3주 3강

일정 계획, 위험 분석

이번 주차에는…

소프트웨어 개발 계획

- 일정 계획
- 위험 분석
- 객체지향이란?
- 클래스와 객체
- 인스턴스 변수, 클래스 변수, 메소드, 오퍼레이션
- 상속
- 다형성

1. 일정 계획

그림 3-10 프로젝트 일정 계획의 예

2. SW 개발 프로젝트에서의 일정 계획

- 일정 계획
  - 작업 순서 결정, 소 작업의 개발 기간, 순서, 필요한 자원 등의 일정 계획

표 3-16 일정 계획

총 개발 기간	6개월(○○○○. 1. 1. ~ ○○○○. 6. 31.)
소작업	교과 과정 관리, 수업 관리, 수강 관리, 성적 관리, 사용자 정보 관리
소작업별 개발 기간	소작업 / 개발 기간 사용자 정보 관리 / 1개월(1. 1. ~ 1. 31.) 교과 과정 관리 / 1개월(2. 1. ~ 2. 28.) 수업 관리 / 1개월(3. 1. ~ 3. 31.) 수강 관리 / 1개월(4. 1. ~ 4. 30.) 성적 관리 / 1개월(5. 1. ~ 5. 31.) 테스트 / 1개월(6. 1. ~ 6. 30.)
개발 순서	사용자 정보 관리 → 교과 과정 관리 → 수업 관리 → 수강 관리 → 성적 관리
필요 자원	개발 툴(파워빌더), 개발용 PC, 개발 공간

3. 작업 분할 구조도(1)

- 작업 분할 구조도(WBS: Work Breakdown Structure)
  - 프로젝트 목표를 달성하기 위해 필요한 활동과 업무를 세분화하는 작업
  - 프로젝트 구성 요소들을 계층 구조로 분류
  - 프로젝트의 전체 범위 정의
  - 프로젝트 작업을 세분화

- 작업 패키지(work package)
  - 계층 구조에서 최하위에 있는 항목

4. 작업 분할 구조도(2)

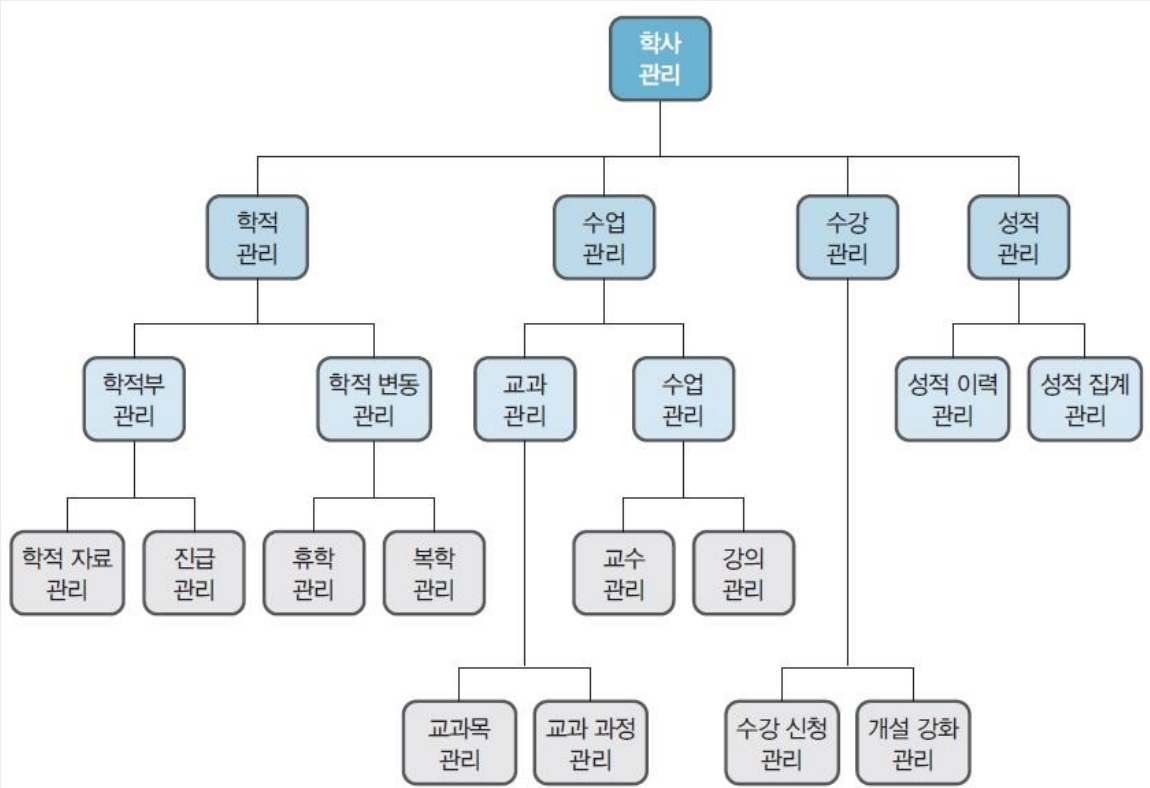


그림 3-11 학사 관리 시스템의 WBS

5. 작업 분할 구조도(3)

- 작업 분할 구조도의 목적과 용도
  - 사용자와 개발자 간의 의사소통 도구로 사용함.
  - 프로젝트 업무 내역을 가시화할 수 있어 관리가 용이함.
  - 프로젝트 팀원의 책임과 역할이 분명함.
  - 필요 인력과 일정 계획을 세우는 데 기초로 활용함.
  - 개발비 산정 시 기초로 활용함.
  - 성과 측정 및 조정 시 기준선으로 활용할 수 있음.

6. 네크워크 차트

- PERT/CPM
  - WBS의 작업 순서, 소요 기간 등을 네트워크 형태의 그래프로 표현한 후 어떤 작업이 중요한지, 또 일정에 여유가 있는 작업은 어떤 것이지 찾아내 중점 관리를 해야 하는 작업을 명확히 하는데 사용
  - 프로젝트를 완료할 수 있는 최소 기간은 얼마인지, 완료 기간을 맞추기 위해서는 각 작업을 언제 시작하고 완료해야 하는지, 지연되지 않으려면 어떤 작업에 특히 주의를 기울여야 하는지, 또 전체 프로젝트 완료 기간을 단축하기 위해서는 어떤 작업들을 단축하는 것이 가장 경제적인지 등 관리자의 고민에 답을 주기 위해 필요한 도구
  - PERT
    - 1958년 미 해군 탄도 미사일 개발 과정에서 사용
    - 각 이벤트 중심의 일정 관리
  - CPM
    - 1957년 랜드사의 캘리와 듀퐁사의 워커
    - 각 활동 시간의 확정적 추정(과거의 경험기반)

