

소프트웨어 공학 (Software Engineering)

**테스팅
(Testing)**



In this chapter ...

테스팅(Testing)

- 테스트 작업의 원리와 목표
- 테스팅 단계와 종류
- 블랙박스, 화이트박스 테스트 기법의 이해와 적용
- 객체지향 프로그램의 테스트 기법
- 통합, 시스템, 인수 테스팅을 하는 방법
- 테스트 자동화 도구





In this chapter ...

테스팅(Testing)



9.1 테스팅 기초



9.2 블랙박스 테스팅



9.3 화이트박스 테스팅



9.4 객체지향 테스팅



9.5 통합 테스팅



9.6 시스템 및 인수 테스트



9.7 테스트 도구



- **소프트웨어 테스팅은 소프트웨어의 정확성을 확증하는 과정으로,**
 - 결함이나 원치 않는 동작을 찾아내고,
 - 요구와 제약에 맞는지 검증하는 작업이다.
- **좋은 테스트란 숨어있는 오류를 잘 발견하는 것**





오류(error)

- 프로그램 실행 결과가 예상결과와 다른 경우
- 결함 및 고장을 일으키게 한 인간의 실수



결함(fault)

- 버그(bug)
- 소프트웨어 오작동의 원인



고장(failure)

- 명세로 작성된 요구와 기능을 제대로 수행할 수 없는 경우
- 모든 결함이 고장을 발생하는 것은 아님



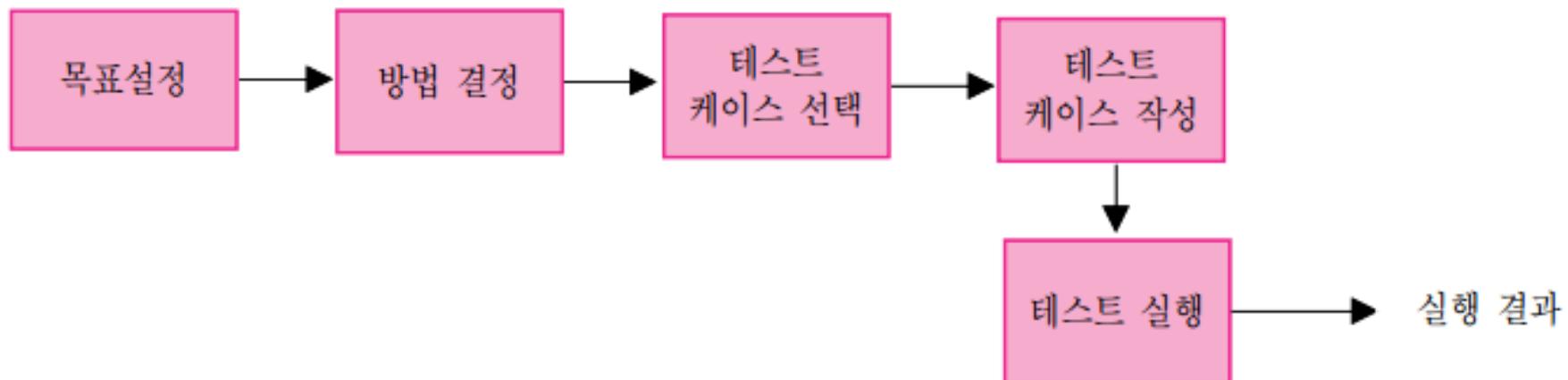
- ❶ 테스팅은 오류를 발견하려고 프로그램을 실행시키는 것이다.
- ❷ 완벽한 테스팅을 불가능하다.
- ❸ 테스팅은 창조적이면서 어려운 작업이다.
- ❹ 테스팅은 오류의 유입을 방지한다.
- ❺ 테스팅은 구현과 관계없는 독립된 팀에 의해 수행되어야 한다





테스팅 과정 (1/2)

