5주 1강

소프트웨어 상위 설계

이번 주차에는…

소프트웨어 상위 설계

- 설계의 이해
- 설계의 원리
- 소프트웨어 아키텍처
- 디자인 패턴

1. 설계의 이해

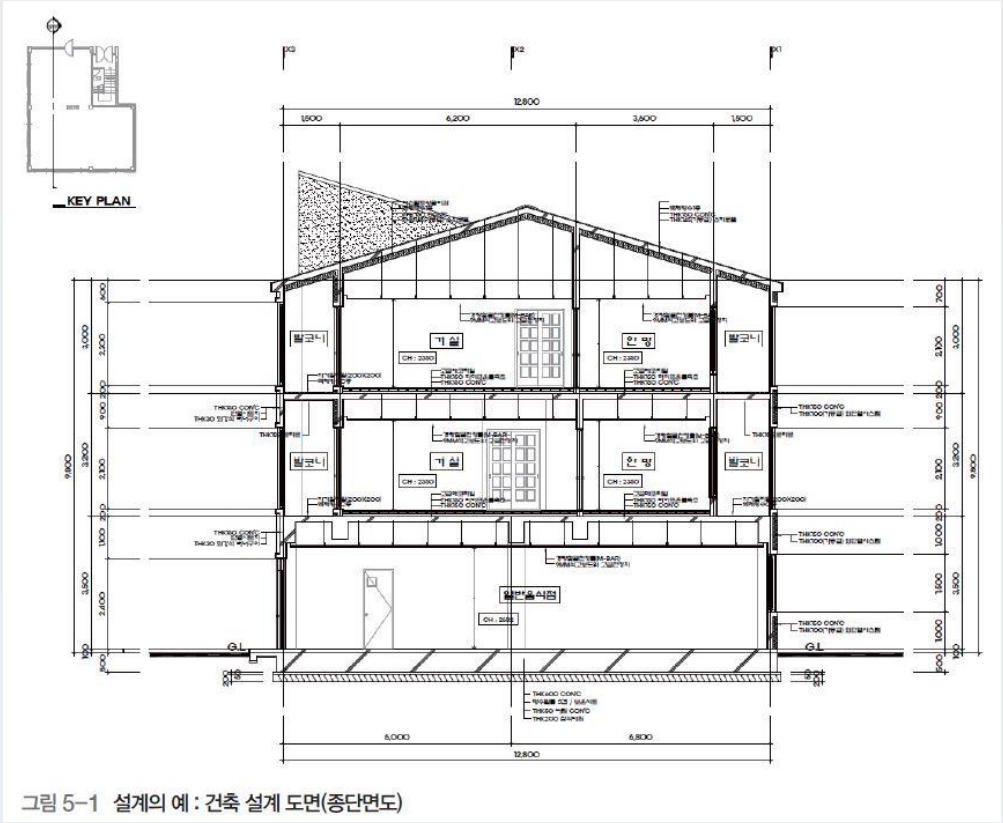


그림 5-1 설계의 예 : 건축 설계 도면(종단면도)

