

소프트웨어 설계



Software Engineering

©2014 Kim, Haeng Kon

우리는 설계를 서둘러 끝냄으로써
개발에 많은 시간을 배려하려고 하지만,
그 개발이란 서둘렀던 설계상의 문제를
해결하는 데 보다 많은 시간을 낭비하고
만다.

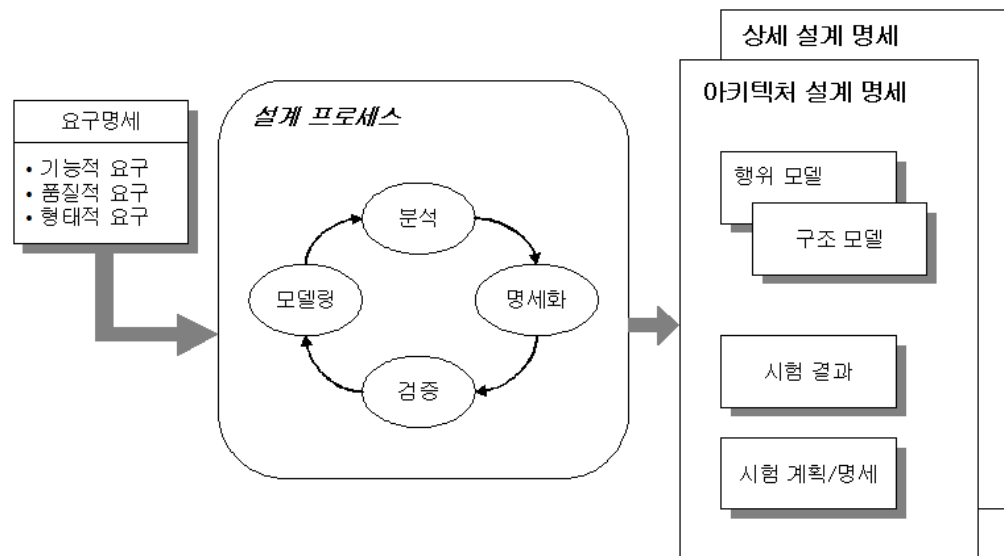
- G.Myers, 1978



소프트웨어 설계 개념(1)

• 소프트웨어 설계

- 요구사항 분석 단계에서 규명된 필수 기능들의 구체적인 구현 방법을 명시하는 단계
- 장치, 프로세스 그리고 시스템을 명확하고 자세하게 정의하며 실질적으로 실현가능 하도록 관련된 기술과 원칙을 적용하는 과정
- 소프트웨어에 요구되는 기능과 성능 조건들을 만족하는 소프트웨어의 내부 기능, 구조 및 동적 행위들을 모델링하여 표현, 분석, 검증하는 과정 혹은 그러한 과정의 산출물을 의미
- 목적 : “무엇을(What)” 으로부터 “어떻게(How)”로 관점이 바뀌며 구현할 소프트웨어의 청사진을 만드는 것



소프트웨어 설계 개념(2)

— 설계 원칙

- 소프트웨어 설계는 변경이 용이하도록 구조화되어야 함
- 하나의 함수 안에는 특정 기능을 수행하는데 필요한 자료만을 사용하도록 규제할 것
- 독립적이고 기능적인 특성을 지닌 모듈단위로 분할 설계
- 계층적 구조를 가져야 함

소프트웨어 설계 대상(1)

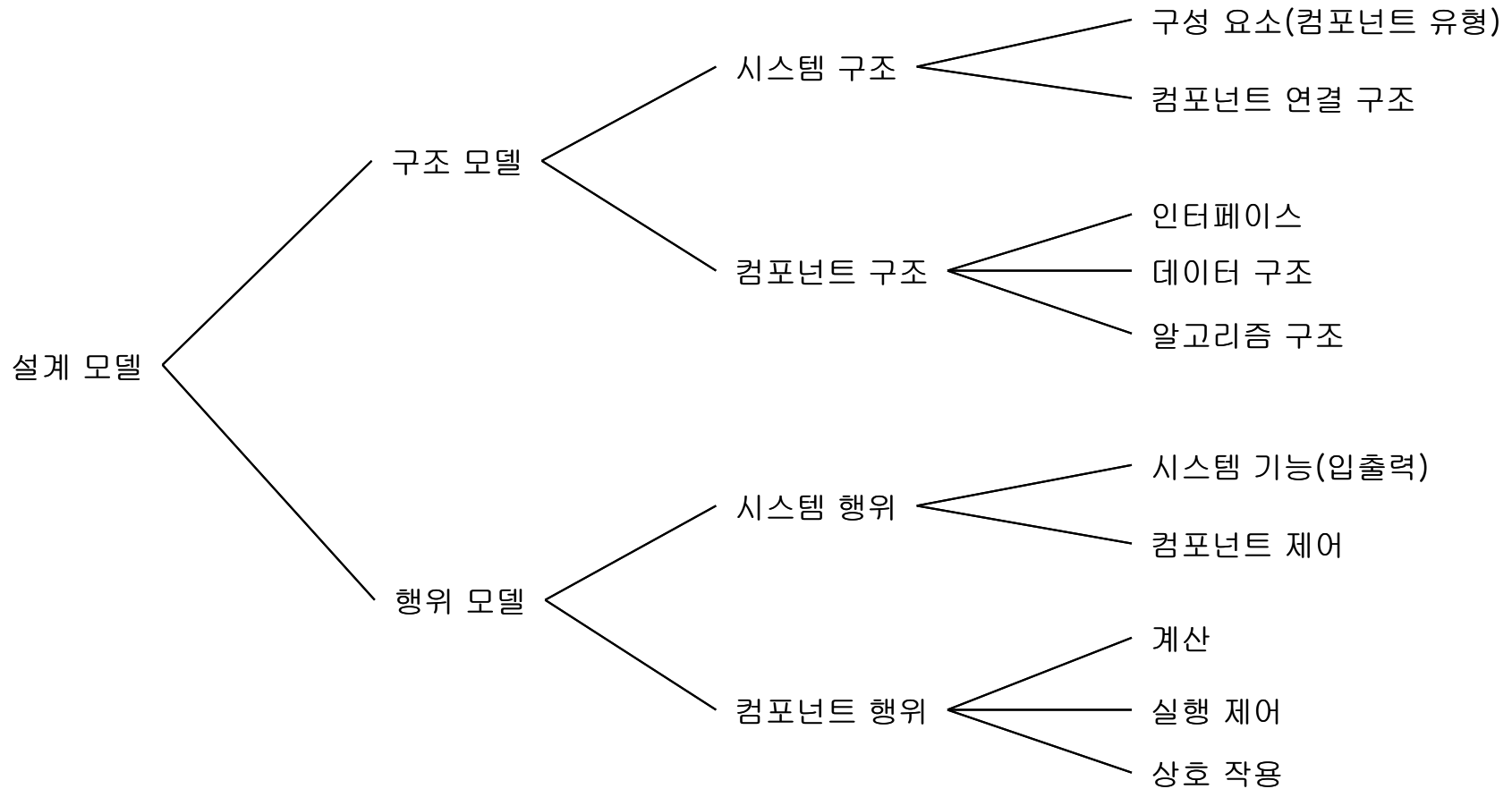
1) 구조와 행위 모델링

- 구조 모델링 : 소프트웨어를 구성하는 컴포넌트들의 유형, 인터페이스, 내부 설계 구조 및 이들의 상호 연결 구조를 모델링
- 행위 모델링 : 소프트웨어의 구성요소들의 기능들과 이들이 언제, 어떠한 순서로 기능을 수행하고 상호 작용하는지를 모델링

모델 유형	개 요	내 용
구조모델 (structure models)	시스템의 구성 요소들과 이들 사이의 구조적인 관계와 특성등을 모델링	구성 요소 : 프로시저, 데이터 구조, 모듈, 파일 구조 시스템 구조 : 구성 요소들의 연결 구조, 포함 관계
행위 모델 (behavior models)	시스템의 각 구성 요소들의 기능적인 특성등을 모델링	입력 데이터, 출력 데이터, 데이터 흐름, 데이터 변환, 데이터 저장 등
	시스템의 구성 요소들이 언제 어떠한 순서로 수행되는가와 같은 동적인 특성등을 모델링	상태 전이, 데이터 흐름 경로, 사건 발생 순서, 실행 경로 등

소프트웨어 설계 대상(2)

- 소프트웨어 설계 모델의 구성



소프트웨어 설계 대상(3)

2) 소프트웨어 설계 모델의 정적 요소와 동적 요소

	정적(Static) 요소	동적(Dynamic) 요소
구조 모델 (Structure model)	<ul style="list-style-type: none"> • 구성 요소의 유형 및 유형 계통 • 구성 요소들의 배열, 결합 관계 • 구성 요소들의 인터페이스 • 구성 요소들의 상호 작용 채널 	<ul style="list-style-type: none"> • 다이내믹 생성 및 소멸 • 동적 결합/연결 • 위치 이동, 복제
행위 모델 (Behavior model)	<ul style="list-style-type: none"> • 입력 데이터, 출력 데이터 • 임추적 mapping • 데이터 흐름 채널 	<ul style="list-style-type: none"> • 제어 • 상호 작용 프로토콜 • 상호 작용 실행 경로 • 상태 전이 • 처리 순서, 임/출력 순서, 알고리즘