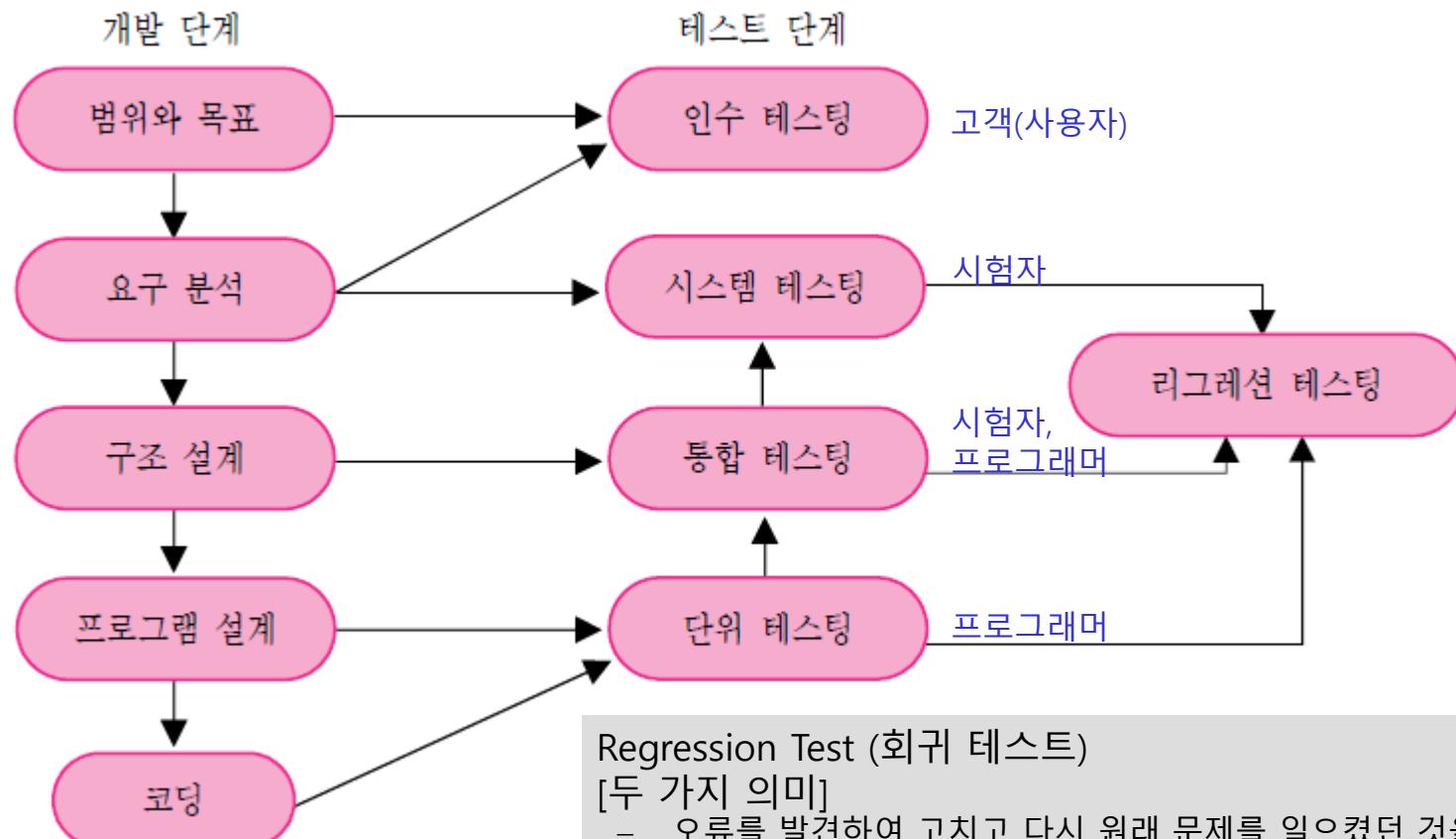
테스팅(Testing)



- ① 목표설정: 무엇을 점검할 것인지 결정한다.
- ② 방법결정: 어떻게 테스트할 것인지 방법을 결정한다.
- ③ 테스트 케이스 선택: 테스트 자료, 실행 조건 등의 테스트 케이스를 선택한다.
- ④ 테스트 케이스 작성: 테스팅의 예상되는 올바른 결과를 작성한다.
- ⑤ 테스트 실행: 테스트 케이스로 실행시킨다.



▣ 테스트 단계와 개발 단계와의 관계 (V 모형에 해당)





테스트 케이스

테스팅(Testing)

- 성공적인 테스팅은 좋은 테스트 케이스를 찾는 것임
- 테스트 케이스는 시험 대상 단위 별로 묶어 미리 준비함
(블랙박스 테스팅에서 테스트 케이스 선정 방법을 자세히 다룸)

고유번호	테스트 대상	테스트 조건	테스트 데이터	예상 결과
FT-1-1	로그인 기능	시스템 초기 화면	정상적인 사용자 ID('gdhong')와 패스워드('1234')	시스템 입장
FT-1-2	"	"	비정상적 사용자 ID('%\$##')와 패스워드(' ')	로그인 오류 메시지



In this chapter ...