2. 설계의 예 : 옷본

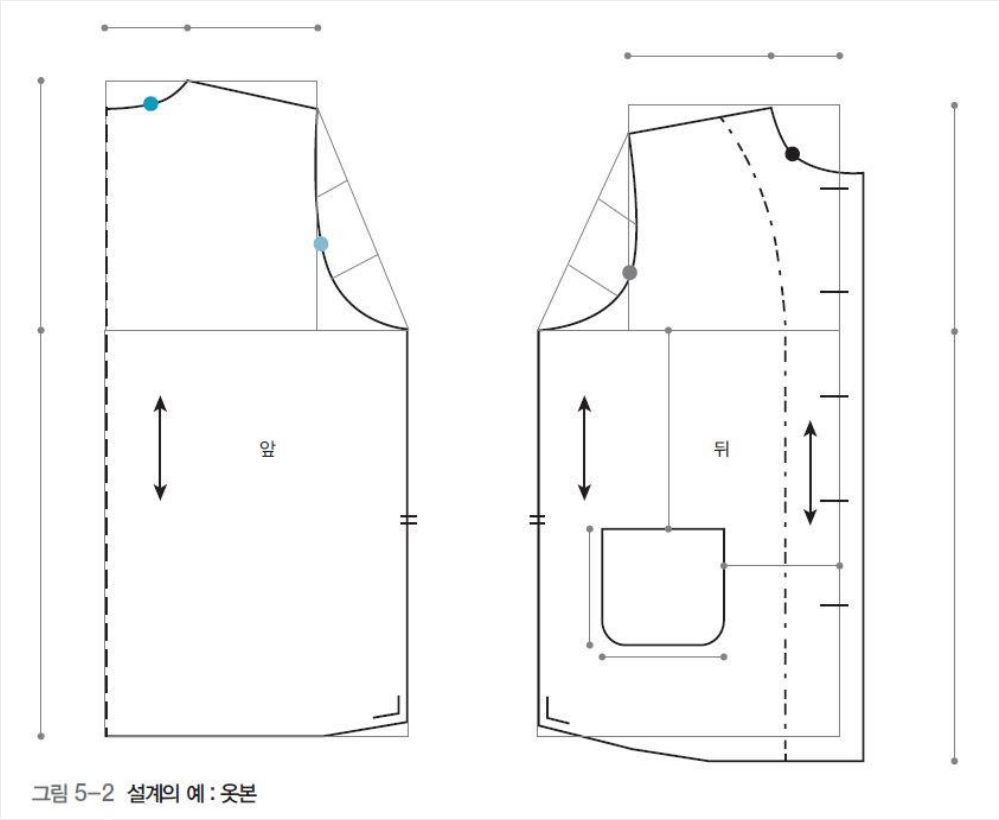


그림 5-2 **설계의 예 : 옷본**

3. 건축 설계 과정

그림 5-3 건축 설계 과정

4. 소프트웨어 설계

- 분석 단계
  - 사용자의 요구 사항을 토대로 요구 분석 명세서 작성
  - 개념적이고 추상적, what(무엇) 관점
- 설계 단계
  - 분석 단계에서 파악한 비기능적 요구 사항과 제약 사항 고려
  - 운영체제·미들웨어·프레임워크 등의 플랫폼 결정, how(어떻게) 관점
- 설계
  - 요구 분석 명세서를 기반으로 어떻게 구축할 것인가를 결정하는 것
  - 설계자: 다양한 제약 조건을 만족시킬 수 있는 최적의 설계안을 만드는 것이 중요
  - 설계를 평가할 수 있는 기준도 정량적으로 명시

5. 좋은 설계의 조건

- 요구 분석 명세서의 내용을 설계서에 모두 포함해야 한다.
- 유지보수가 용이하도록 추적이 가능해야 한다.
- 변화에 쉽게 적응할 수 있어야 한다.
- 시스템 변경으로 인한 영향이 최소화되도록 국지적이어야 한다.
- 설계서는 읽기 쉽고, 이해하기 쉽게 작성해야 한다.

6. 설계의 종류(1)

그림 5-4 상위 설계와 하위 설계

7. 설계의 종류(2)

- 상위 설계(예비 설계preliminary design)
  - 아키텍처(구조) 설계 : 시스템의 전체적인 구조
  - 데이터 설계 : 시스템에 필요한 정보를 자료구조와 데이터베이스 설계에 반영
  - 시스템 분할 : 전체 시스템을 여러 개의 서브시스템으로 나눈다.
  - 인터페이스 정의 : 시스템의 구조와 서브시스템들 사이의 인터페이스가 명확히 정의
  - UI 설계 : 사용자가 익숙하고 편리하게 사용할 수 있도록 사용자 인터페이스설계
- 하위 설계
  - 각 모듈의 실제적인 내부를 알고리즘(pseudo-code) 형태로 표현
  - 인터페이스에 대한 설명, 자료구조, 변수 등에 대한 상세한 정보를 작성

