9.4 객체지향 테스팅

- 객체지향 방식의 프로그램에 적용
- 사용사례 기반 테스팅
- 상태 기반 테스팅

사용 사례 기반 테스팅

- **사용 사례 명세로부터 테스트 케이스 추출**

1. **액터의 입력과 액션을 파악**
   <예> 사용자 등록 사용 사례로부터 입력요소 추출

UC1: 새 고객 등록

액터: 새 고객	시스템: 웹 애플리케이션
	0. 시스템이 사용자 등록 링크를 가진 홈페이지를 디스플레이 한다.
1. 사용자가 고객 등록 링크를 클릭한다.	2. 시스템이 새 고객의 등록 양식을 디스플레이한다.
3. 사용자가 *사용자 ID, 패스워드, 재입력 패스워드*를 넣고 제출 버튼을 누른다.	4. 시스템이 로그인 ID와 패스워드를 검증하고 4.1 등록이 성공되었음을 디스플레이하거나 4.2 오류 메시지를 디스플레이하고 사용자에게 다시 시도할 것을 요구한다.
5. 사용자가 등록 성공 페이지를 본다.	

사용자 입력 요소

사용 사례 기반 테스팅

2. 입력 값을 결정

- **정상/비정상/예외 값 분류**

<예> 파악된 입력 요소의 값 결정

입력 요소	타입	값의 명세	정상	비정상	예외
로그인 ID	스트링	문자 길이가 8에서 20 사이	로그인 ID가 명세를 만족하여야 하며 다른 사용자와 중복되지 않아야	값의 명세를 만족하지 않거나 다른 사용자와 중복	스트링의 길이가 0, 1 또는 매우 큰 값 하나 이상의 빈칸이나 특수문자가 존재
패스워드	패스워드	길이가 8에서 12개의 문자, 적어도 하나의 문자, 숫자, 특수문자를 포함	패스워드 규칙에 맞는 패스워드	패스워드 규칙에 맞지 않는 패스워드	길이가 0, 1, 매우 큰 길이를 가진 패스워드 하나 또는 그 이상의 빈칸, 제어문자를 가진 패스워드
재입력한 패스워드	패스워드	패스워드와 같음	패스워드와 매치됨	재입력된 패스워드가 패스워드와 매치되지 않거나 복사-붙여넣기로 입력됨	

사용 사례 기반 테스팅

3. 테스트 케이스 생성
   - 입력 값 조합 규칙
     - 테스트 '입력 값 조합'이 프로그램 기능과 동작의 정확성을 가진다면 선택
     - 테스트 '입력 값 조합'이 오류를 발견할 가능성이 있다면 선택
     - 테스트 '입력 값 조합'이 선택된 다른 테스트 조합에 의해 포함될 수 있다면 삭제(중복제거), 유지할지 삭제할지 불분명한 것은 선택

테스트 케이스	로그인 ID	패스워드	재입력된 패스워드	예상 결과
1	정상	정상	정상	등록이 성공되었다는 페이지가 보임
2	정상	정상	비정상	오류 메시지가 보임
3	정상	비정상	정상	오류 메시지가 보임
~~4~~	~~정상~~	~~비정상~~	~~비정상~~	<테스트 케이스 3에 포함>
5	정상	예외	정상	오류 메시지가 보임
~~6~~	~~정상~~	~~예외~~	~~비정상~~	<테스트 케이스 2, 3에 포함>
7	비정상	정상	정상	오류 메시지가 보임
~~8~~	~~비정상~~	~~정상~~	~~비정상~~	<테스트 케이스 2, 7에 포함>
~~9~~	~~비정상~~	~~비정상~~	~~정상~~	<테스트 케이스 3, 7에 포함>
~~10~~	~~비정상~~	~~비정상~~	~~비정상~~	<테스트 케이스 2, 3, 7에 포함>
~~11~~	~~비정상~~	~~예외~~	~~정상~~	<테스트 케이스 5, 7에 포함>
~~12~~	~~비정상~~	~~예외~~	~~비정상~~	<테스트 케이스 2, 5, 7에 포함>
13	예외	정상	정상	오류 메시지가 보임
~~14~~	~~예외~~	~~정상~~	~~비정상~~	<테스트 케이스 7, 13에 포함>
~~15~~	~~예외~~	~~비정상~~	~~정상~~	<테스트 케이스 3, 13에 포함>
~~16~~	~~예외~~	~~비정상~~	~~비정상~~	<테스트 케이스 3, 7, 13에 포함>
~~17~~	~~예외~~	~~예외~~	~~정상~~	<테스트 케이스 5, 13에 포함>
~~18~~	~~예외~~	~~예외~~	~~비정상~~	<테스트 케이스 2, 5, 13에 포함>