테스팅(Testing)

9.1 테스팅 기초

9.2 블랙박스 테스팅

9.3 화이트박스 테스팅

9.4 객체지향 테스팅

9.5 통합 테스팅

9.6 시스템 및 인수 테스트

9.7 테스트 도구

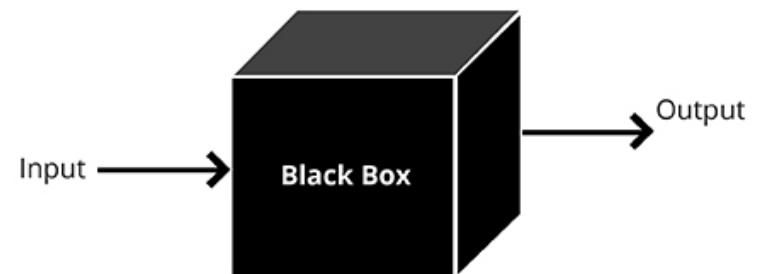


블랙박스 테스팅

테스팅(Testing)

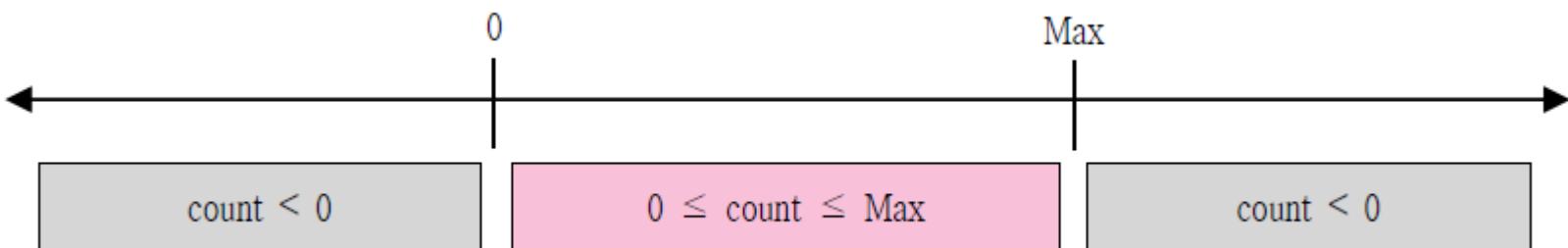
- ▣ 프로그램 내부 구조를 고려하지 않음 (vs 화이트박스 테스팅)
- ▣ 테스트 케이스는 프로그램이나 모듈의 요구나 명세를 기초로 결정
- ▣ 입력과 출력에 대해 알아야 함
- ▣ 기능 테스트(functional testing)에 해당함
- ▣ 가능한 모든 기능을 전부 테스트 하는 것이 좋음

BLACK BOX TESTING APPROACH



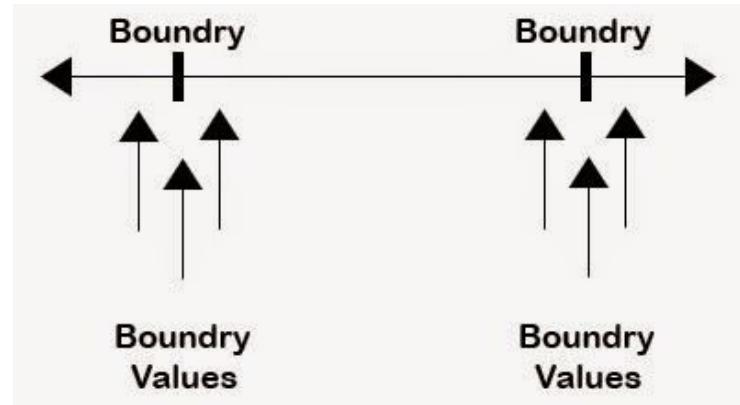


- ▣ 프로그램에서 같은 부분을 실행하고 결과를 확인하는 대표 값을 설정
 - 정상적인 동치클래스
 - 비정상적인 동치클래스
- ▣ 모든 대표 값을 찾아내어 각 클래스에서 하나의 값으로 프로그램을 실행시킨다면 전수 테스트와 같음
- ▣ 예제: 입력 범위 조건 $0 \leq \text{count} \leq \text{Max}$ 의 동치 클래스
 - 정상 $0 \leq \text{count} \leq \text{Max}$
 - 비정상 $\text{count} < 0$, $\text{count} > \text{Max}$