소프트웨어 설계 유형

- 자료구조 설계(Data structure design)
 - 요구분석 단계에서 생성된 정보를 바탕으로 소프트웨어를 구현하는데 필요한 자료구조로 변환하는 과정
- 아키텍처 설계(Architecture design)
 - 예비 설계 또는 상위 수준 설계
 - 소프트웨어 시스템의 전체 구조를 기술
 - 소프트웨어를 구성하는 컴포넌트들 간의 관계를 정의
- 인터페이스 설계(Interface design)
 - 소프트웨어와 상호 작용하는 컴퓨터 시스템, 사용자 등이 어떻게 통신하는지를 기술
- 프로시저 설계(Procedure design)
 - 알고리즘 설계
 - 프로그램 아키텍처의 컴포넌트를 소프트웨어 컴포넌트의 프로시저 서술로 변환하는 과정

설계 방법

- 구조적 설계(structured design)
 - 소프트웨어에 요구된 기능, 자료 처리 과정, 알고리즘 등을 중심으로 시스템을 분해하여 설계하는 방식(기능적 관점)
 - 시스템의 각 모듈은 최상위 기능에서 하위 계층으로 하향적 세분화
 - Yourdon/Coad
- 자료구조 중심 설계
 - 입출력 자료의 구조를 파악함으로써 소프트웨어 구조를 추출하는 방식
 - Jackson, Warnier-Orr
- 객체지향 설계(object-oriented design)
 - 자료와 자료에 적용될 기능을 함께 추상화하는 개념(객체=자료+기능)
 - 시스템은 객체의 모임
 - Yourdon, Sheller/Meller, Rumbaugh, Booch...

설계시 고려 사항들

- **모듈(Module)**
 - 독립적으로 처리할 수 있는 구별단위(Identifiable unit), 그 단위들은 하나 이상의 프로시저들을 포함.
 - 모듈의 개념 사용
 - 모듈은 소프트웨어에 꼭 나타남
- **모듈 분할의 바람직한 방법의 특징**
 - 설계의 질을 측정 시 사용
 - 유지보수와 재사용 용이
 - 영향을 주는 설계 형태
 - 추상화(Abstraction)
 - 모듈화(Modularity)
 - 정보 은폐(Information Hiding)
 - 복잡도(Complexity)
 - 시스템 구조(System Structure)

설계 원칙(design principles)

- **모듈화(Modularity)**
 - 프로그램을 작고 독립적인 단위로 분할하여 개발하는 것
 - 변경에 의한 영향을 최소화
- **추상화(Abstraction)**
 - 구체적인 데이터의 내부 구조를 외부에 알리지 않으면서 데이터를 사용하는데 필요한 함수만을 알려주는 기법
- **정보 은닉(Information hiding)**
 - 각 모듈의 자세한 처리 내용이 시스템의 다른 부분에게 감추어짐
- **단계적 분해(Stepwise refinement)**
 - 상위 개념에서 좀더 상세화된 하위개념으로 구체화시키는 과정
- **구조화(Structure)**
 - 시스템을 소프트웨어의 구성요소인 모듈의 계층적 구조형태로 표현

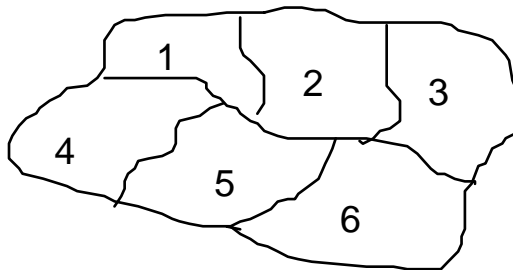
모듈화(Modularity) (1)

• 모듈화 과정 중 고려해야 할 사항들

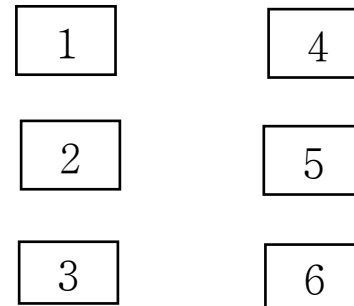
- 시스템을 어떻게 분해할 것인가?
- 한 모듈의 규모는 어느 정도인가?
- 모듈의 응집력(cohesion)은 높게
- 모듈 간의 결합력(coupling)은 약하게

결합도: 모듈간의 상호 의존도

응집도: 모듈 내부의 처리 요소들간의 기능적 연관도



문제영역



시스템 분해

$$C(p1) > C(p2) \Rightarrow E(p1) > E(p2)$$

$$E(p1 + p2) > E(p1) + E(p2)$$

모듈화(Modularity) (2)

- 모듈(Module)

- 소프트웨어 구조를 이루는 기본 단위
- 서브시스템(subsystem), 서브루틴(subroutine), 작업 단위(work unit)
- 시스템을 기능단위로 구분하며 독립적으로 컴파일되고 하나의 입구와 하나의 출구를 가짐
- 모듈당 비용과 접속 관계 소요 비용
 - 모듈의 수가 증가하면 전체 개발 비용은 감소하지만 인터페이스 비용 증가
 - ⇒ 최적의 모듈 개수 결정이 중요
- 장점
 - 복잡도 감소, 수정 용이, 구현 용이
 - 확장성, 융통성, 유지보수성, 재사용성, 경제성

추상화(Abstraction)

- 객체의 속성 중 불필요한 부분은 생략하거나 숨기고 가장 필수적인 속성만으로 주어진 객체를 묘사하는 것
- 전체적이고 포괄적인 개념으로부터 차례로 자세하게 세분화 함으로써 구체화하는 방법
- 유형
 - 기능 추상화(Functional Abstraction)
 - 입력 자료를 출력 자료로 변환하는 과정을 추상화하는 방법
 - 자료 추상화(Data Abstraction)
 - 자료와 자료에 적용 가능한 연산을 함께 정의함으로써 자료 객체를 구성하는 방법(객체 = 자료 + 연산)
 - 제어 추상화(Control Abstraction)
 - 제어의 정확한 메커니즘을 정의하지 않고 원하는 효과를 정의하는 데 이용

정보은닉(Information hiding)

- 모듈간의 불필요한 상호작용을 제거하기 위해 최소한의 정보만 보여주고, 각 모듈의 자세한 처리 내용은 시스템의 다른 부분으로부터 감추어져 접근되지 않도록 하는 것
- 각 모듈이 다른 모듈에 독립적으로 설계 가능
 - 캡슐화(Encapsulation)를 통해 모듈의 추상화와 독립성 향상
- 모듈 단위의 수정, 시험, 유지보수에 큰 장점
 - 모듈 설계 평가에 기초

단계적 분해(Stepwise refinement)

- 단계적인 프로그램 개발
 - 시스템을 상위 레벨에서 좀더 구체화된 하위 레벨로 분할하는 하향식 기법(Wirth에 의해 최초로 제시)
- 활동 과정
 - 문제를 기본 단위로 분해
 - 독립된 문제로 구별
 - 가능한 상세한 내역(알고리즘, 자료구조)에 관한 결정은 뒤로 연기
 - 계속 점증적으로 구체화 작업을 진행
- 단계적 분해의 주요 이점
 - 하나의 문제는 관리 가능한 작은 단위로 분리
 - 임의의 특정 시간에 다루어야 하는 상세 내역의 양은 최소화
 - 설계자의 사고 과정은 적절한 시기에 적절한 관점으로 모아짐

소프트웨어 구조도(Structure)

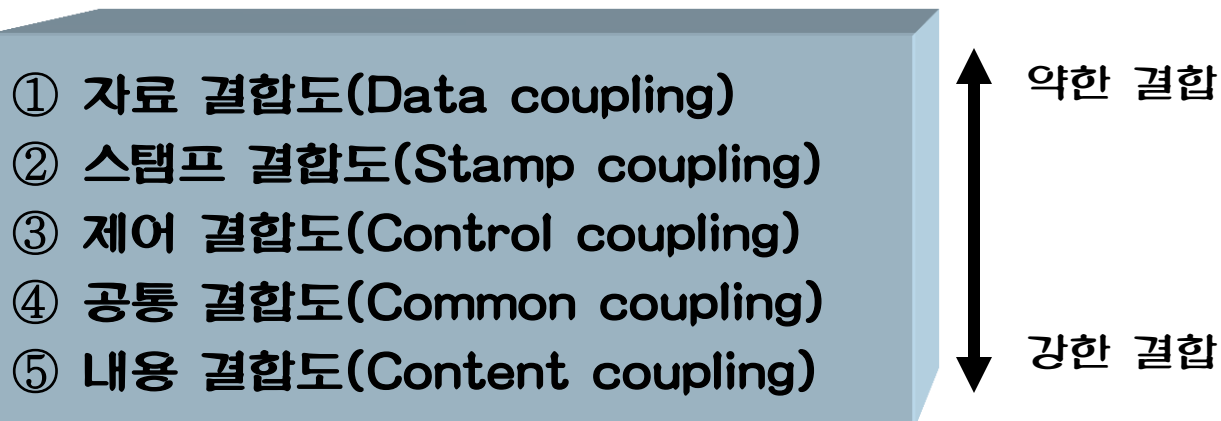
- 소프트웨어 구조도

- 시스템이 어떤 모듈로 구성되어 있으며 모듈 사이의 관계가 어떠한가를 표현
- 각 모듈은 명칭을 가지며, 다른 모듈을 호출하거나 호출될 수 있음
- Fan-In
 - 임의의 한 모듈을 호출하는 상위 모듈의 수
 - 얼마나 많은 모듈이 주어진 모듈을 호출하는가를 나타냄
- Fan-Out
 - 임의의 한 모듈에 의해 호출되는 모듈의 수