7. 학사 관리 애플리케이션에 대한 CPM의 예

표 3-17 CPM 네트워크를 위한 활동 목록

작업	작업 설명	선행 작업	소요 기간(주)
A	개인 정보 등록/수정/조회/삭제 프로그램 개발	–	2
B	학적 변동 자료 등록/수정/조회/삭제 프로그램 개발	A	6
C	휴 · 복학 및 자퇴 등록/수정/조회/삭제 프로그램 개발	A	4
D	교육 과정 등록/수정/조회/삭제 프로그램 개발	B, C	2
E	유사/동일 과목 등록/수정/조회/삭제 프로그램 개발	D	4
F	개설 강좌 등록/수정/조회/삭제 프로그램 개발	D	3
G	수강 과목 등록/수정/조회/삭제 프로그램 개발	E	2
H	시간표 등록/수정/조회/삭제 프로그램 개발	E	4
I	성적 등록/수정/조회/삭제 프로그램 개발	F	2
J	장학생 등록/수정/조회/삭제 프로그램 개발	F	1
K	등록금 등록/수정/조회/삭제 프로그램 개발	G, H	2
L	졸업 사정 등록/수정/조회/삭제 프로그램 개발	I, K	2
M	사회봉사 실적 등록/수정/조회/삭제 프로그램 개발	J, L	3

8. CPM 네트워크를 그린다

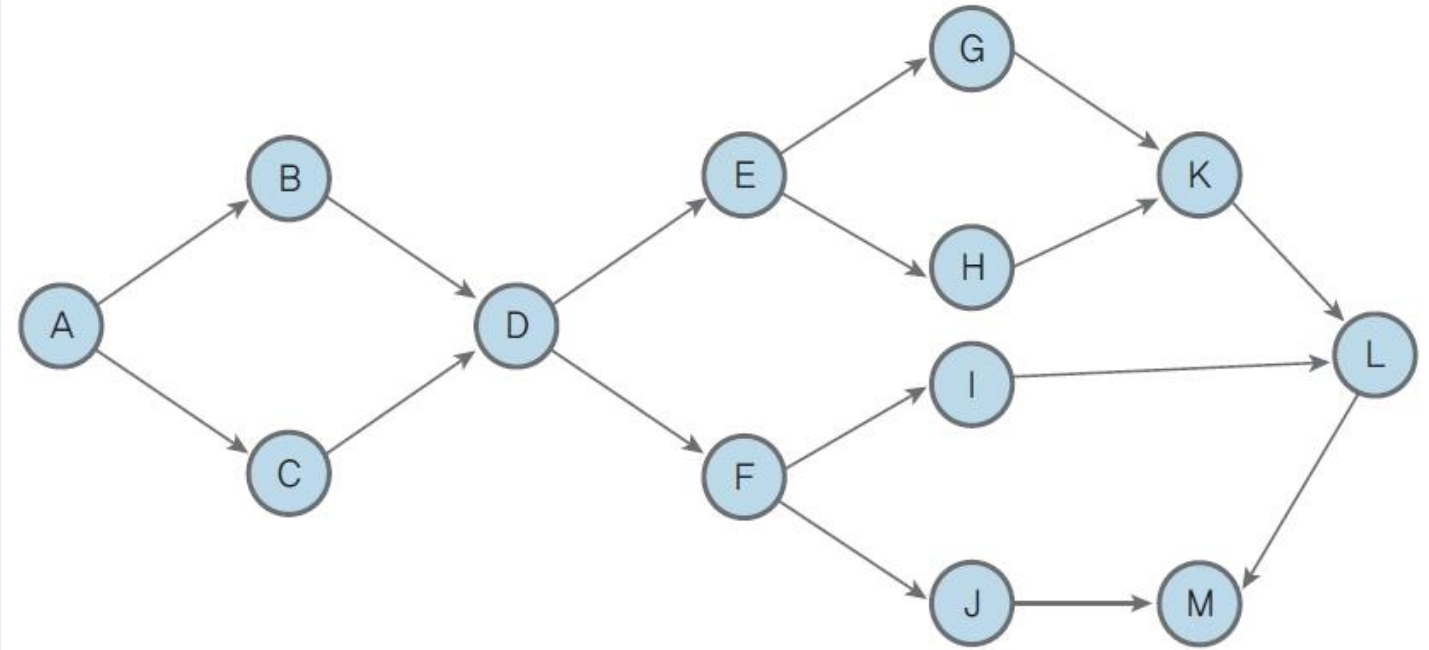


그림 3-12 CPM 네트워크

9. ES 값을 구한다

- ES Earliest Start Time
  - 가능한 빨리 시작할 수 있는 시간으로, 선행 작업이 완료되었을 때 해당 작업을 시작할 수 있는 가장 빠른 시점
  - ES 값을 구할 때는 맨 앞(작업 A)에서 끝 방향으로 가며 계산

표 3-18 작업별 가장 빨리 시작할 수 있는 시간 (단위 : 주)

작업	A	B	C	D	E	F	G	H	I	J	K	L	M
작업 시작 시간	0	2	2	8	10	10	14	14	13	13	18	20	22
작업 시간	2	6	4	2	4	3	2	4	2	1	2	2	3

10. EF값을 구한다

- EF Earliest Finish Time
  - 가장 빠른 시작 시간(ES)으로 시작했을 때의 가장 빠른 완료 시간
  - (즉) 가능한 빨리 끝낼 수 있는 시간으로 'ES+작업 소요 시간'

표 3-19 작업별 가장 빨리 완료할 수 있는 시간 (단위 : 주)

작업	A	B	C	D	E	F	G	H	I	J	K	L	M
EF	2	8	6	10	14	13	16	18	15	14	20	22	25

11. ES값을 구한다

- LS Latest Start Time
  - 어떤 작업을 늦어도 시작해야 하는 시간, 즉 가장 늦게 시작할 수 있는 시간
  - 이 시간에 시작하지 않으면(이 시간보다 늦게 시작하면) 총 일정이 지연

표 3-20 작업별 가장 늦게 시작할 수 있는 시간 (단위 : 주)

작업	A	B	C	D	E	F	G	H	I	J	K	L	M
작업 시작 시간	0	2	4	8	10	15	16	14	18	21	18	20	22
작업 시간	2	6	4	2	4	3	2	4	2	1	2	2	3

12. LF값을 구한다

- LF Latest Finish Time
  - 가장 늦게 시작할 수 있는 시간(LS)에 시작해 작업을 완료한 시간
  - (즉) 작업을 가장 늦게 끝낼 수 있는 시간으로 'LS+작업 소요 시간'

표 3-21 작업별 가장 늦게 시작했을 때 완료할 수 있는 시간 (단위 : 주)

작업	A	B	C	D	E	F	G	H	I	J	K	L	M
작업 시작 시간	0	2	4	8	10	15	16	14	18	21	18	20	22
작업 시간	2	6	4	2	4	3	2	4	2	1	2	2	3
작업 완료 시간	2	8	8	10	14	18	18	18	20	22	20	22	25

13. ST값을 구한다

- ST Slack Time
  - '느슨한 시간', '여유 있는 시간'

표 3-22 여유 시간 (단위 : 주)