- ▣ 동치 클래스의 **경계**에서 문제를 발생하는 특수한 값이 존재
 - ▣ 동치 클래스 경계에 있는 값을 가진 테스트 케이스는 높은 효율을 가짐
 - ▣ 동치클래스의 경계에 있는 값을 테스트 입력으로 선택
-
- ▣ 예제: 하나의 입력 값 X에 대한 테스트 케이스
 - 입력 값 조건: $\min \leq X \leq \max$
 - 경계 값 케이스: $\min-1, \min, \min+1, \max-1, \max, \max+1$

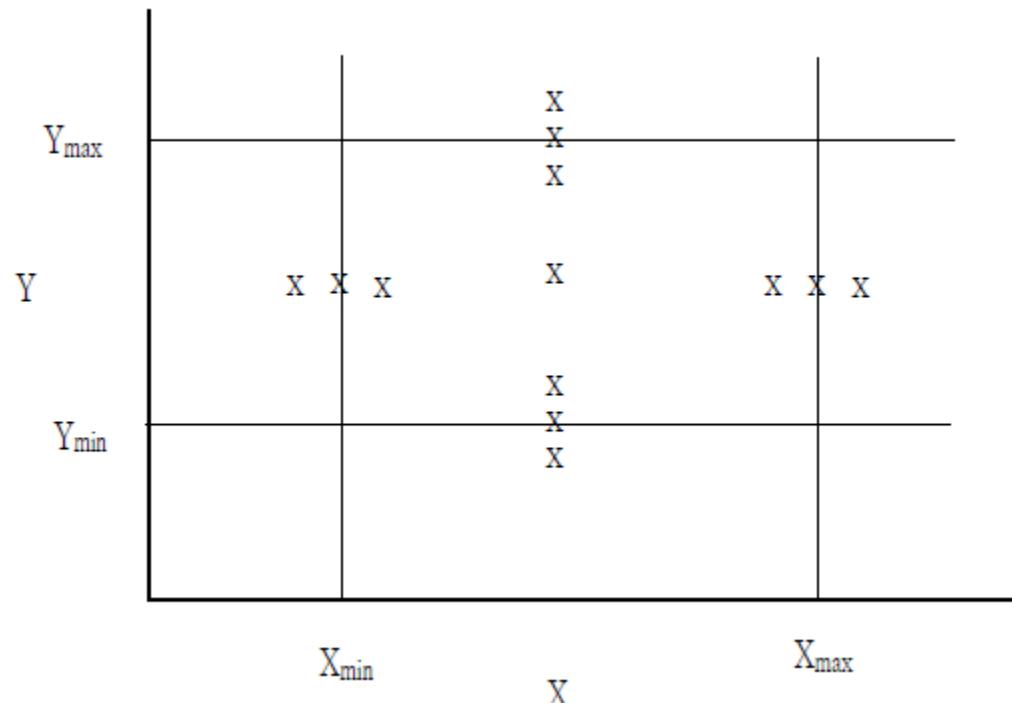




예제: 두개의 입력 값 X, Y에 대한 테스트 케이스

- 입력 값 조건: $\min \leq X \leq \max, \min \leq Y \leq \max$

- 경계값 케이스





- 입력 조건이 너무 많아 동치 클래스/경계 분석이 어려울 경우,
입력 조건의 조합을 체계적으로 선택하는 테스트 기법

노드와 기호로 표시

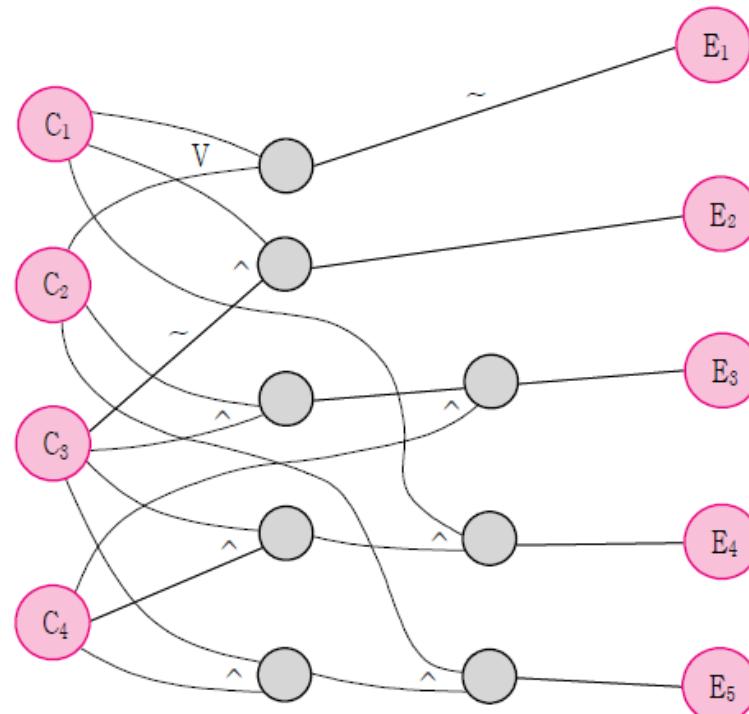
- 노드: 원인(입력조건), 결과(출력 조건)
- 기호: \wedge (and), \vee (or), \sim (not)
- 예제

원인:

- c1. 명령어가 입력
- c2. 명령어가 출금
- c3. 계정 번호가 정상
- c4. 트랜잭션 금액이 정상

결과:

- e1. ‘명령어 오류’라고 인쇄
- e2. ‘계정 번호 오류’라고 인쇄
- e3. ‘출금액 오류’라고 인쇄