모듈의 독립성(1)-결합도

• 결합도(Coupling)

- 모듈간의 상호의존도 또는 연관 관계
- 목표: 가능한 낮은 결합력을 갖도록 설계
- 장점
 - 파급효과(Ripple Effect)의 최소화 가능
 - 모듈의 독립적 유지보수 및 변경 가능
 - 특정 모듈의 내부 상황을 자세히 알지 못해도 그와 관련된 다른 모듈의 효과적 취급 가능



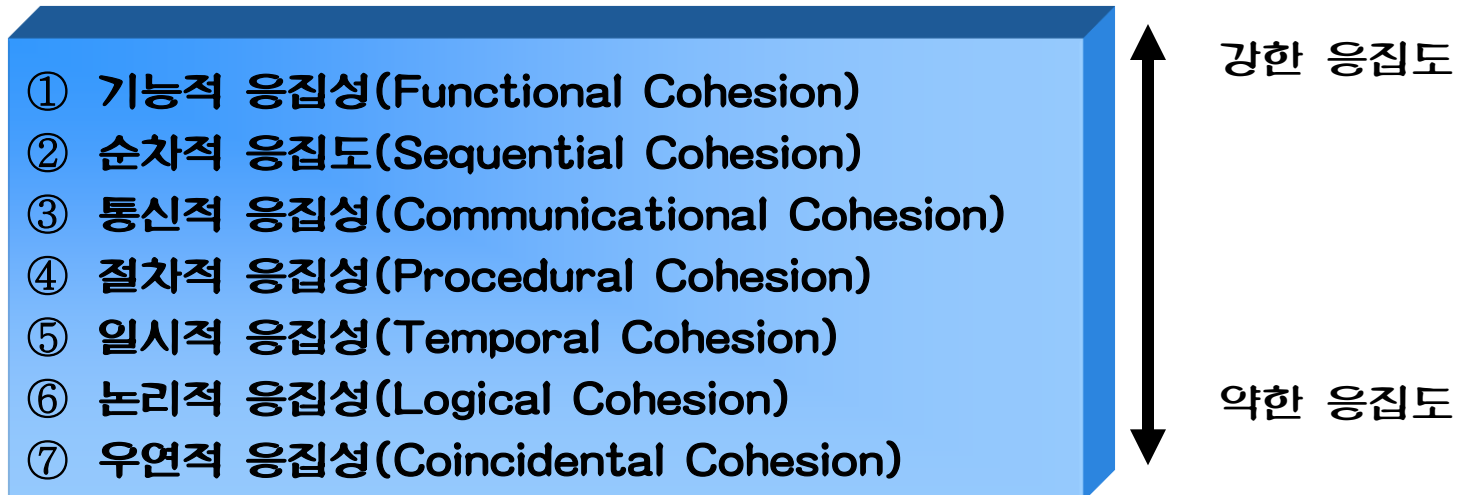
모듈의 독립성(2)-결합도

- 1) Data Coupling(자료 결합도)
 - 모듈들이 간단한 변수를 파라메타로 교환하는 경우
- 2) Stamp Coupling(스탬프 결합도)
 - 두 모듈이 동일한 자료구조를 조회하는 경우
- 3) Control Coupling(제어 결합도)
 - 제어신호를 이용하여 통신하는 경우
- 4) Common Coupling(공통 결합도)
 - 두 모듈이 동일한 총체적 자료 영역(global data area)을 공동으로 조회하는 경우
- 5) Content Coupling(내용 결합도)
 - 한 모듈이 다른 모듈의 내부 기능 및 내부 자료를 조회하는 경우
⇒ 한 모듈이 다른 모듈내의 값을 수정할 수 있는 상태

모듈의 독립성(3)-응집도

• 응집도(Cohesion)

- 한 모듈 내부의 처리 요소들간의 기능적 연관도
- 모듈 안의 구성 요소들이 공동의 목적을 달성하기 위하여 관련되어 있는 정도
- 목표 : 가능한 강한 응집도를 갖게 설계
 - 한 모듈이 단일 기능을 갖도록 설계



모듈의 독립성(4)-응집도

1) 기능적 응집성(Functional Cohesion)

- 한 모듈이 한가지 기능만을 수행하는 경우
 - ex) 제공근 계산, 판매 세금 계산...

2) 순차적 응집성(Sequence Cohesion)

- 한 모듈내의 소 작업 결과가 작업의 입력이 된 경우
 - ex) 보험료 계산하고 분기별 지출액 산출

3) 통신적 응집성(Communication Cohesion)

- 동일한 입력과 출력자료를 이용하여 서로 다른 기능을 수행
 - ex) 출력 파일을 출력하고 저장

4) 절차적 응집성(Procedural Cohesion)

- 모듈안의 작업들이 큰 테두리 안에서는 같은 작업에 속하고 입출력을 공유하지만 순서에 따라 수행될 필요가 있는 경우
 - ex) 재 수행 처리 : 합계를 출력하고 화면을 지우고 메뉴를 보여주고 메뉴 선택을 받아들임

모듈의 독립성(5)-응집도

5) 일시적 응집성(Temporal Cohesion)

- 모듈의 기능 요소들이 시간에 의해 연결되는 단위처리에 포함된 경우
 - ex) 프로그램 초기화 작업

6) 논리적 응집성(Logical Cohesion)

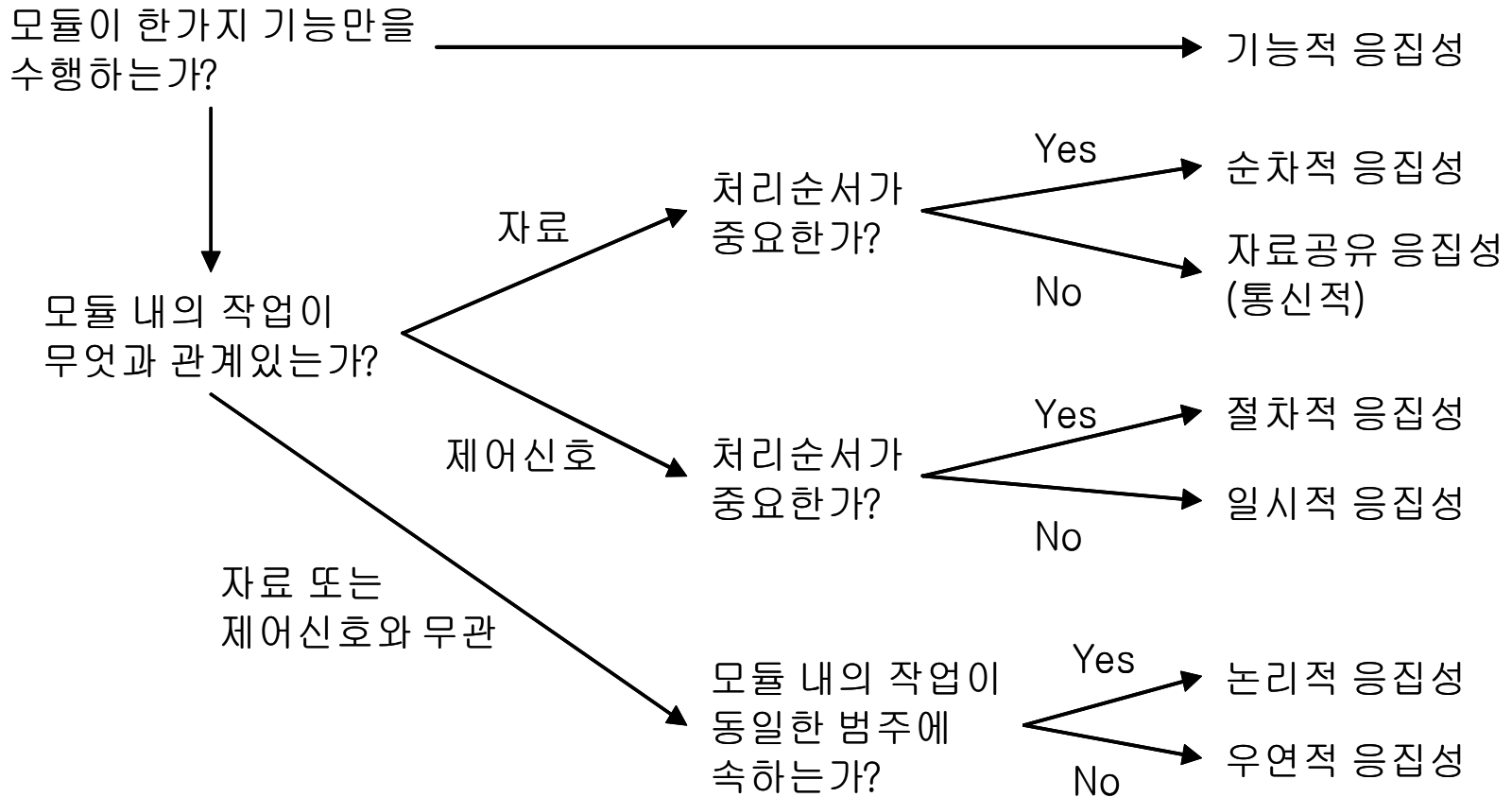
- 유사한 성격의 작업을 하나의 모듈로 구성하는 경우
 - ex) 입출력 처리