작업	A	B	C	D	E	F	G	H	I	J	K	L	M
작업별 빠른 시작 시간	0	2	2	8	10	10	14	14	13	13	18	20	22
작업별 늦은 시작 시간	0	2	4	8	10	15	16	14	18	21	18	20	22
여유 시간	0	0	2	0	0	5	2	0	5	8	0	0	0

14. 임계 경로를 구한다

- 임계 경로critical path

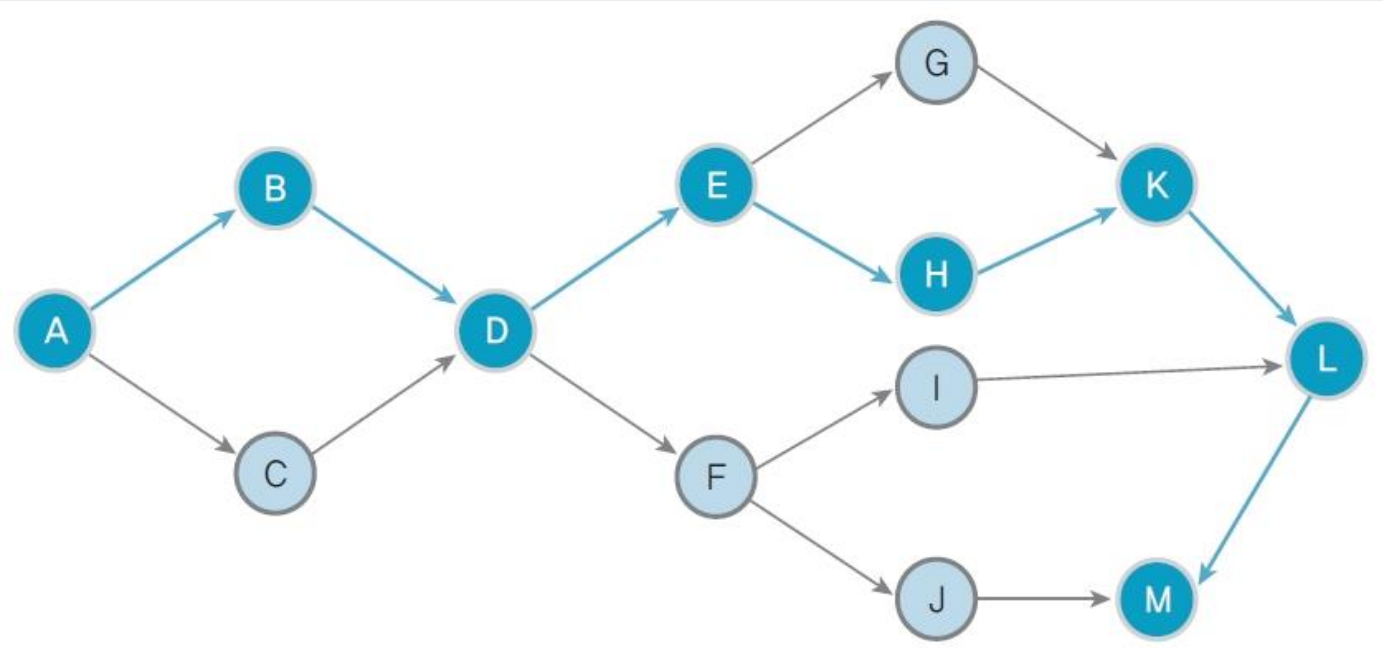


그림 3-13 CPM 네트워크와 임계 경로

15. 간트 차트를 이용한 일정표 작성

- 간트 차트Gantt chart
  - 프로젝트 일정 관리를 위한 바bar 형태의 도구

ID	활동명	시작일	완료일	기간	2016년 1	2	3	4	5	6	7	8	9	10	11	12
1	학적부 관리	16. 01. 01.	16. 02. 28.	2												
2	학생 변동 관리	16. 01. 01.	16. 01. 31.	1												
3	교과 관리	16. 03. 01.	16. 05. 31.	3												
4	수업 관리	16. 06. 01.	16. 07. 31.	2												
5	수강관리	16. 07. 01.	16. 09. 30.	3												
6	성적 관리	16. 10. 01.	16. 12. 31.	3												

계획된 작업량 현재까지의 작업량

그림 3-14 간트 차트를 이용한 프로젝트 일정 계획표

16. 위험 분석

그림 3-15 위험 예방

그림 3-16 도구를 사용한 위험 예방

17. 위험 예방

그림 3-17 위험 예방을 위한 도구 준비

18. 위험 관리 절차(1)

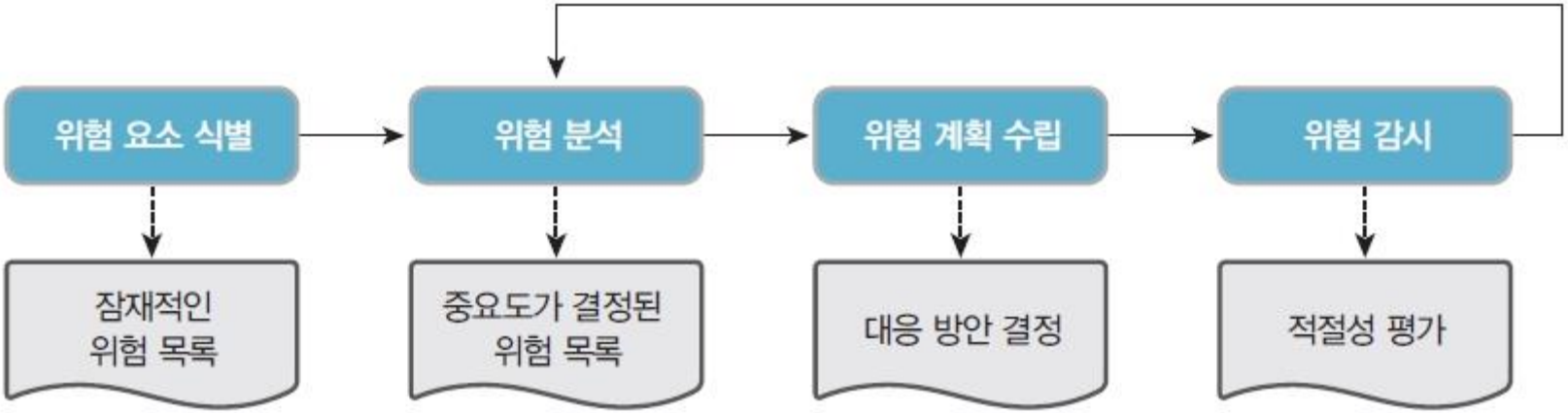


그림 3-18 위험 관리 절차

19. 위험 관리 절차(2)

- 위험 요소 식별

표 3-23 프로젝트 수행 시 발생할 수 있는 위험 요소

위험 요소	위험 내용
개발자의 이직	프로젝트 수행 중 개발자들이 이직한다.
요구 사항 변경	요구 사항 확정 이후에도 변경 요구가 계속된다.
발주사의 재정적 어려움	프로젝트 수행 중 고객사에 경제적인 어려움이 발생한다.
예상을 빗나간 투입 인력	처음에 예측한 인력보다 인력이 더 많이 필요하다.
개발 기간 부족	처음에 예측한 개발 기간을 초과한다.
개발비 초과	처음에 예측한 개발비로 개발을 완료할 수 없다.