8. 설계의 원리

그림 5-5 **설계의 원리**

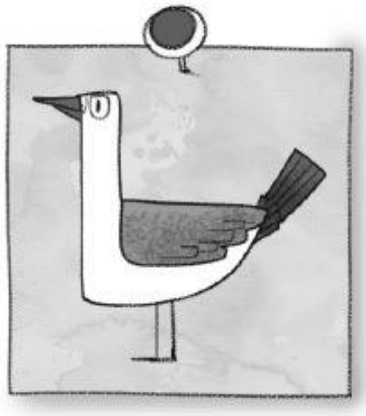
9. 분할과 정복의 원리

- 분할과 정복
  - ✓ 큰 문제를 소 단위로 나누고 소 단위의 작업을 하나씩 처리하여 전체 일을 끝내는 방법

    (예) 대학의 종합정보시스템: 학사 관리, 회계 관리, 인사 관리 등으로 나눔

    학사 관리: 수강 관리, 수업 관리, 성적 관리 등으로 나눔
- 분할 형태
  - ✓ 분산 시스템은 클라이언트와 서버로 분할
  - ✓ 시스템은 여러 서브시스템으로 분할
  - ✓ 서브시스템은 하나 이상의 패키지로 분할
  - ✓ 패키지는 여러 클래스로 분할
  - ✓ 클래스는 여러 메서드로 분할

10. 추상화의 원리



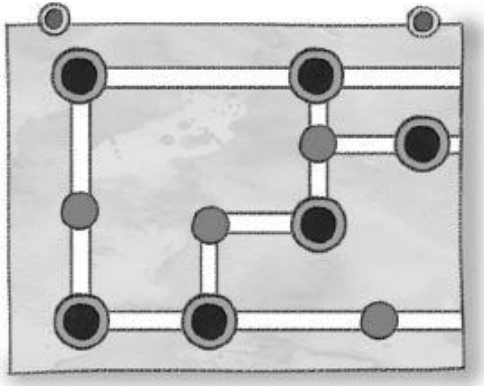


그림 5-6 추상화 표현의 예

- 추상화abstraction
  - 주어진 문제(건물 도면)에서 현재의 관심사에 초점을 맞추기 위해, 특정한 목적과 관련된 필수 정보만 추출하여 강조하고(전기 배선도, 상하수도 배관도 등) 관련이 없는 세부 사항을 생략함으로써 본질적인 문제에 집중할 수 있도록 하는 작업