7) 우연적 응집성(Coincidental Cohesion)

- 아무 관계없는 요소들을 하나의 모듈로 구성하는 경우
 - ex) 혼합 기능 수행

모듈의 독립성(6)

응집성 구분 흐름도

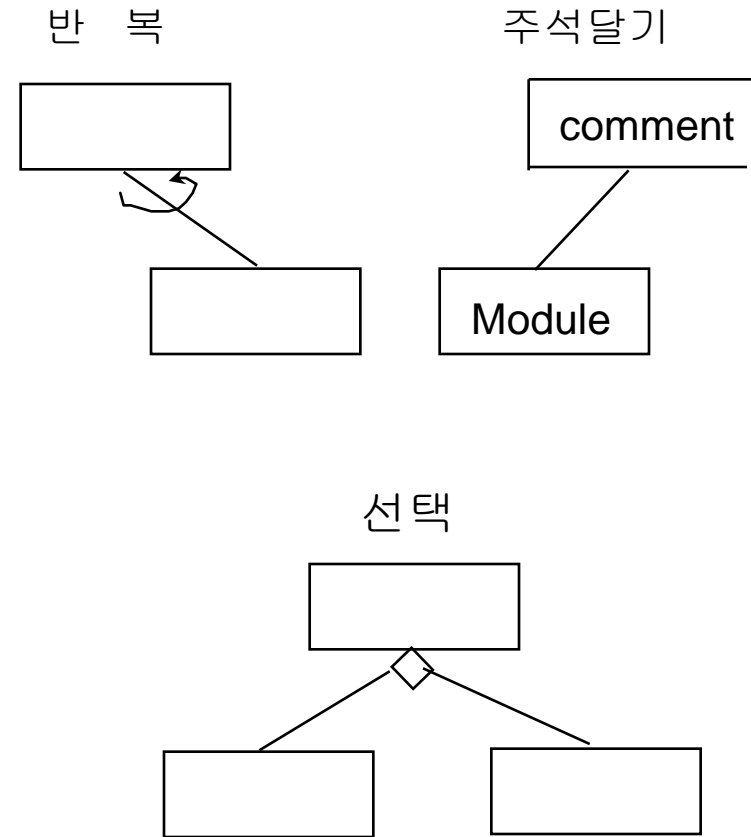
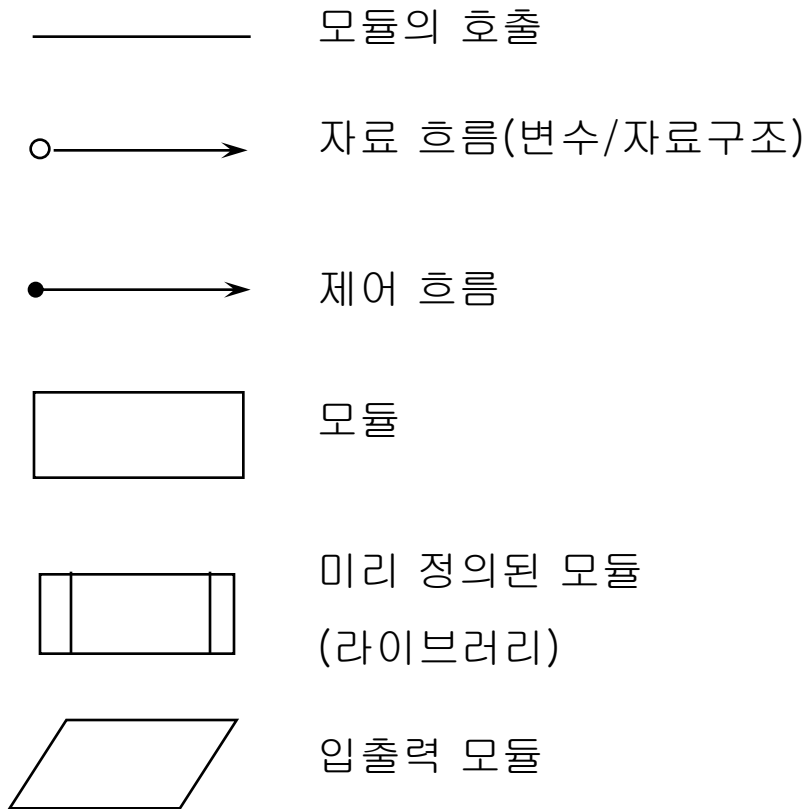


소프트웨어 구조도(1)

- 소프트웨어 구조도
 - 시스템이 어떤 모듈로 구성되어 있으며 모듈 사이의 관계가 어떠한가를 표현
 - 각 모듈에 의해 수행되는 기능도 기술
 - 각 모듈은 명칭을 가지며, 다른 모듈을 호출
- 자료 흐름도를 구조도로 변환하는 전략
 - 변환 흐름 전략(transition flow)
 - 거래 흐름 전략(transaction flow)

소프트웨어 구조도(2)

표준 기호

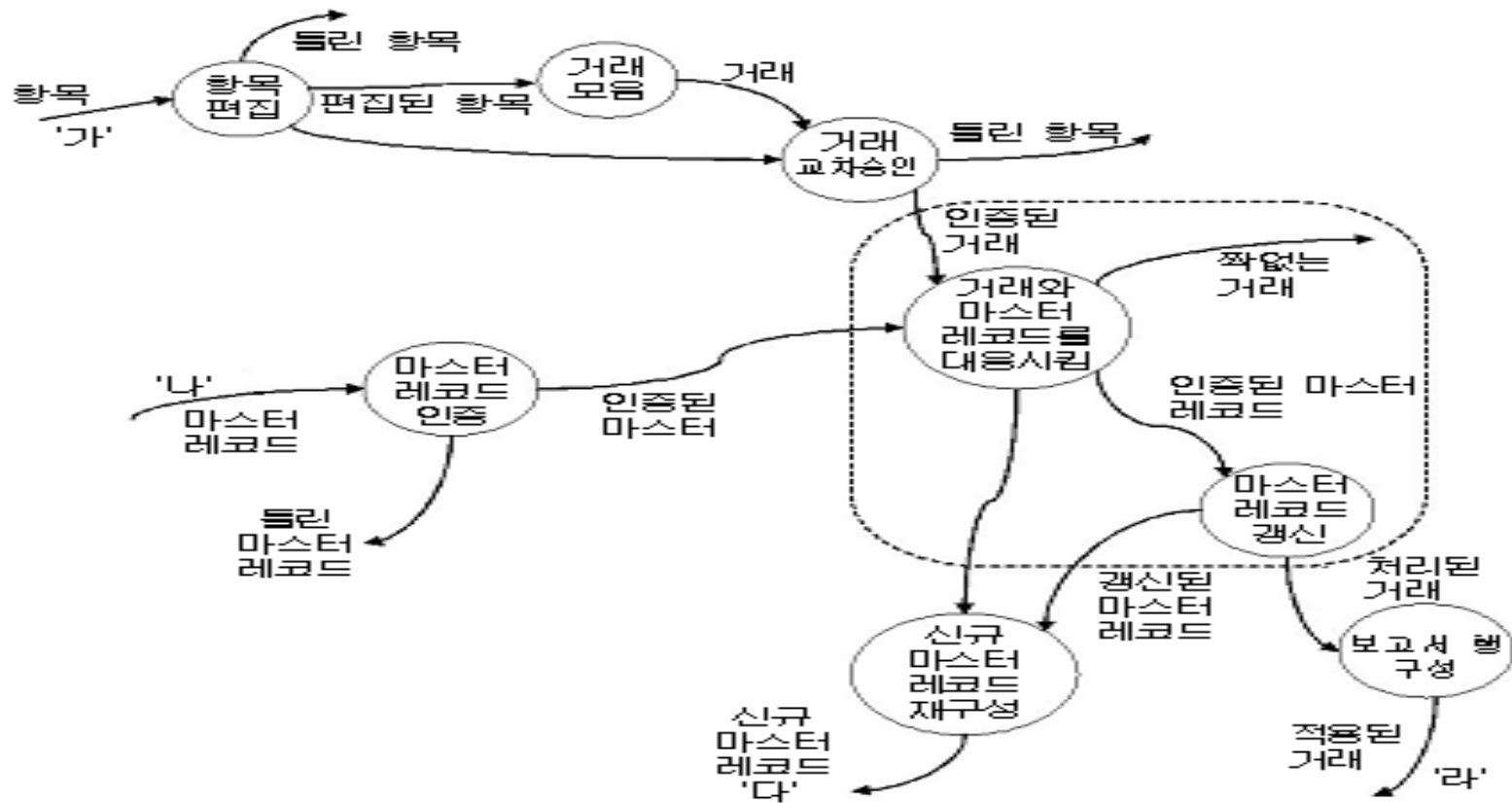


소프트웨어 구조도(3)

- 변환흐름 전략(transition flow)
 - 일반적인 자료의 흐름
 - 입력흐름 ⇒ 변환 흐름 ⇒ 출력 흐름
 - 과정
 - DFD에서 입력 자료와 출력 자료 흐름을 파악
 - 중심 기능인 변환 부분 식별
 - 시스템의 필수적인 기능을 내포
 - 입출력의 특정 구현과는 독립적
 - 최상위 구조도 작성
 - 분할된 각 부분을 같은 방법으로 분할
 - 구조도는 한 단계씩 추가

소프트웨어 구조도(4)