20. 위험 관리 절차(3)

- 위험 분석
  - 위험 요소가 발생할 가능성과 영향력을 판단
  - 과거 프로젝트에서 데이터와 위험을 분석한 경험이 많은 개발자에 의존해 판단
  - 위험 발생 가능성의 척도
  - '매우 낮음(<10%), 낮음(10~25%), 보통(25~50%), 높음(50~75%), 매우 높음(>75%)'
  - 위험 발생 확률
  - 상: 80% 이상, 중: 30~80%, 하: 30% 미만
  - 영향력
  - 재앙, 심각함, 해결 가능함, 경미함 등으로 분류
  - 비용과 일정으로 분류: 상(20% 이상 초과), 중(5~20%), 하(5% 이하)

21. 위험 관리 절차(4)

- 위험 계획 수립
  - 식별된 위험 요소의 위험을 관리하기 위해 전략을 찾는 과정
  - 위험을 처리하는 위험 대응 방안 수립

- 위험 감시
  - 식별된 위험 요소의 발생 확률과 변화 등을 관리
  - 예측한 위험의 실제 발생 여부 확인
  - 실전에서 위험 대응 방안의 적절성 여부 확인

22. 객체지향

- 절차적 패러다임

  소프트웨어가 프로시저 단위로 구성됨

  프로시저 추상

    단순한 데이터에는 적합하나 복잡한 데이터를 가진

- 응용문제에는 부적합

  데이터 추상

    특정한 의미를 이루는 데이터의 조각들을 모아

- 그루핑

  시스템의 복잡도를 줄이는데 도움

    예) 레코드나 구조체

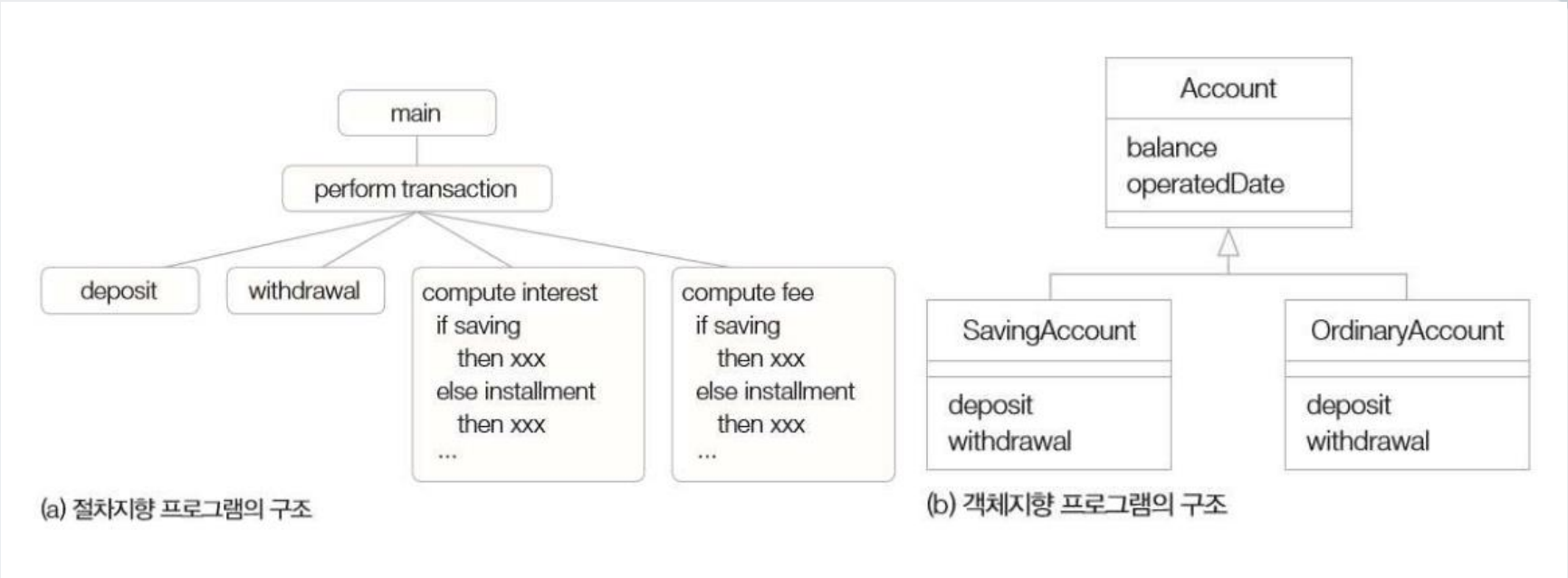
- 객체지향 패러다임

  프로시저 추상을 데이터 추상 관점으로 구성

23. 객체 지향 패러다임

- 문제를 해결하기 위한 모든 계산이 객체(object)라는 것을 기본으로 수행되는 방법
  - 객체는 클래스의 인스턴스. 클래스는
    - 자료가 추상화된 것이며, 동시에
    - 객체를 조작하는 프로시저에 대한 추상
  - 실행 프로그램은 특정 작업을 수행하기 위하여 모인 객체의 집합