상태 기반 테스팅

- 같은 입력에 대해 같은 동작을 보이며 동일한 결과를 생성하는 시스템(state-less system)을 대상
  - 배치 처리 시스템
  - 계산 중심 시스템
  - 하드웨어로 구성된 회로

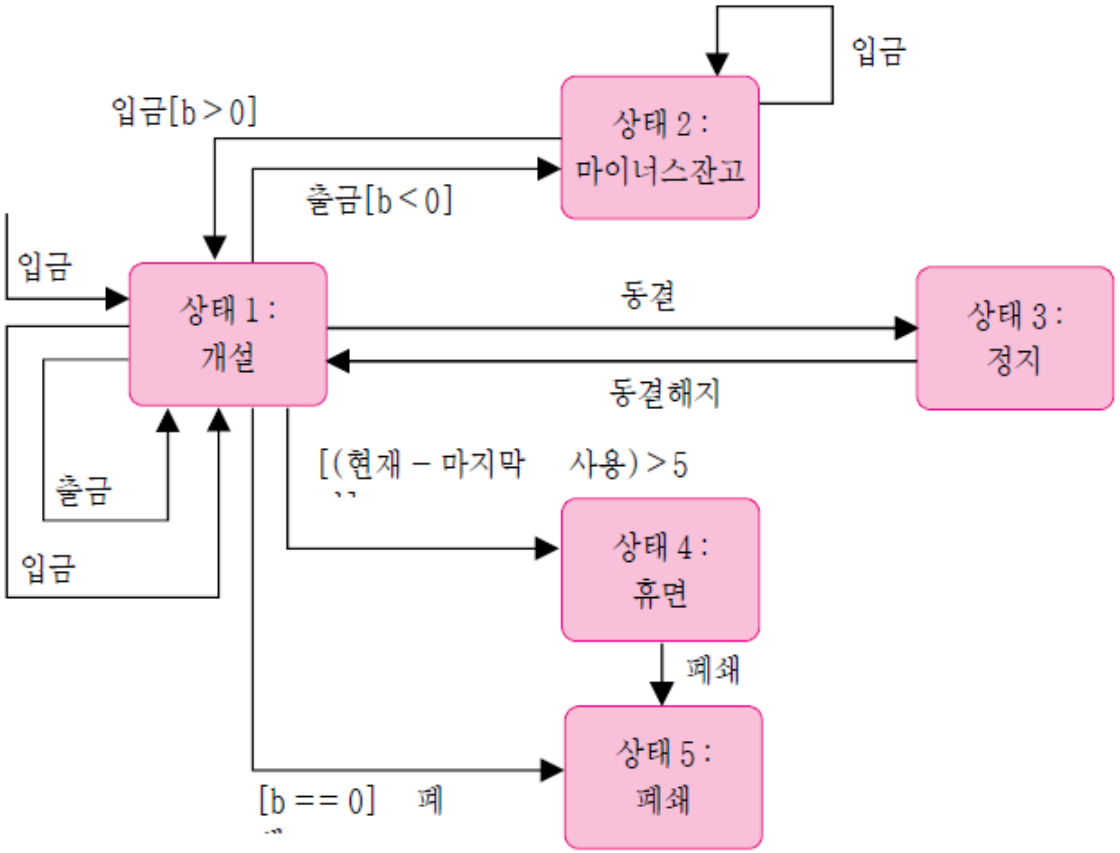
- 시스템의 동작은 시스템의 상태에 의해 좌우됨

- 상태 모델 구성요소
  - 상태 – 시스템의 과거 입력에 대한 영향을 표시
  - 트랜지션 – 이벤트에 대한 반응으로 시스템이 하나의 상태에서 다른 상태로 어떻게 변해가는지를 나타냄
  - 이벤트 – 시스템에 대한 입력
  - 액션 – 이벤트에 대한 출력

상태 기반 테스팅

음성 설명 없음

<예> 예금 계좌의 상태 모델 예시



상태 기반 테스팅

- 검증 기준(coverage)
  - 모든 트랜지션
    - 테스트 케이스 집합이 상태 그래프의 모든 트랜지션을 점검
  - 모든 트랜지션 쌍
    - 테스트 케이스 집합이 모든 이웃 트랜지션의 쌍을 점검
    - 유입(incoming)과 방출(outgoing) 트랜지션 쌍을 의미
  - 트랜지션 트리
    - 테스트 케이스 집합이 모든 단순 경로를 만족시키는 기준