- 변환흐름을 가진 자료 흐름도

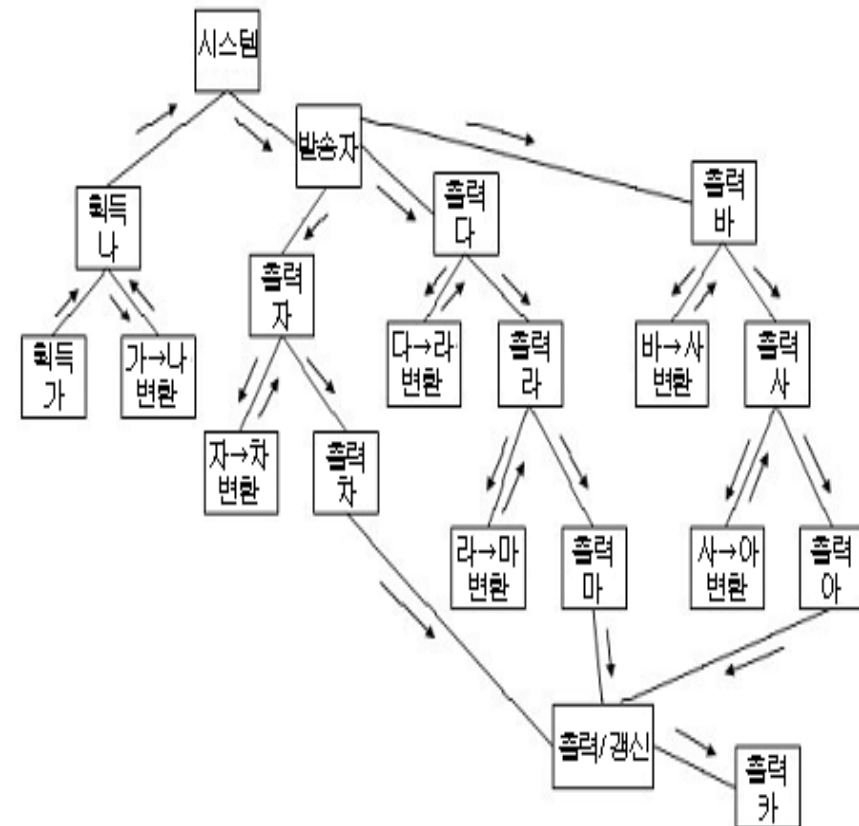
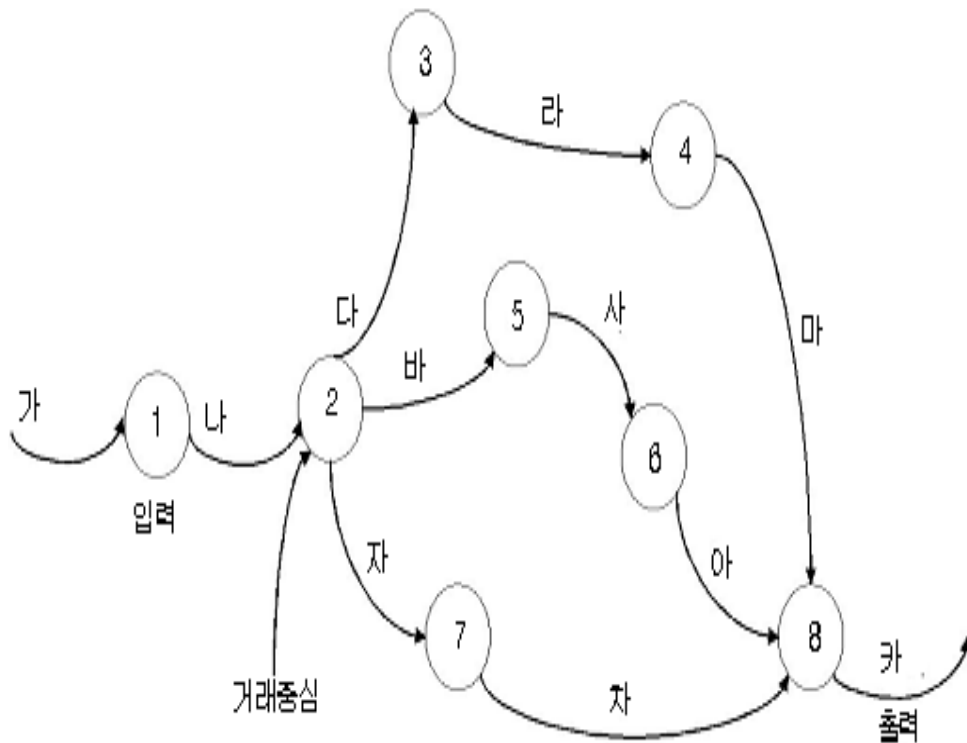


소프트웨어 구조도(6)

- 거래 흐름 전략(transaction flow)
 - 자료 흐름도의 한 프로세스에서 여러 개의 자료 흐름이 유출되는 경우에 사용
 - 거래 : 어떤 동작이나 또는 연속된 동작을 발생, 유발 또는 시작시키는 어떤 자료요소, 제어, 신호, 사건 또는 상태의 변화등
 - 과정
 - DFD에서 처리 센터 식별
 - 처리 모듈을 중심으로 구조도 작성
 - 구조도 상세화 - 하위 구조도 작성

소프트웨어 구조도(7)

- 거래 흐름에 의해 유도된 구조도
- 거래흐름을 가진 DFD



모듈 명세서(1)

- 모듈 명세서
 - Module 내부에 대한 명세
 - 시스템 구조도에 표현되지 않은 상세 알고리즘 기술
 - 알고리즘 선택 시 주의 사항
 - 정확성 : 알고리즘은 기본적으로 정확해야 함
 - 효율성 : 알고리즘의 수행에 소요되는 컴퓨터 자원의 양
(기억장치의 소요량과 수행시간)
 - 적합성 : 알고리즘은 목표 시스템의 h/w와 s/w에 적합

모듈 명세서[2]

- 모듈 명세화 기법
 - 흐름도(flow chart)
 - N-S 도표(Nassi-Schneiderman Chart)
 - 의사 코드(pseudo code)
 - 의사 결정표(decision table)
 - 의사 결정도(decision diagram)
 - PDL(Program Design Language)
 - 상태천이도(state transition diagram)
 - 행위도(action diagram)

모듈 명세서(3)

모듈 총자산 구함(고객 구조표, 표길이:총자산)

/× 구조번호에 의해 분류된 은행 고객 집단의 순자산 총액 구함 ×/

/× 고객 구조표에는 합계될 모든 고객의 인증된 구조번호가 들어 있음 ×/

/× 표길이는 고객수임 ×/

/× 총자산은 모든 고객의 순자산임 ×/

총자산 = 0

고객번호 = 1

고객번호 > 표길이 까지 반복

고객구조번호 := 고객구조표(고객번호).

자산 업임(고객 구조번호 : 차변, 대변)호출

총자산 +=(대변 - 차변)

고객번호 +=1

반복 끝

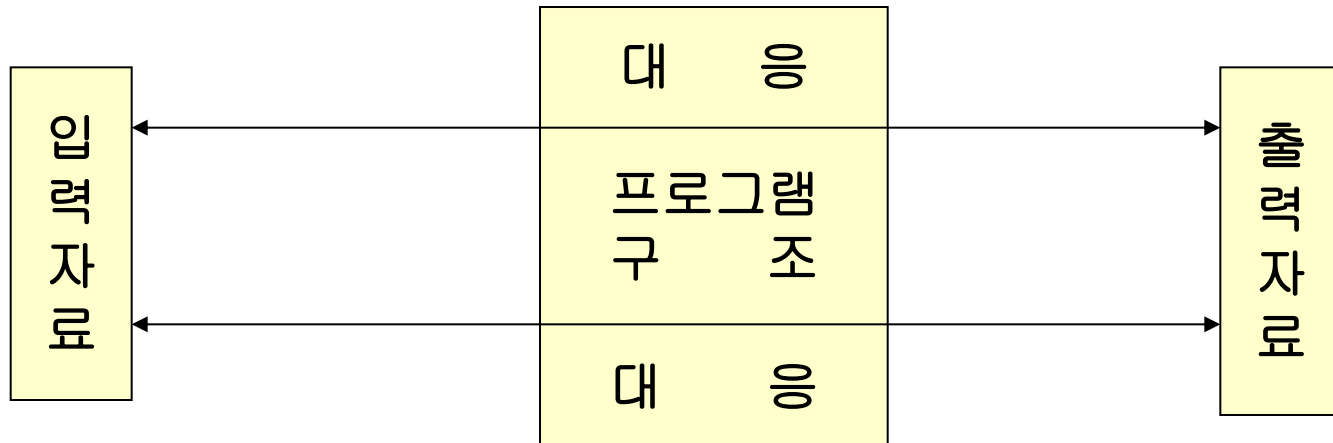
끝 총자산 구함.