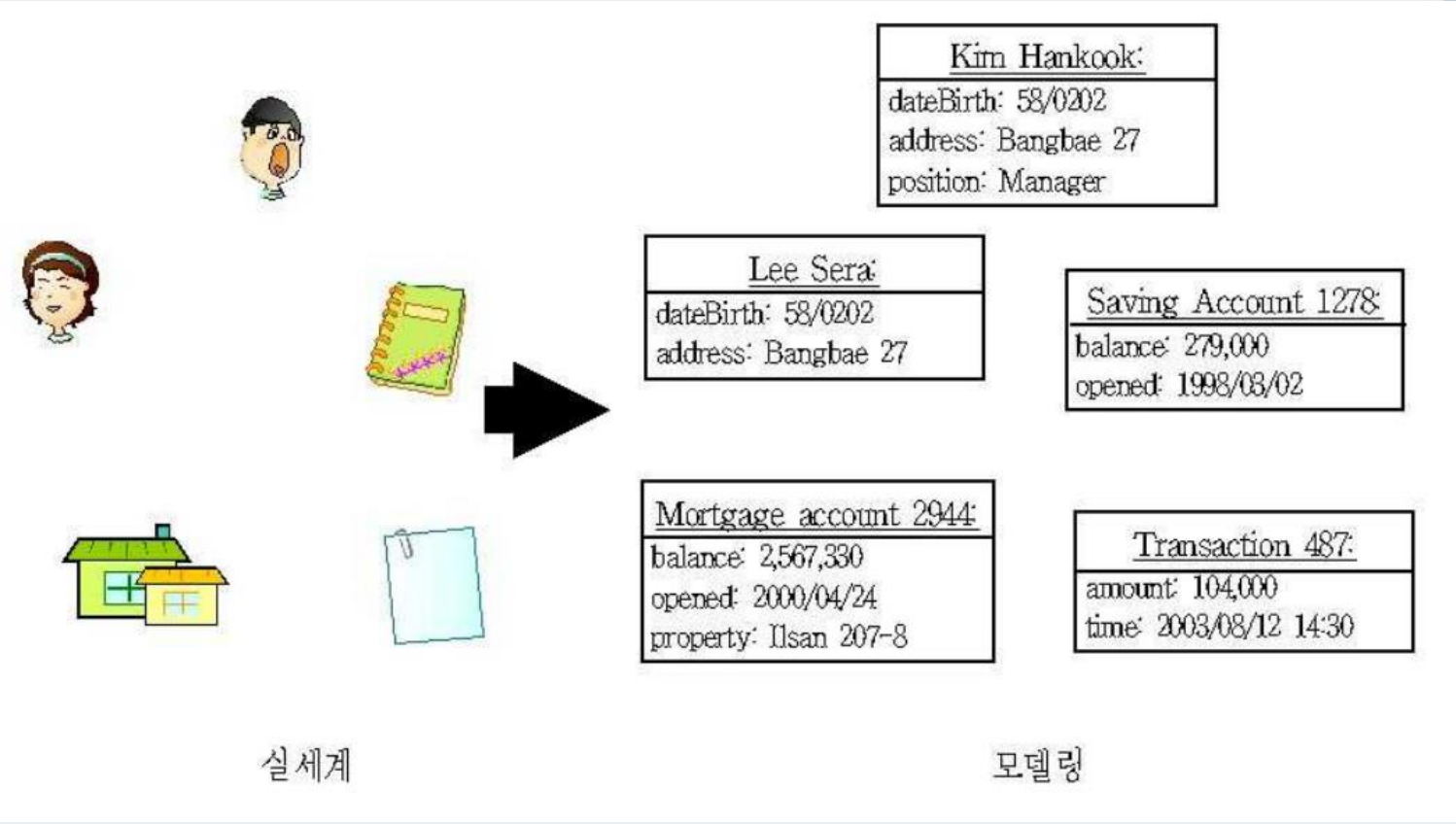
24. 프로그램 구조의 비교



25. 클래스와 객체

- 객체
  - 실행되는 소프트웨어 시스템에 존재하는 구조화된 데이터의 덩어리
  - 성질(property)를 가짐
    - 객체의 상태(state)를 나타냄
  - 행위(behavior)를 가짐
    - 어떤 식으로 작용하고 반응하는지를 나타냄
    - 실세계에 존재하는 객체의 행위를 시뮬레이션 할수도 있음

26. 객체


Kim Hankook:
dateBirth: 58/0202
address: Bangbae 27
position: Manager
Lee Sera:
dateBirth: 58/0202
address: Bangbae 27
Saving Account 1278:
balance: 279,000
opened: 1998/03/02
Mortgage account 2944:
balance: 2,567,330
opened: 2000/04/24
property: Ilsan 207-8
Transaction 487:
amount: 104,000
time: 2003/08/12 14:30
실세계
모델링

27. 클래스

- 클래스
  - 객체지향 프로그램에서 추상화의 단위
  - 유사 객체들을 정의
    - 인스턴스
  - - 소프트웨어 모듈
    - 인스턴스의 구조(속성)를 나타냄
    - 행위를 구현하는 메소드를 가짐

28. 클래스가 될 조건

- 인스턴스들을 가지고 있다면 클래스
- 클래스에 의하여 정의된 집합에 포함된 일개의 멤버라면 인스턴스

29. 인스턴스 변수

- 클래스 안에 정의된 변수로 각 인스턴스에 존재하는 데이터와 같다.
  - 속성
    - 단순한 데이터
    - 예) name, dateOfBirth
  - 연관
    - 주요 클래스 사이의 관계
    - 예) supervisor, courseTaken

30. 변수와 객체

- 변수
  - 객체를 refer(객체를 담고 있는 그릇)
  - 주어진 시점마다 다른 객체를 담고 있을 수 있음
- 한 객체가 어느 한 시점에 여러 다른 변수에 담겨 있을 수도 있음
- 변수의 타입
  - 변수가 담고있는 객체의 유형(클래스)을 결정

31. 클래스와 변수

- 클래스 변수는 속하는 모든 인스턴스가 공유
  - static 변수로 선언
  - 어떤 인스턴스가 클래스 변수의 값을 저장하면

- 클래스에 속한 다른 객체가 변수의 값의 변화를 알 수 있음
  - 클래스 변수
    - 디폴트, 상수 값 저장에 적합
    - 룩업 테이블이나 유사 구조에 적합

32. 오퍼레이션

- 절차적으로 추상화 된 것으로 클래스의 행위적 특성을 나타낸다.

- 행위를 구현한 코드와는 별개임
  - ✓ calculating area

33. 메소드

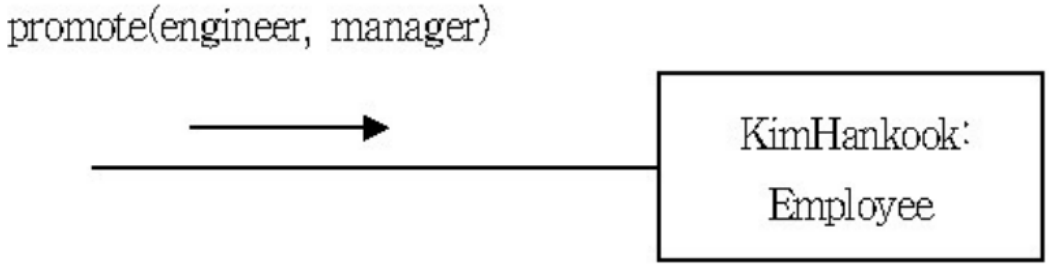
- 클래스의 행위를 구현하는 데 사용한 일종의 프로시저

- 여러 개의 클래스가 같은 이름의 메소드를 가질 수 있음
  - ✓ 동일한 추상 오퍼레이션을 각 클래스에 맞게 다르게 구현할 수 있음
  - ✓ 사각형의 면적 구하는 메소드는 원의 면적을 구하는 메소드와는 다르게 구현

34. 메소드

- 메소드를 통하여 객체의 속성을 조작

```
Employee    KimHankook;
KimHankook.promote(engineer, manager);
```