11. 도형의 추상화

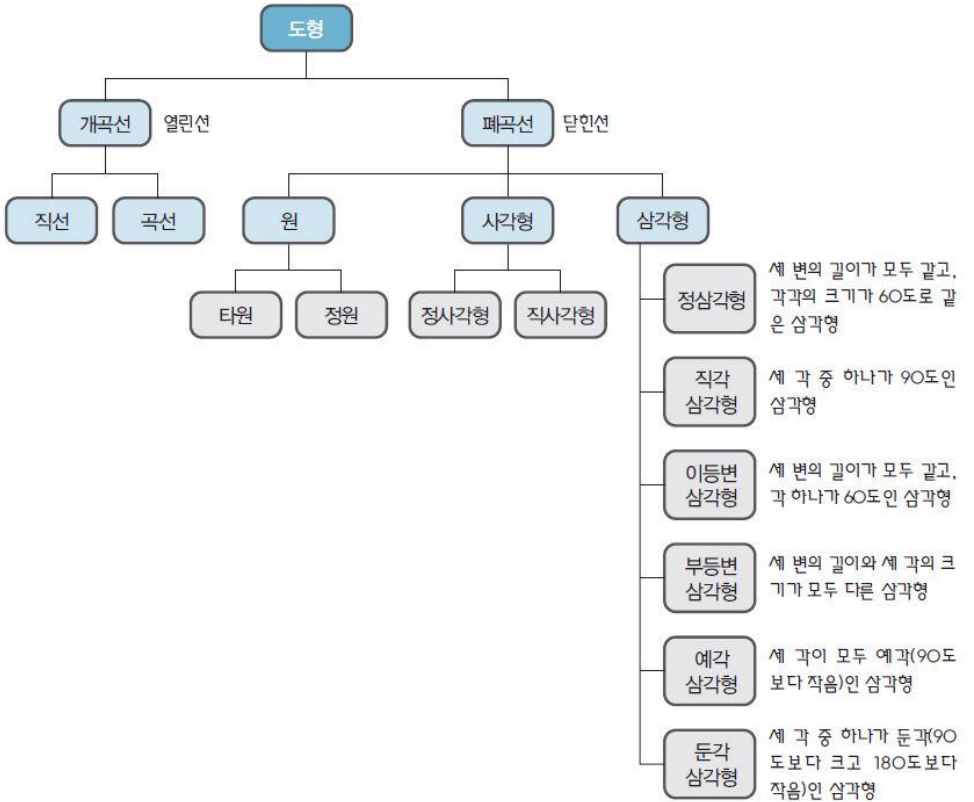


그림 5-7 **도형의 추상화**

12. 객체지향 설계에서의 추상화

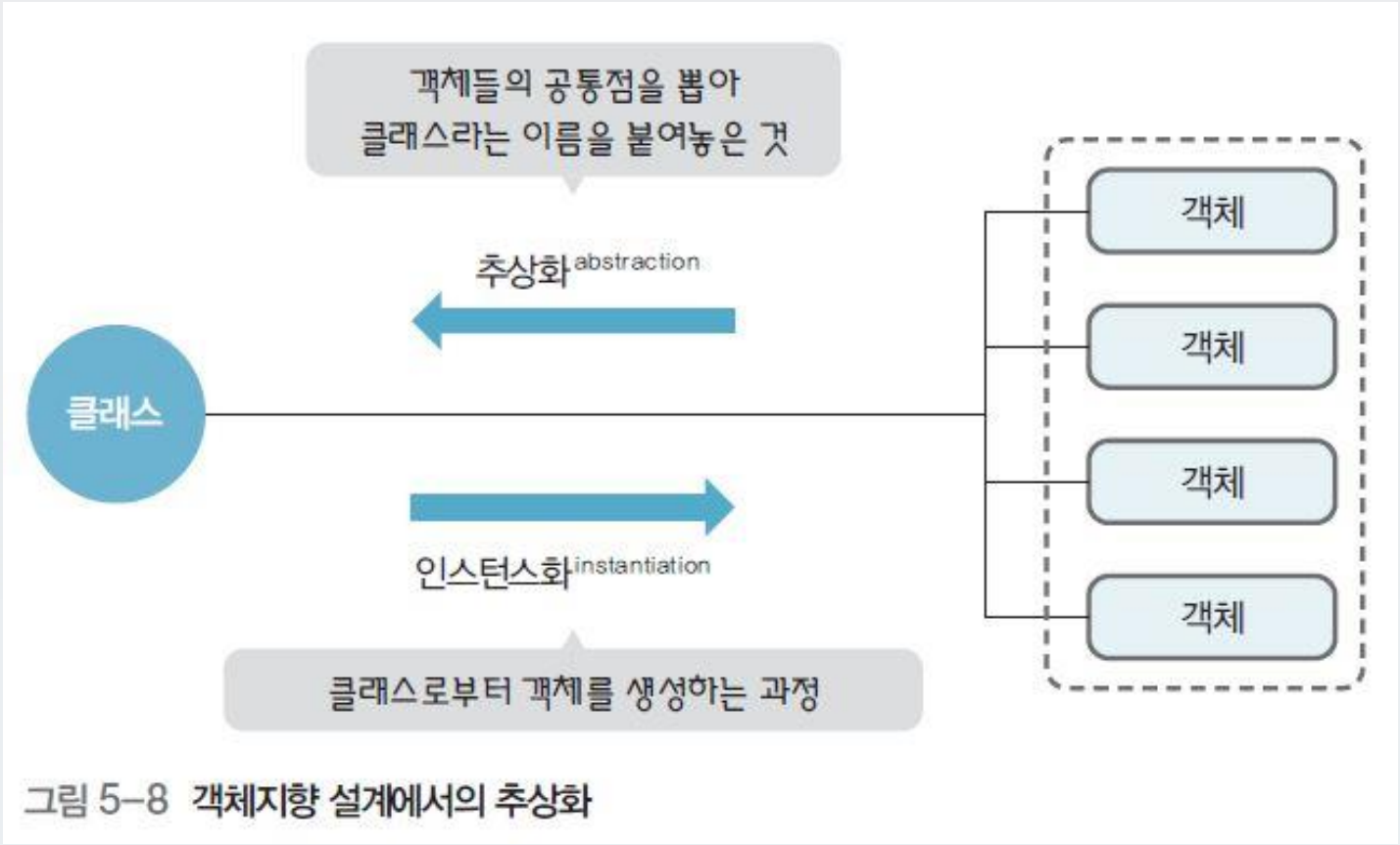


그림 5-8 객체지향 설계에서의 추상화

13. 추상화의 종류

- 과정 추상화Procedure abstraction

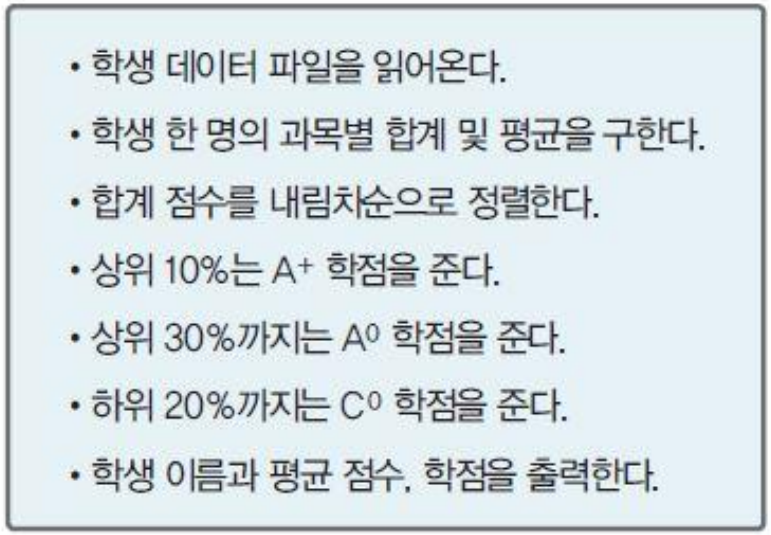


그림 5-9 과정 추상화의 예 1 : 학점 계산 알고리즘

```
int main(void) {
   int result;

   result = GetSum(10);
   printf("1~10의 합계 = %d\n", result);
   return 0;
}