자료구조 지향 설계

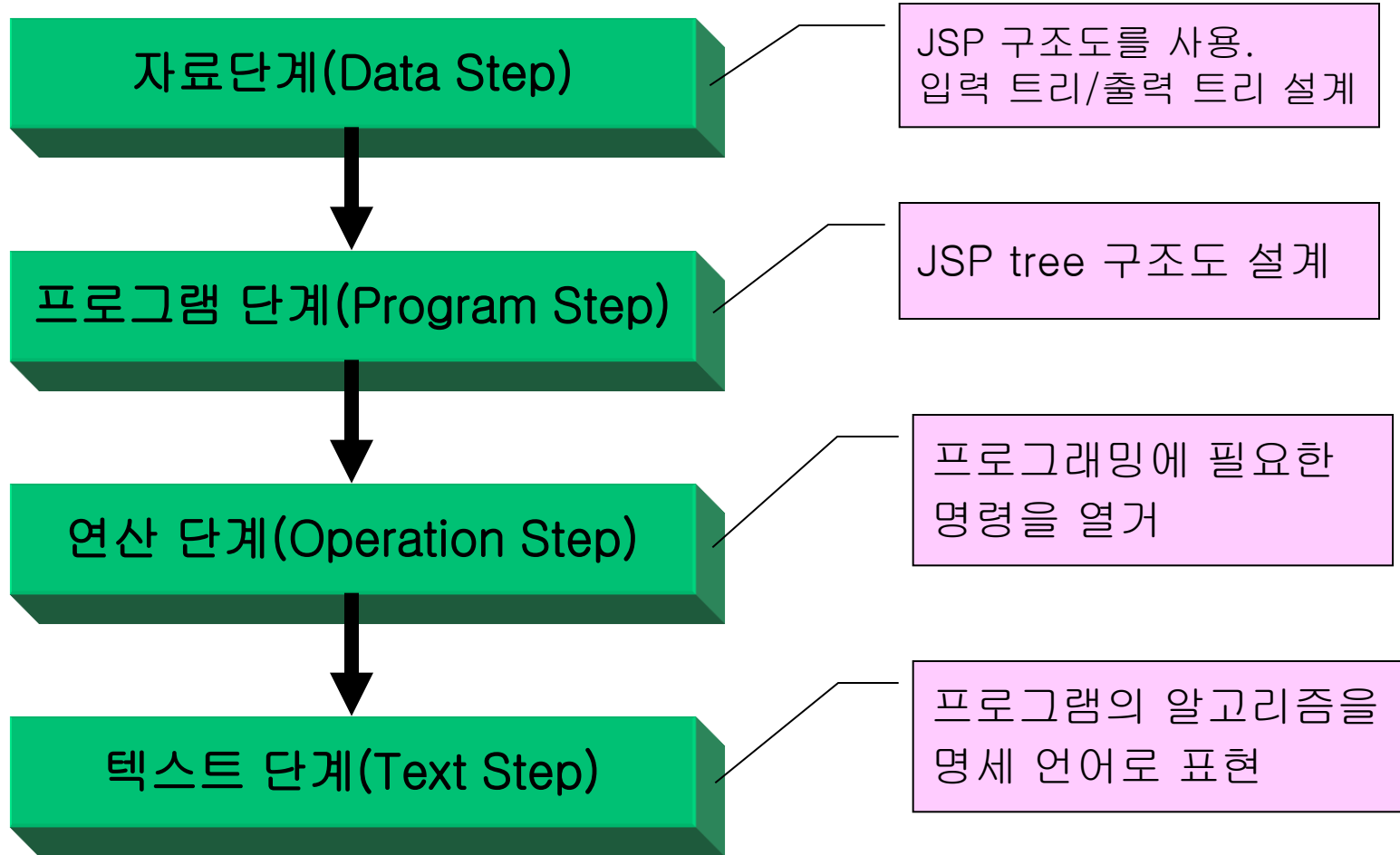
- 입출력자료의 구조 파악으로 소프트웨어 구조를 추출
- 자료는 실세계에 존재하는 각종 실체를 반영하며 특정한 처리와는 독립적인 특성
 - 변화되기 쉬운 처리보다는 상대적으로 안정적인 자료를 중심으로 소프트웨어 설계
 - ⇒ 변화에 크게 영향을 받지 않는 시스템 설계 가능
- 구현 방법
 - JSP(Jackson system of Programming) 설계법
 - Warnier 설계법

JSP 설계법(1)


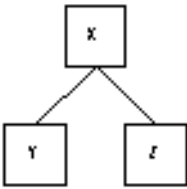
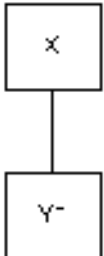
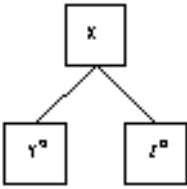
- JSP : Jackson System of Programming
 - 프로그램이 다루는 입출력 자료의 구조를 설계함으로써 프로그램의 제어 구조를 유도하는 방법
 - 입력 자료구조와 출력 자료구조를 각각 설계한 후 서로간의 대응관계가 바로 프로그램의 구조



JSP 설계법(2)-설계순서



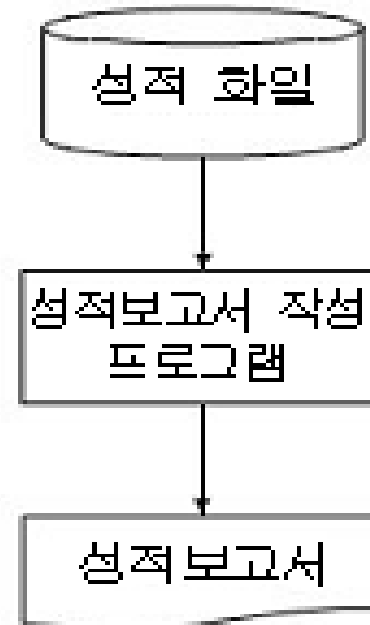
JSP 설계법(3)-JSP 트리 구조도

명칭	구조도	의미	적용분야	
			자료구조	프로그램 구조
기본		더 이상 분할이 불가능한 최소 단위	자료 항목에 대응	명령문에 대응
순차		하나 이상의 구성요소가 순차적으로 처리	레코드에 대응	순차적인 처리에 대응
반복		기본 구성요소가 반복	파일 또는 배열에 대응	루프처리에 대응
선택		기본 구성요소 중 선택	자료의 분류에 대응	선택처리에 대응

JSP 설계법(4)-설계 예

- 요구사항
 - 반별, 학생 별로 기록된 성적 파일을 읽고 성적 보고서를 작성하라.
 - 성적 보고서에는 학생 별 성적의 출력 이외에, 반별 평균 및 총 평균도 출력한다. 또한, 성적 파일의 데이터에는 합격자와 불합격자가 섞여 있다.