35. 상속

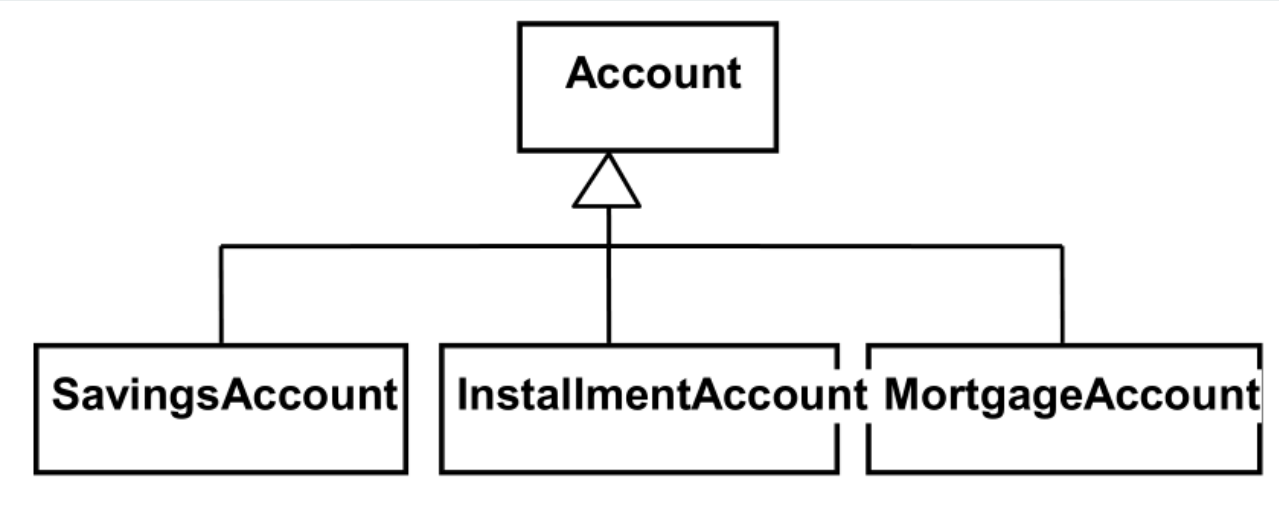
- 슈퍼 클래스
  - ✓ 서브 클래스의 공통적인 기능(feature)들을 가짐

- 상속 구조
  - ✓ 슈퍼 클래스와 서브 클래스 사이의 관계를 나타냄
  - ✓ 삼각형 화살표는 일반화(generalization)을 나타냄

- 상속(inheritance)
  - ✓ 슈퍼 클래스에 정의한 기능들을 서브 클래스가 묵시적으로 소유하는 것

36. 상속 구조의 예

- 상속(inheritance)
  - 슈퍼 클래스에 정의한 기능들을 서브 클래스가 묵시적으로 소유하는 것



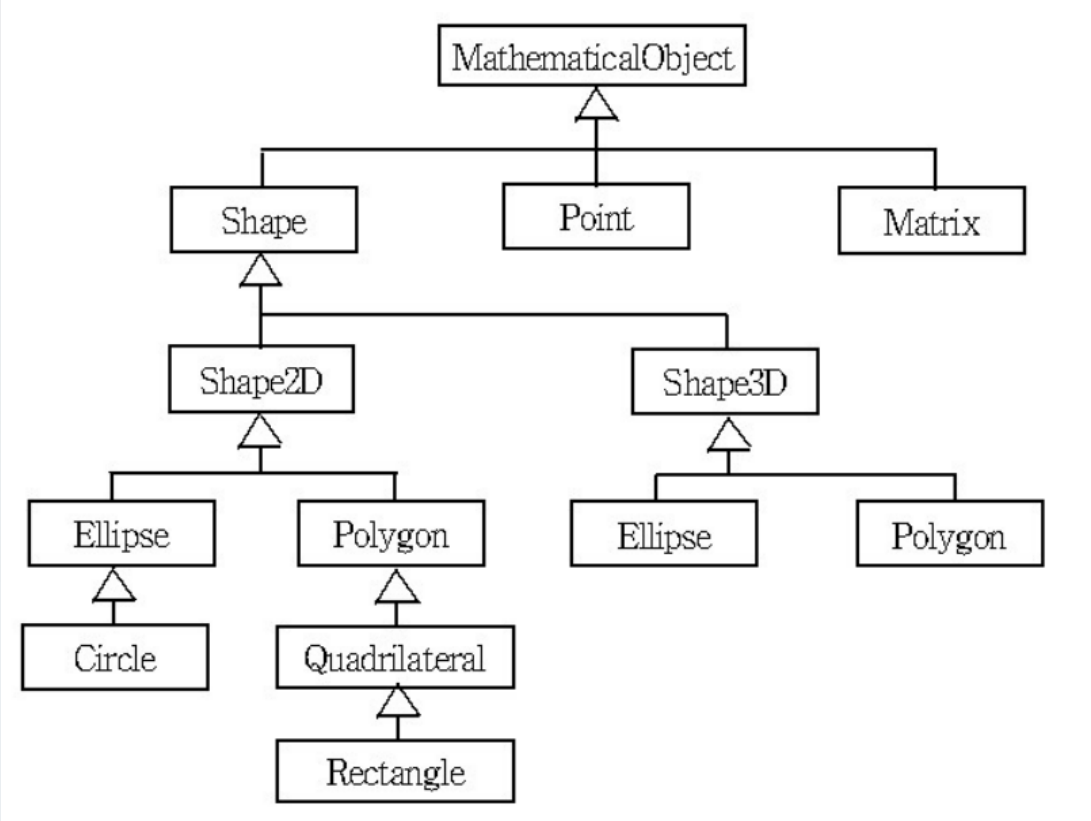
37. Is-a 규칙

- 일반화를 확인하려면 Is-a 관계를 만족시키는지를 체크
  - ✓ “A saving account is an account.” (보통예금 구좌는 은행구좌의 일종이다.)
  - ✓ “A graduate student is a student.” (대학원생은 학생의 일종이다.)

- “A school is a university.”이 성립하나?
  - ✓ Is-a 관계가 아님.