    (단순경로: 시작 상태에서 다른 상태로 중복되지 않고 방문될 수 있는 경로)

No.	트랜지션	테스트 케이스
1	start →1	입금
2	1→1	출금
3	1→1	입금
4	1→2	입금(잔고〉0)
5	2→2	입금
6	2→1	출금(잔고〈0)
7	1→3	동결
8	3→1	동결해지
9	1→4	(현재–마지막 사용일〉>5년
10	1→5	폐쇄(잔고=0)
11	4→5	폐쇄

9.5 통합 테스팅

- 모듈의 인터페이스 결합을 테스트
  - 여러 개발 팀에서 개발한 각각의 단위 모듈을 대상
  - 모듈-모듈 간의 결합을 테스트

- 모듈의 결합 순서에 따라 방법이 다름
  - 빅뱅(big-bang)
  - 하향식(top-down)
  - 상향식(bottom-up)
  - 연쇄식(threads)

- 용어
  - 드라이버
    - 시험 대상 모듈을 호출하는 간이 소프트웨어
  - 스텁
    - 시험 대상 모듈이 호출하는 또 다른 모듈

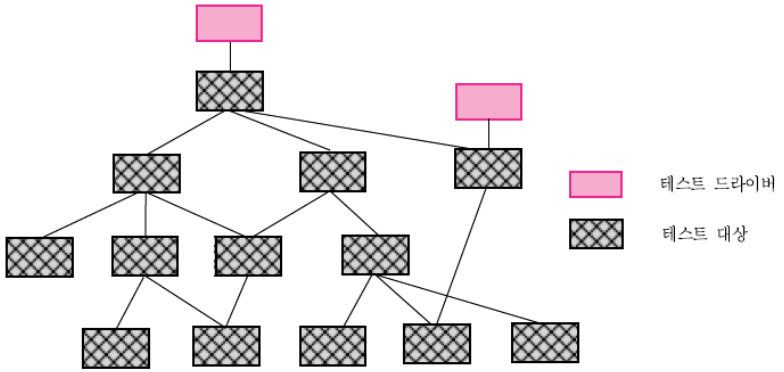
테스트 하니스(a test harness): a collection of software and test data configured to test a program unit.

빅뱅 통합

- 한 번에 모든 모듈을 모아 통합. 모든 모듈에 대한 단위 테스팅이 끝난 후.

- 장점
  - 고도의 신뢰도가 요구되는 시스템의 경우 중요 부분을 먼저 구현하기 때문에 의뢰자에게 신뢰감을 줄 수 있음.
  - 중요 부분을 먼저 구현함으로써 여러 번 테스트가 반복되어 완고한 개발이 가능함. ← 일부 모듈들은 stub이나 driver 형태로 구현해 통합 테스팅 후, 점차적으로 모듈을 추가해 가면서 통합 테스팅 진행

- 단점
  - 오류의 위치와 원인을 찾기 어려움
  - 단위 테스트에 많은 시간과 노력이 듬
    - 준비해야 할 드라이버/스텁 수가 많음
  - 개발 진도를 예측하기 어려움