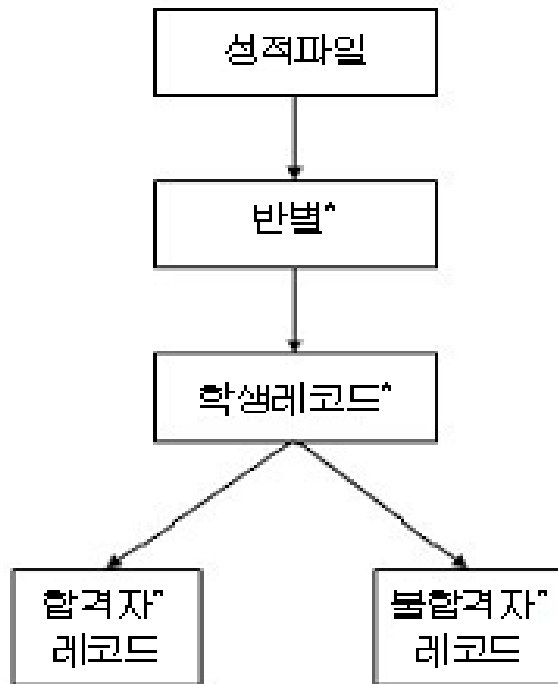
- 입출력 연관도



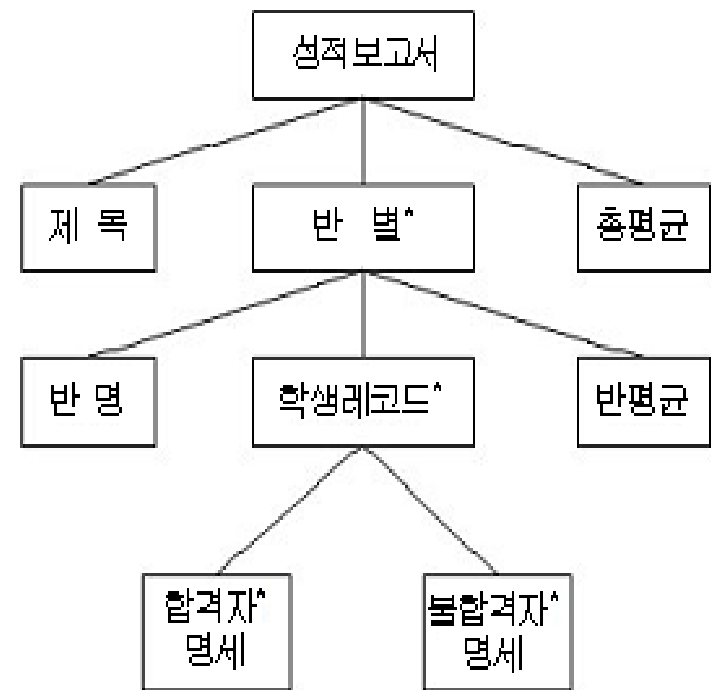
JSP 설계법(5)-설계 예

- 자료 단계

- 입력 트리



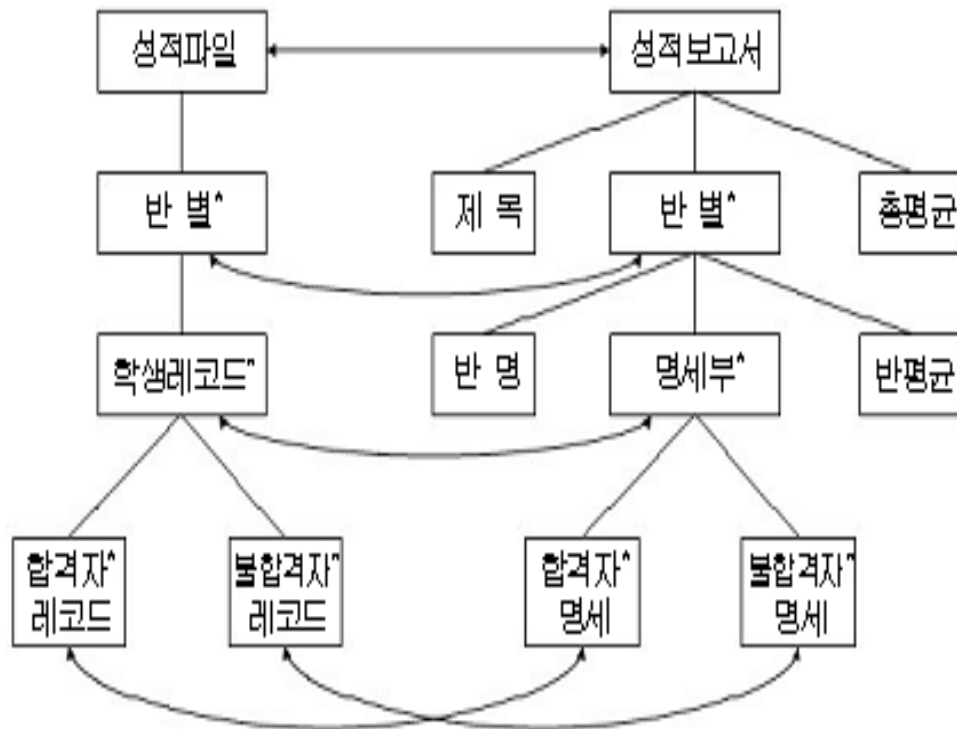
- 출력 트리



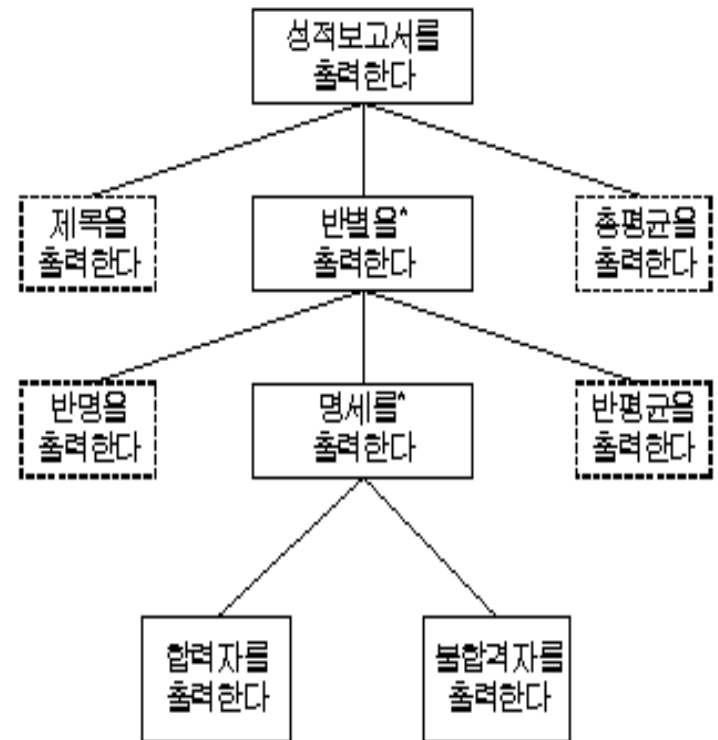
JSP 설계법(6)-설계 예

- 프로그램 단계

- 입/출력 트리의 대응

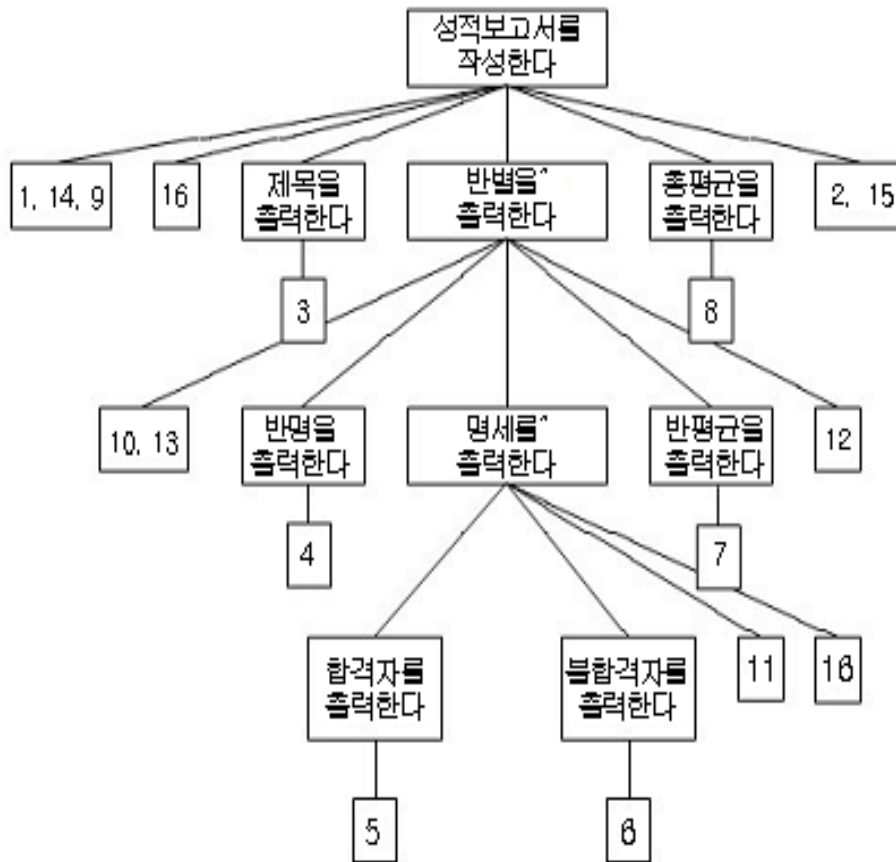


- 프로그램 트리



JSP 설계법(7)-설계 예

• 연산 단계



- 1) 성적 보고서 file을 출력 file로 개방
- 2) 성적 보고서 file의 폐쇄
- 3) 성적 보고서의 제목 출력
- 4) 성적 보고서의 학반 명 출력
- 5) 성적 보고서의 명세(합격자) 출력
- 6) 성적 보고서의 명세(불합격자) 출력
- 7) 성적 보고서의 반 평균 출력
- 8) 성적 보고서의 총 평균 출력
- 9) 모든 합계변수 초기화
- 10) 반별 합계 초기화
- 11) 과목별 점수 및 인원수를 반별 합계에 누적
- 12) 반별 합계를 총 합계에 누적
- 13) 비교를 위한 반 코드를 저장
- 14) 성적 file을 입력 file로 개방
- 15) 성적 file의 폐쇄
- 16) 성적 file에서 읽음

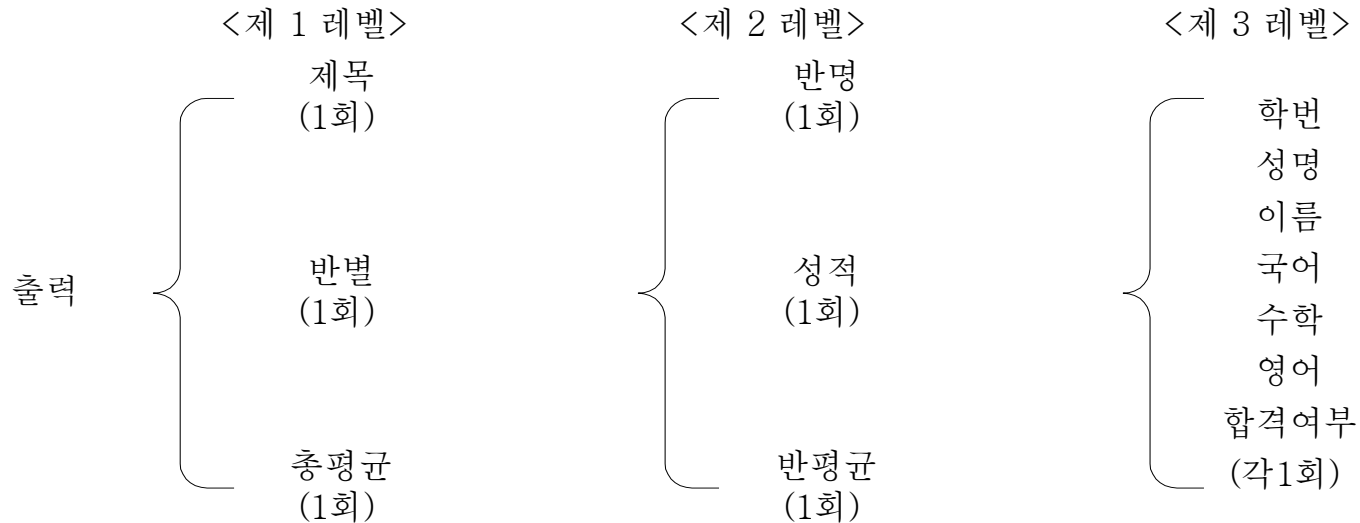
JSP 설계법(8)-설계 예

- Text 단계

seq	· 성적보고서를 작성
do 1, 14, 9	· <u>입출력파일</u> 개방 및 초기화
do 16	· <u>성적파일</u> 읽음
seq do 3 end	· 제목을 출력
<u>itr</u> until(EOF)	· 반별을 출력
do 10, 13	· 반별합계 지움, 반별합계를 총합계에 누적
seq do 4 end	· <u>반명을</u> 출력
<u>itr</u> until(<u>반코드=저장한 반코드</u>)	· 명세를 출력
<u>sel</u> (합격여부=0)	· 합격자를 출력
seq do 5 end	· 불합격자를 출력
alt(합격여부=1)	
seq do 6 end	
end	
do 11	· 과목별 점수 및 인원수 반별합계에 누적
do 16	· 성적파일 읽음
end	
seq do 7 end	· 반별 평균을 출력
do 12	· 반별합계를 총합계에 누적
end	
seq do 8 end	· 총평균을 출력
do 2, 15	· 입출력파일 폐쇄
end	