- e4. 트랜잭션 금액 출금
- e5. 트랜잭션 금액 입금





graph LR; A[원인 결과 그래프 (2/2)] --> B[테스팅(Testing)]; B --> C[그리드 테스트 케이스 생성];

그리드 테스트 케이스 생성

• 원인 결과 그래프로부터 테스트 케이스를 만들 수 있음

→ 각각의 결과들에 대하여 조건의 조합을 나열한 표를 작성

- x: don't care
- 0: false
- 1: true

No.	1	2	3	4	5
c1	0	1	x	x	1
c2	0	x	1	1	x
c3	x	0	1	1	1
c4	x	x	0	1	1
e1	1				
e2		1			
e3			1		
e4				1	
e5					1



In this chapter ...

테스팅(Testing)

9.1 테스팅 기초

9.2 블랙박스 테스팅

9.3 화이트박스 테스팅

9.4 객체지향 테스팅

9.5 통합 테스팅

9.6 시스템 및 인수 테스트

9.7 테스트 도구

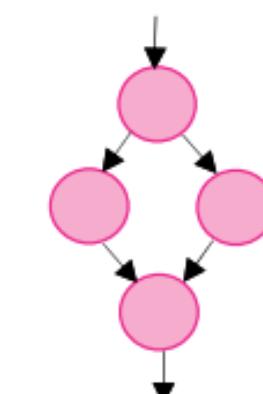




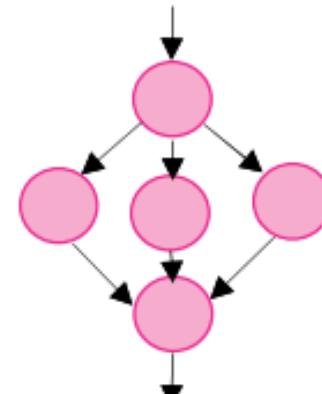
- 모듈 내의 작동을 자세히 관찰하는 방법 (vs 블랙박스 테스팅)
- 모듈의 논리적인 구조를 체계적으로 점검하는 구조적 테스팅
- 여러가지 프로그램 구조를 기반으로 테스트
- 논리 흐름도(logic-flow diagram)을 이용
 - 노드: 모듈 내의 세그먼트
 - 간선: 제어 흐름