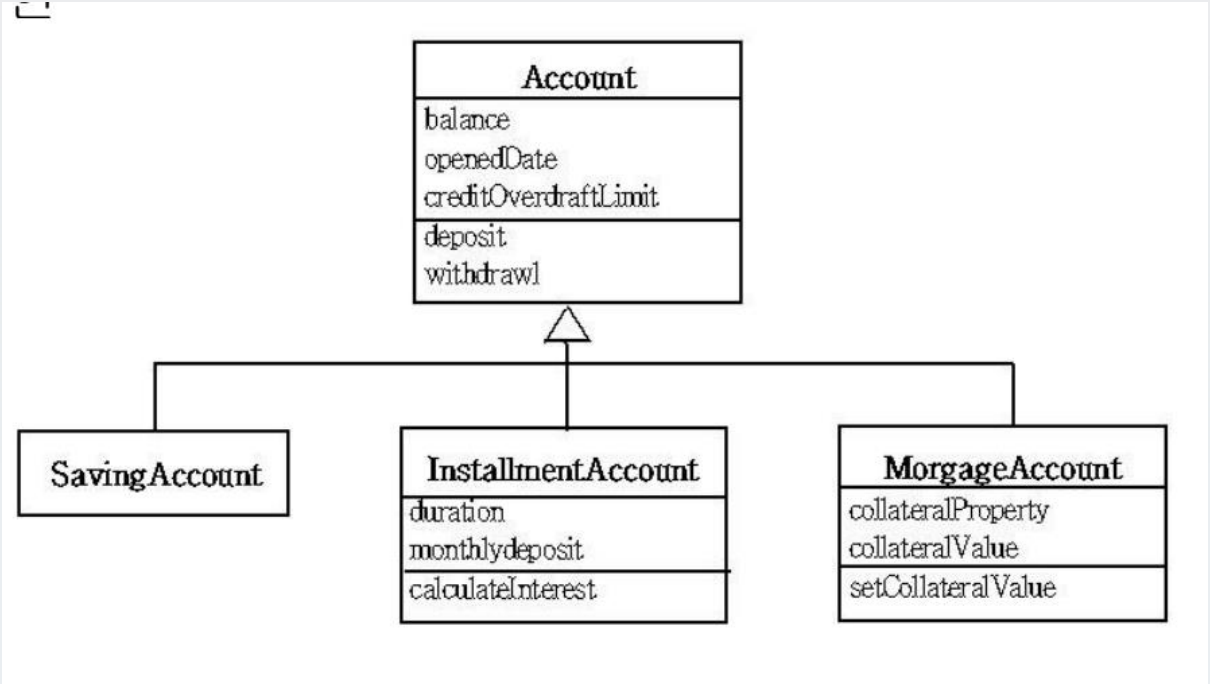
- ‘대학교는 학교의 일종이다.’가 되어야 함

38. 도형 관련 객체의 상속 구조



39. 상속 구조

- 상속된 모든 기능이 서브클래스에서도 의미가 통하는지 확인



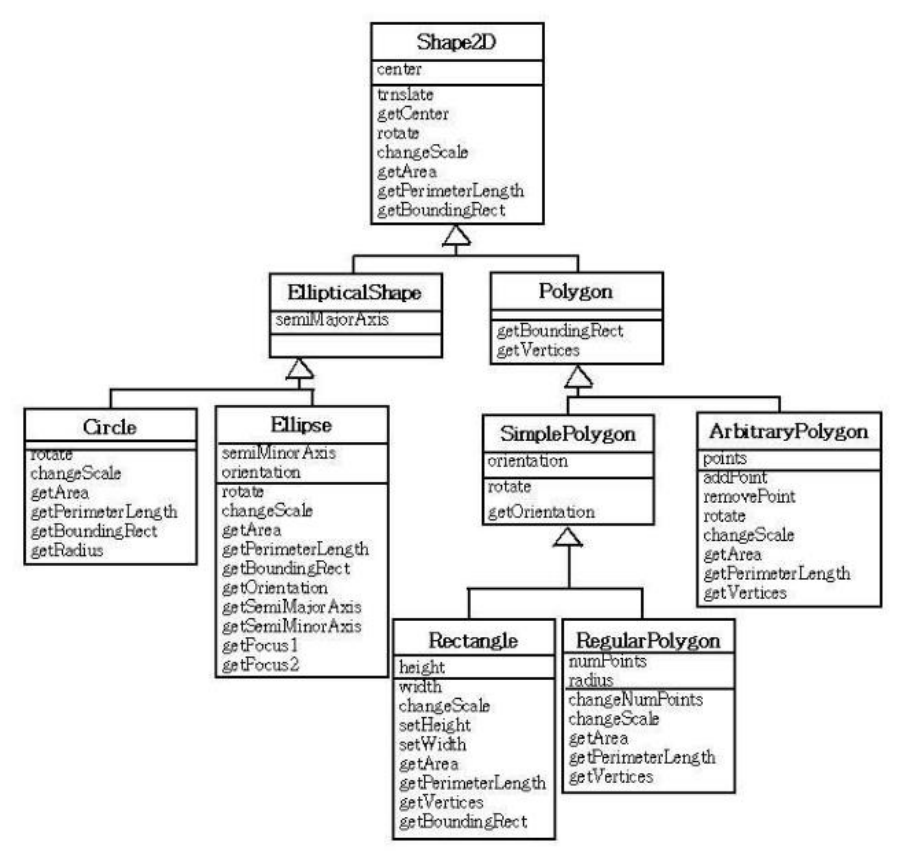
40. 다형성

- 하나의 추상 오퍼레이션이 서로 다른 클래스에서 다른

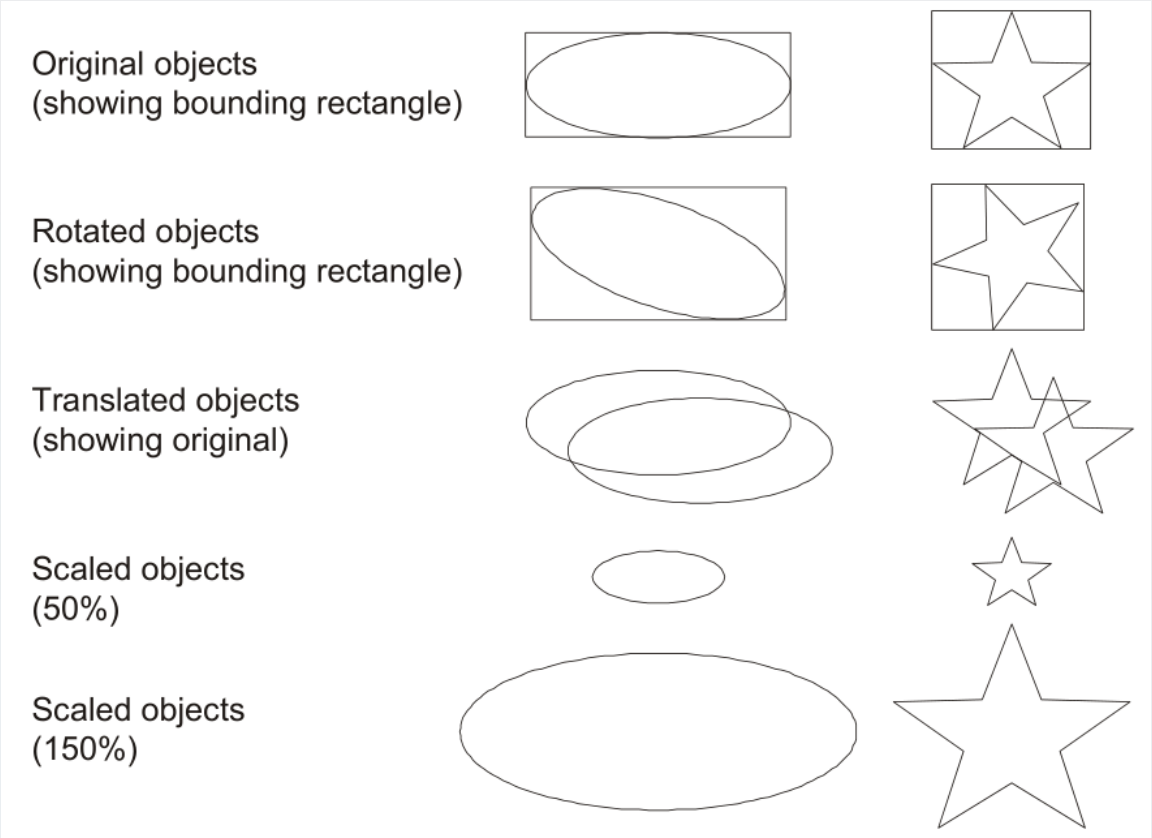
- 방법으로 구현될 수 있는 객체지향의 성질
  - 같은 이름을 가진 여러 개의 다른 메소드가 있어야 함
  - 어떤 메소드가 실행될지는 변수 안에 있는 객체의