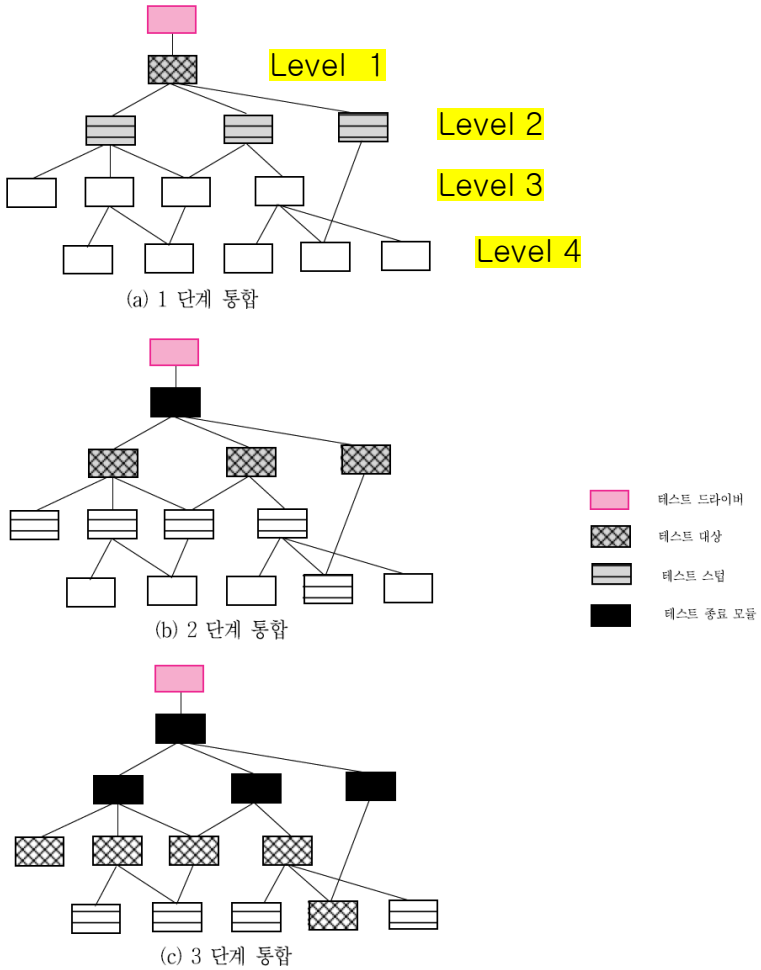


하향식 통합

- 시스템 구조상 최상위에 있는 모듈부터 통합

- 장점
  - 중요한 모듈의 인터페이스를 조기에 테스트
  - 스텁을 이용하여 시스템 모습을 일찍 구현가능
  - 개발자 입장에서 용이함

- 단점
  - 입출력 모듈이 상대적으로 하위에 있음
    - 테스트 케이스 작성 및 실행이 어려움
  - 중요 기능이 마지막에 구현됨

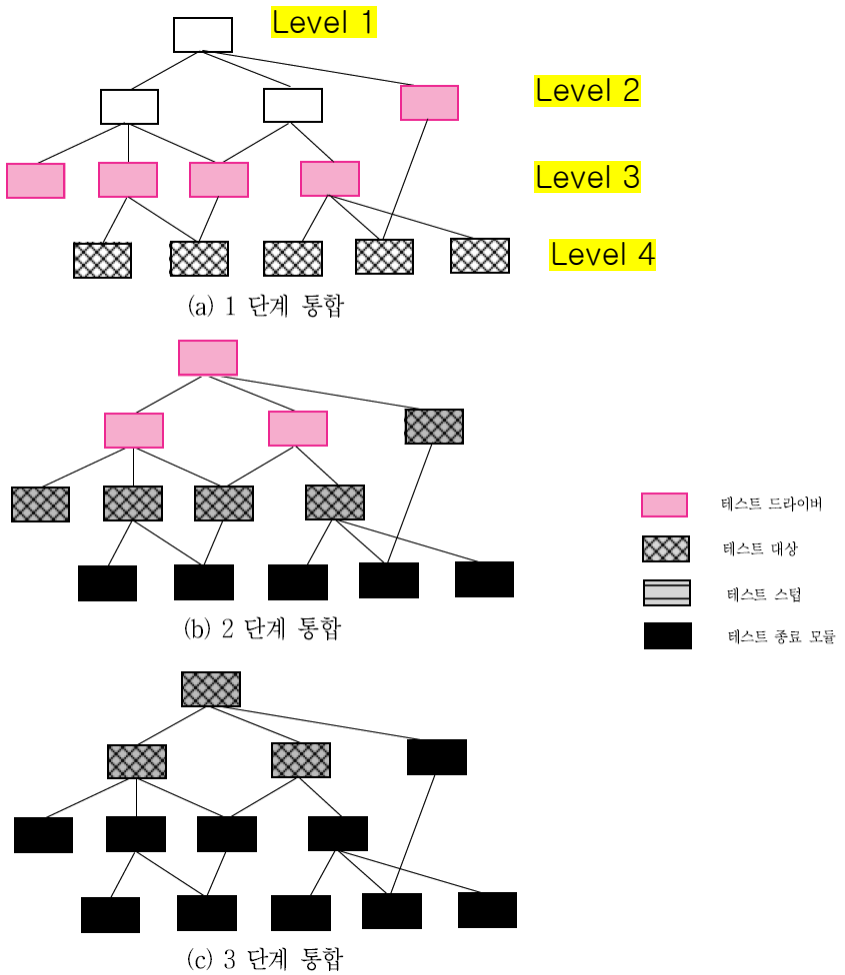


각 기능은, 시스템 구조도 (tree 형태)의 한 subtree 로 구현되는데 일반적으로 이런 subtree는 여러 level들에 걸친 모듈들로 구성됨.