int GetSum(int num) {
   int i;
   int sum = 0;
   for(i=1; i<=num; i++)
      sum += i;
   return sum;
}
```

그림 5-10 과정 추상화의 예 2 : 함수 호출

14. 추상화의 종류(2)

- Class의 특징
  - 사용자에게 클래스가 제공할 수 있는 사용법만 알려주고, 불필요한 데이터와 연산을 감춤
  - 사용자는 클래스에서 제공하는 연산 기능만 알고 그 연산을 사용하여 데이터 값을 변경
- 데이터 추상화data abstraction
  - 대표적인 예: C++ 언어의 클래스

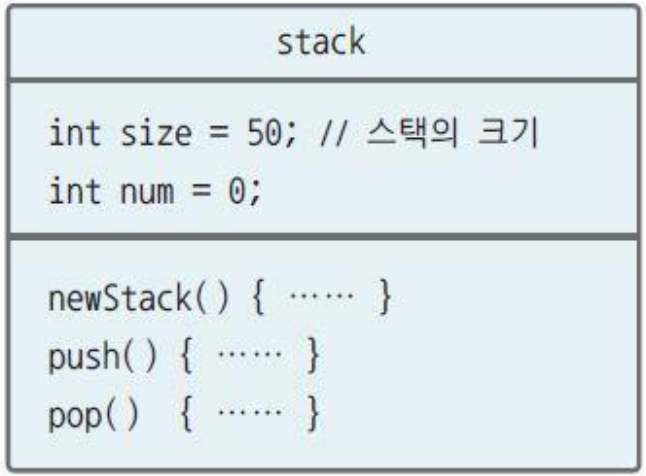


그림 5-11 데이터 추상화의 예

15. 추상화의 종류(3)