- 종류에 좌우 됨
  - If-else나 switch 문장을 줄여주는 효과

41. 상속, 다형성, 변수


Shape2D
center
trnslate
getCenter
rotate
changeScale
getArea
getPerimeterLength
getBoundingRect
EllipticalShape
semiMajorAxis
Polygon
getBoundingRect
getVertices
Circle
rotate
changeScale
getArea
getPerimeterLength
getBoundingRect
getRadius
Ellipse
semiMinorAxis
orientation
rotate
changeScale
getArea
getPerimeterLength
getBoundingRect
getOrientation
getSemiMajorAxis
getSemiMinorAxis
getFocus1
getFocus2
SimplePolygon
orientation
rotate
getOrientation
ArbitraryPolygon
points
addPoint
removePoint
rotate
changeScale
getArea
getPerimeterLength
getVertices
Rectangle
height
width
changeScale
setHeight
setWidth
getArea
getPerimeterLength
getVertices
getBoundingRect
RegularPolygon
numPoints
radius
changeNumPoints
changeScale
getArea
getPerimeterLength
getVertices

42. Shape 객체의 오퍼레이션



43. 추상 클래스

- 오퍼레이션은 의미가 있는 가장 높은 상속구조에서 선언되는 것이 바람직함
  - 오퍼레이션은 그 수준에서 구현되지 않고 추상화 될 수 있음
  - 이런 클래스를 추상 클래스라 함
    - ✓ 인스턴스가 생성될 수 없음
    - ✓ 추상 클래스의 반대는 구체적(concrete) 클래스
  - 슈퍼 클래스가 추상 오퍼레이션을 가지고 있다면 서브 클래스 어딘가에는 그 오퍼레이션의 구체적 메소드를 가지고 있어야 함
    - ✓ 리프 클래스는 모든 오퍼레이션의 구체적 메소드를 가지고 있거나 상속 받아야 함

44. UML

- 설계 작업
  - ✓ 복잡한 아이디어를 간결하고 정확하게 표현, 교환
- 표현 방법
  - ✓ 추상화
  - ✓ 표준 방법
- Unified Modeling Language
  - ✓ 기능적 관점
  - ✓ 정적 관점
  - ✓ 동적 관점

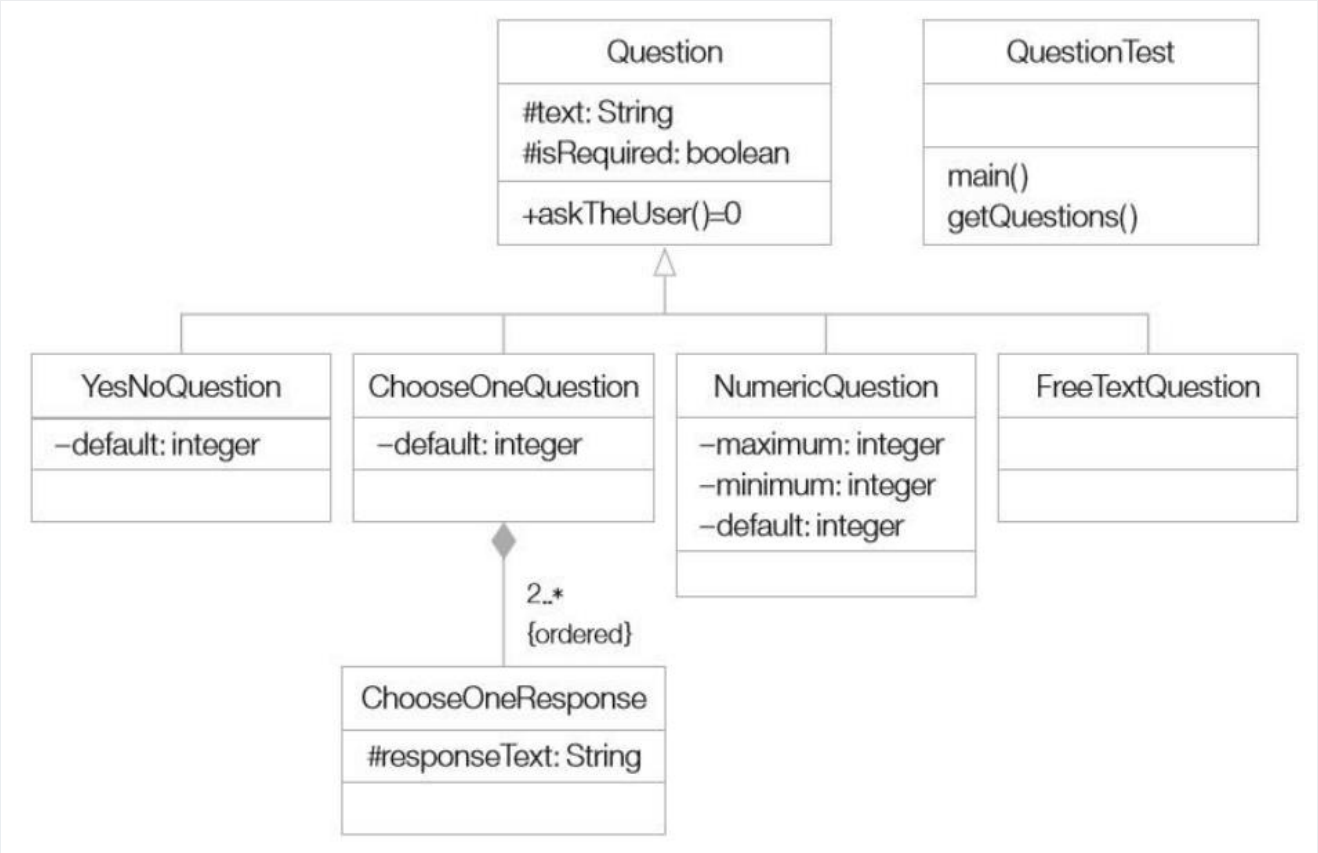
45. UML 다이어그램

- 사용 사례 다이어그램
- 클래스 다이어그램
- 시퀀스 다이어그램
- 상태 다이어그램
- 액티비티 다이어그램
- 패키지 다이어그램
- 배치 다이어그램

46. UML의 특징

- 비주얼화
- 명세
- 구축
- 문서화
- 테스트 기준

47. 문제 은행 만들기


Question
#text: String
#isRequired: boolean
+askTheUser()=0
QuestionTest
main()
getQuestions()
YesNoQuestion
-default: integer
ChooseOneQuestion
-default: integer
NumericQuestion
-maximum: integer
-minimum: integer
-default: integer
FreeTextQuestion
2..*
{ordered}
ChooseOneResponse
#responseText: String

다음 시간

소프트웨어 요구 분석