상향식 통합

- 시스템 구조상 최하위에 있는 모듈부터 통합

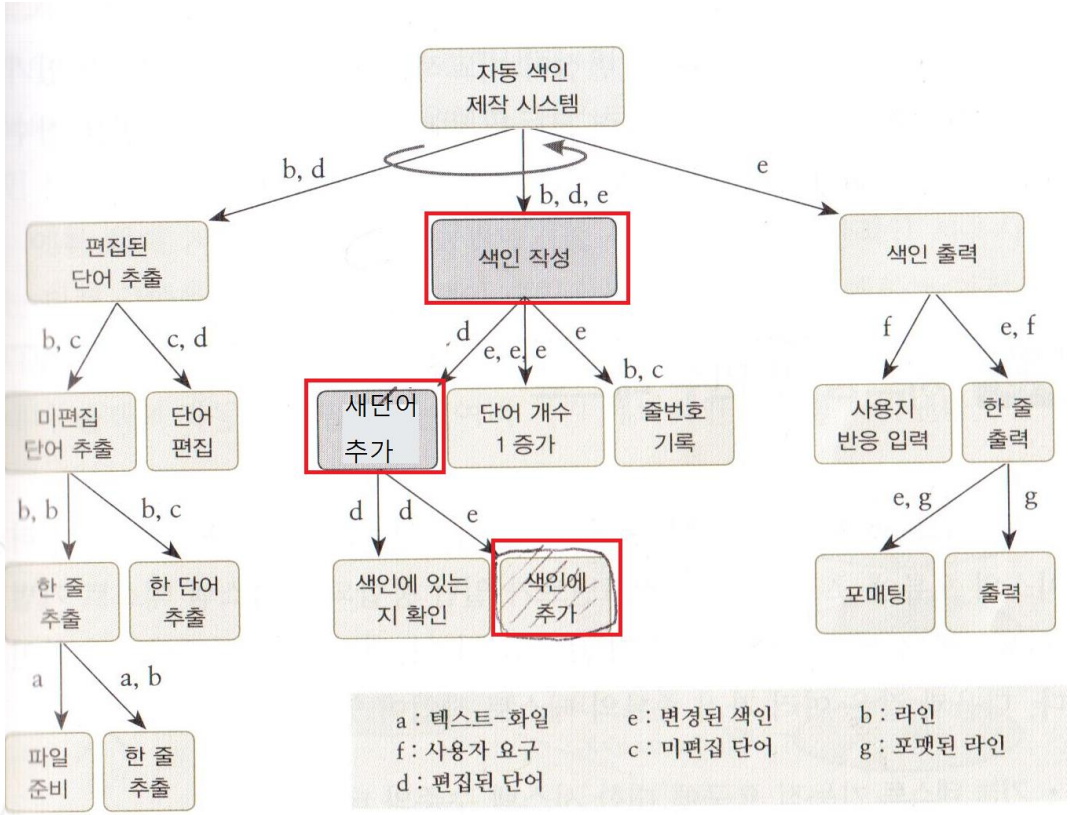
- 장점
  - 점증적 통합 방식(하향식에도 해당)
    - 오류 발견이 쉬움
    - 하드웨어 사용 분산
  - 하위층 모듈을 상위층보다 더 많이 테스트
- 단점
  - 초기에 시스템의 뼈대가 갖추어지지 않음
  - 상위층의 중요한 인터페이스가 마지막에 가서야 확인 가능
  - 의뢰자에게 시스템을 시험해 볼 기회를 충분히 제공하지 못함



연쇄식 통합

- 특정 기능을 수행하는 모듈의 최소 단위(thread)로 부터 시작 ➔ 여러 계층에 걸쳐 있을 수 있고, 하나의 서브 트리가 되기도 함.
  - 입력, 출력
  - 어느 정도의 기본 기능을 수행하는 모듈
- 한 개 thread로 또는 여러 개의 독립된 thread로 출발해, 첨차 다른 모듈을 추가해 테스팅해 나감
- 상대적으로 중요한 모듈부터 개발

- 장점
  - 초기에 시스템의 골격이 형성
    - 사용자 의견을 빨리 확인 가능
  - 시스템을 나누어 개발 하기 쉽다



- 빨간 사각형으로 된 모듈들이 thread가 될 수 있음.
- - 이를 테스팅하기 위해서 필요한 주변 모듈들은, stub이나 driver로 구현

9.6 시스템 및 인수 테스팅

- 컴포넌트 통합 후 수행하는 테스트 기법

- 테스트 종류
  - 기능 테스트
  - 성능 테스트
  - 보안 테스트
  - 사용성 테스트

  - 인수 테스트

    테스터: 사용자(또는 개발의뢰인이나 대리인)

    알파테스트: 개발환경에서 테스트, 베타테스트(필드테스트, 시험사용): 운영환경에서 테스팅

  - 설치 테스트

(강의 종료)