- 제어 추상화control abstraction

`Z = X + Y;`	`LOAD A, X`
	`ADD A, Y`
	`STORE Z, A`
		`00100101`
		`10000110`
		`01000111`

(a) 고급 언어 표현	(b) 어셈블리 언어 표현	(c) 기계 언어 표현

- LOAD A, X // X번지의 내용을 읽어 레지스터 A에 적재
- ADD A, Y // Y번지의 내용을 A에 적재된 값과 더한 결과를 레지스터 A에 적재
- STORE Z, A // A 레지스터의 값을 Z번지에 저장

그림 5-12 제어 추상화의 예

16. 단계적 분해

- 단계적 분해
  - 하향식 설계에서 사용
  - 기능을 점점 작은 단위로 나누어 점차적으로 구체화하는 방법

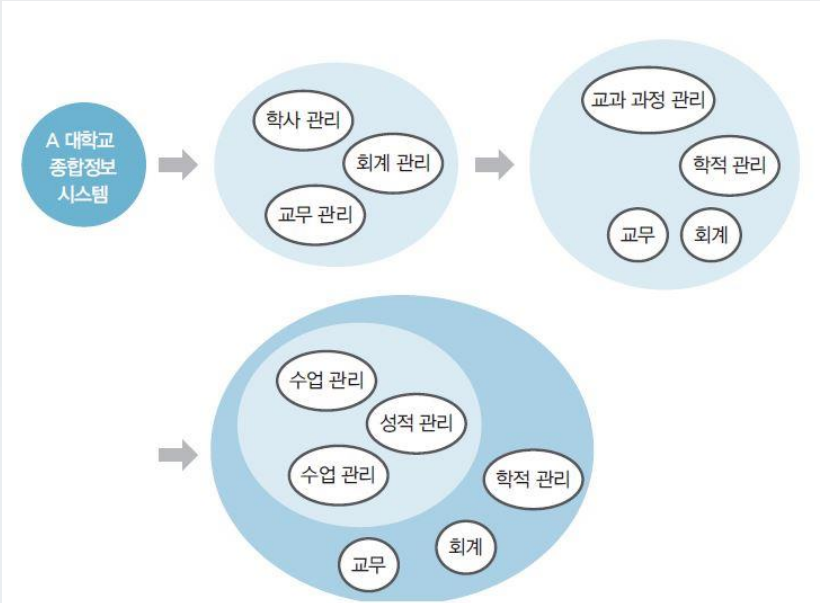


그림 5-13 DFD로 표현한 단계적 분해의 예

17. 모듈화

- 모듈
  - ‘규모가 큰 것을 여러 개로 나눈 조각’
  - ‘소프트웨어 구조를 이루는 기본적인 단위’
  - ‘하나 또는 몇 개의 논리적인 기능을 수행하기 위한 명령어들의 집합’
  - 라이브러리 함수, 그래픽 함수
  - 라이브러리 함수, 그래픽 함수, 추상화된 자료, subroutine, procedure, object, method
  - → 독립 프로그램도 하나의 모듈로 가능, 함수들도 하나의 모듈로 가능
- 모듈의 특징
  - 다른 것들과 구별될 수 있는 독립적인 기능을 갖는 단위이다.
  - 유일한 이름을 가져야 한다.
  - 독립적으로 컴파일이 가능하다.
  - 모듈에서 또 다른 모듈을 호출할 수 있다.
  - 다른 프로그램에서도 모듈을 호출할 수 있다.

다음 시간

소프트웨어 아키텍처