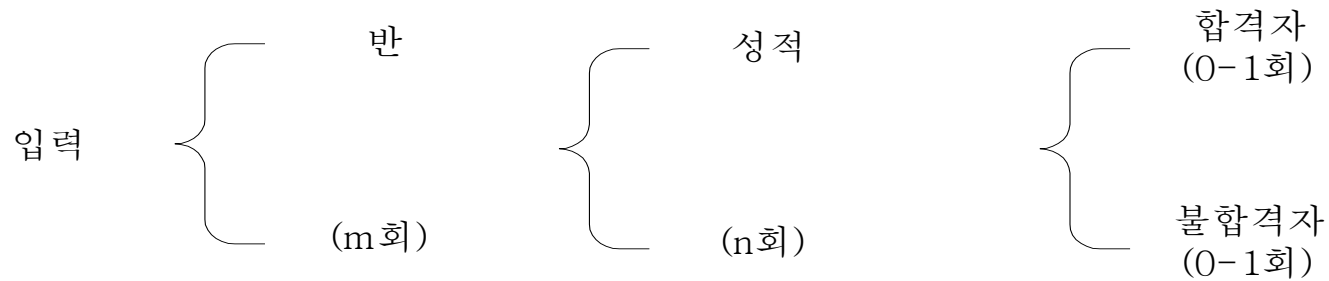
Warnier 설계법(1)

- 자료구조로부터 프로그램 구조를 유도하는 방법
 - 프로그램의 구조는 취급하는 자료의 구조에 의해 결정
- 작업 순서
 - 출력자료 구조의 설계

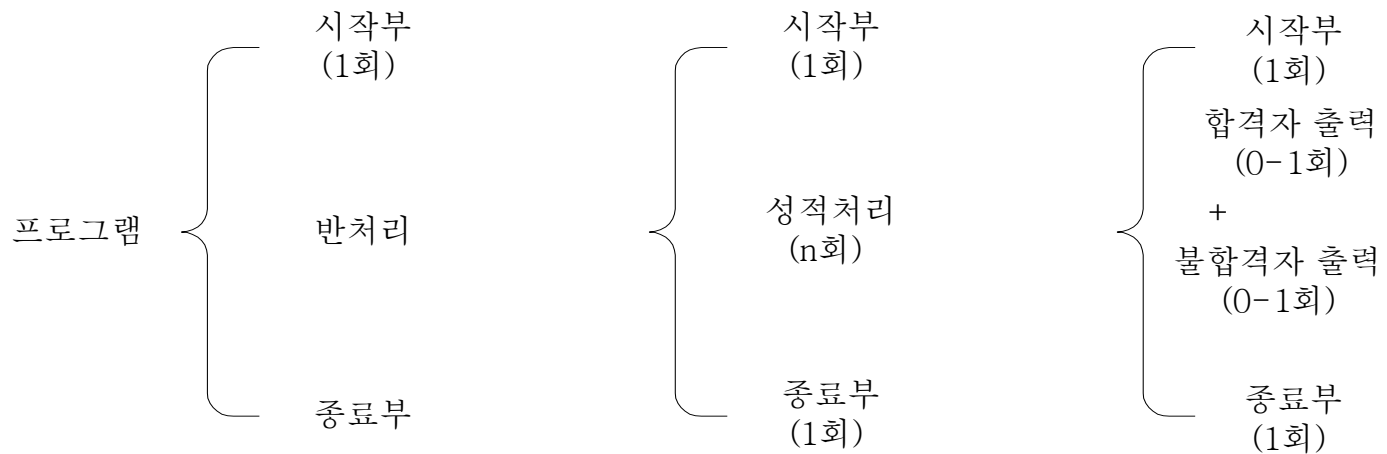


Warnier 설계법(2)

입력자료 구조의 설계

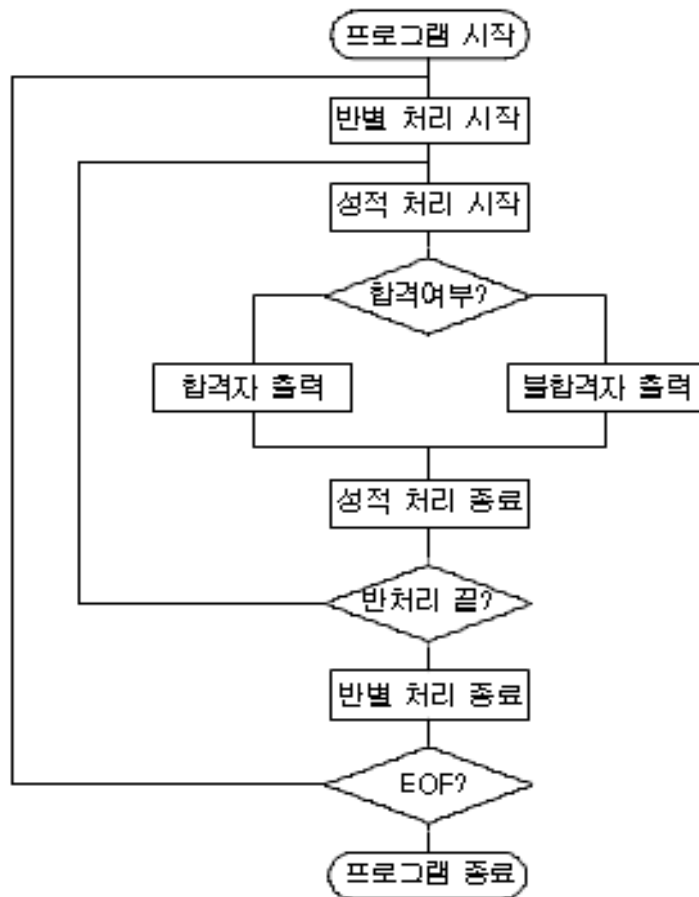


프로그램 구조도의 설계



Warnier 설계법(3)

구조적 흐름의 설계



절차명령의 설계

구분	명령	해당위치
입력 처리	첫 성적레코드의 읽음 성적레코드의 읽음	프로그램 시작부 반별 처리 종료
반복 처리	성적화일이 끝날때까지 반모드가 다를때까지 반코드의 저장	반별 처리 시작부터 반별 처리 종료까지 성적 처리 시작부터 성적처리 종료까지 반별처리 시작
선택 처리	합격 여부 판정	성적처리 시작
연산 처리	성적점수 및 인원수를 반별 합계에 누적 반별 합계를 총합계에 누적 반별 합계를 지움	성적 처리 종료 반별 처리 종료 반별 처리 시작
출력 처리	제목 출력 반명 출력 합격자 출력 불합격자 출력 반평균 출력 총평균 출력	프로그램 시작 반별 처리 시작 합격자 출력 불합격자 출력 반별 처리 종료 프로그램 종료

Warnier 설계법(4)

- Program 상세사양의 설계

<pre> 프로그램 <프로그램 시작> 성적파일 및 성적 보고서 파일 개방 제목 출력 첫 성적레코드를 읽음 반별 처리의 반복 실행(성적확일이 끝날때까지) <프로그램 종료> 총평균 출력 성적파일 및 성적 보고서 파일 폐쇄 STOP RUN </pre>
<pre> 반별처리 <반별처리 시작> 반별합계를 지움 반코드의 저장 반명 출력 성적처리의 반복실행(반코드가 달라질 때 까지) <반별처리 종료> 반평균 출력 반별 합계를 총합계에 누적 EXIT </pre>
<pre> 성적처리 <성적처리 시작> 합격여부 = '0'이면 합격자 출력 합격여부 = '1'이면 불합격자 출력 <성적처리 종료> 과목별 성적 및 인원수를 반별합계에 누적 성적레코드를 읽음 EXIT </pre>

재사용성

- 여러 수준에서 고려될 수 있는 재사용
 - 응용 시스템 재사용
 - 부시스템 재사용
 - 모듈 혹은 객체의 재사용
 - 함수 재사용
- Focus
 - 재사용을 이용한 소프트웨어 개발
 - 재사용을 위한 소프트웨어 개발
 - 생성기에 기반한 재사용
 - 응용 시스템의 재사용