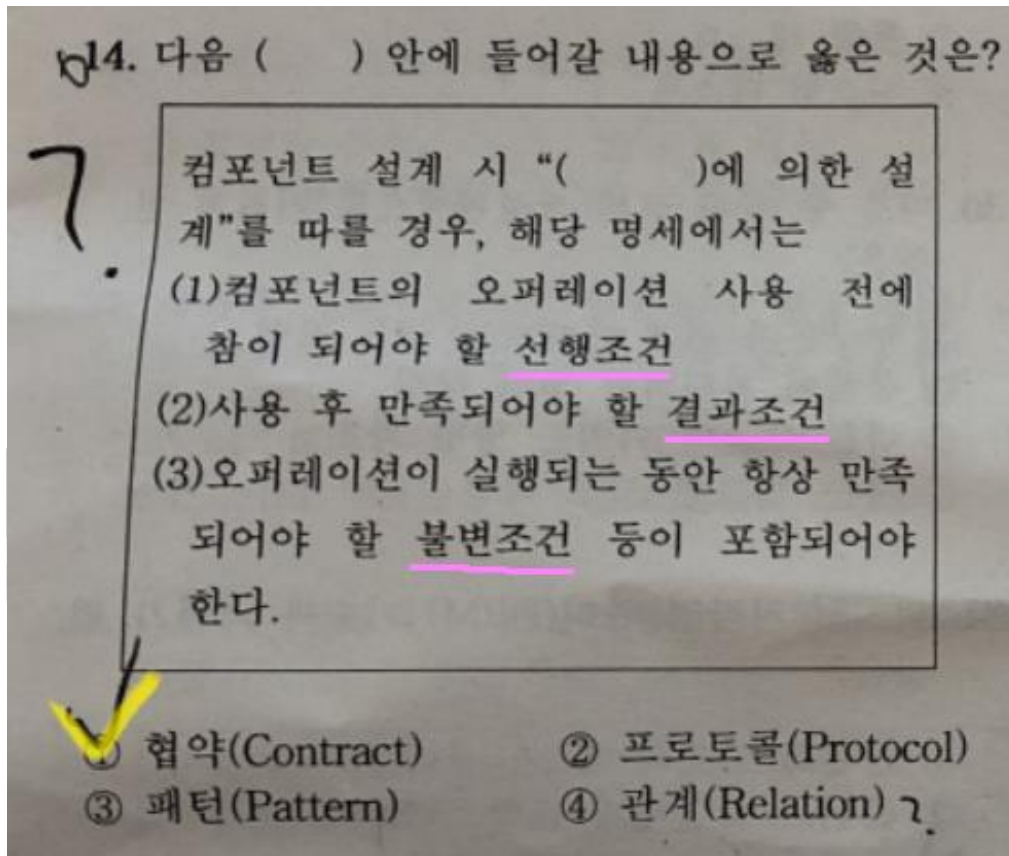
#SOLID

21) 디자인 패턴 구성요소 _ 20 년 3 회 기출문제

구성요소	설명
패턴의 이름과 구분	패턴을 부를 때 사용하는 이름과 패턴의 유형
문제 및 배경	패턴이 사용되는 분야 또는 배경, 해결하는 문제를 의미
솔루션	패턴을 이루는 요소들, 관계, 협동(Collaboration) 과정
사례	간단한 적용 사례
결과	패턴을 사용하면 얻게 되는 이점이나 영향
샘플코드	패턴이 적용된 원시코드(Source Code)

#패문솔 사결샘

22) DBC(Design by Contract, 계약에 의한 설계) _ 20 년 3 회 기출문제



- 프로그램 모듈들의 책임을 문서화하는데 초점을 맞춤
- 각각의 모듈이 가져야 하는 기능만큼만 동작하도록 함
- 위의 개념들을 문서화하고 검증하는 것이 핵심임

23) CASE(Computer-Aided Software Engineering) 도구 _ 20 년 3, 4 회 기출문제

- 소프트웨어 개발 과정의 일부 또는 전체를 자동화하기 위한 도구
- 표준화된 개발 환경 구축 및 문서 자동화 기능 제공
- 작업 과정 및 데이터 공유를 통해 작업자 간의 커뮤니케이션 증대

주요기능: S/W 라이프 사이클 전 단계의 연결, 그래픽 지원, 다양한 소프트웨어 개발 모형 지원