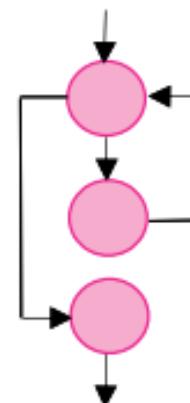
선행 블록



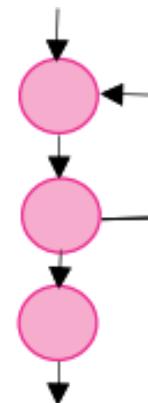
if-then-else 블록



switch 블록



while/for 블록

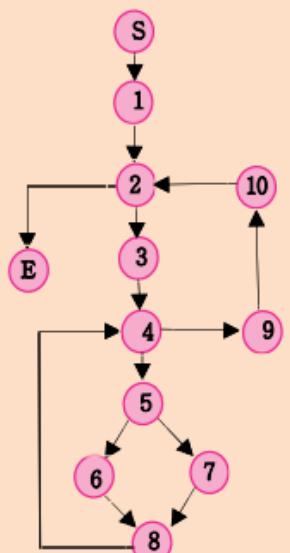


until 블록



- 기본 경로(basis path): 독립적인 논리 흐름을 검사하는 테스트 케이스
- 시작 노드에서 종료 노드까지의 서로 다른 경로로 싸이클은 최대 한번만 지나야 함
- 예제: Remove 함수에 대한 논리흐름 그래프와 테스트 케이스

```
public void Remove(LinkedList list) {  
    Item p, q;  
    ① p = IItem(list.getFirst());  
    ② while(p!=null) {  
        ③ q=(Item)p.getNext();  
        ④ while (q!=null) {  
            ⑤ if(p.compareTo(q)==0)  
                list.remove(q);  
            ⑥ else q=(Item)p.getNext();  
        ⑦ }  
        ⑧ p=(Item)p.getNext();  
    ⑨ }  
}
```



1. S-1-2-E: 빈 리스트
2. S-1-2-3-4-9-10-2-E: 한 개의 요소를 가진 리스트
3. S-1-2-3-4-5-6-8-4-9-10-2-E: 중복 요소를 가진 리스트
4. S-1-2-3-4-5-7-8-4-9-10-2-E: 중복 요소가 없는 리스트



④ 기본 경로의 수를 결정하는 이론

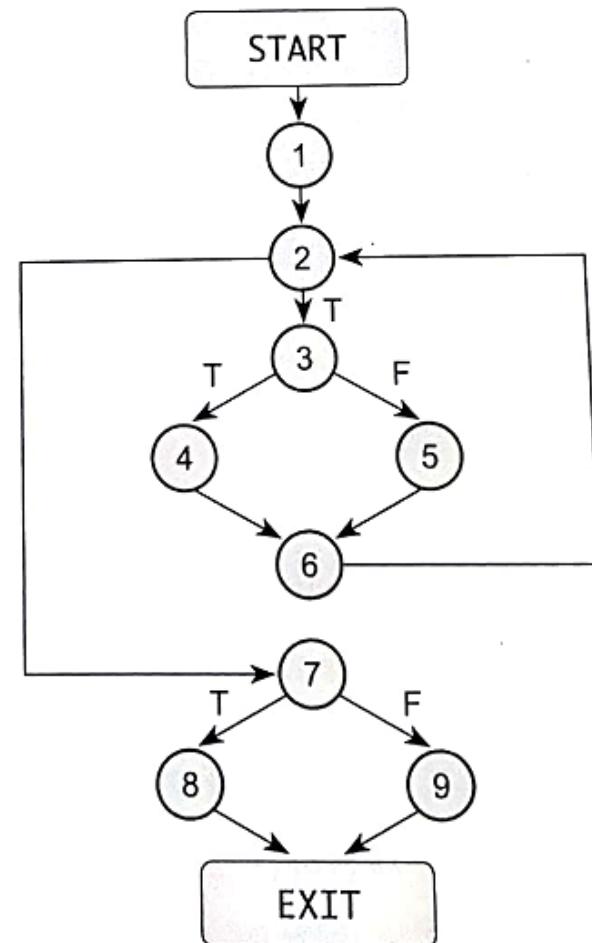
⑤ 싸이클로매틱 복잡도 계산 세 가지 방법

- 폐쇄 영역의 수+1: 논리 흐름 그래프는 이차원 평면을 여러 영역으로 나누며, 이 중 폐쇄된 영역의 수에 1을 더한 값
- 노드와 간선의 수: 간선의 수에서 노드의 수를 빼고 2를 더한 값
- 단일 조건의 수+1 : 참과 거짓으로 판별되는 원자적 조건의 수에 1을 더한 값
(AND, OR로 연결된 복합 조건이 아니라 단일 조건이라는 뜻)
→ 예제: 다음 페이지

⑥ 세 가지 방법의 계산 값이 같아야 함 (정리로서 증명됨)



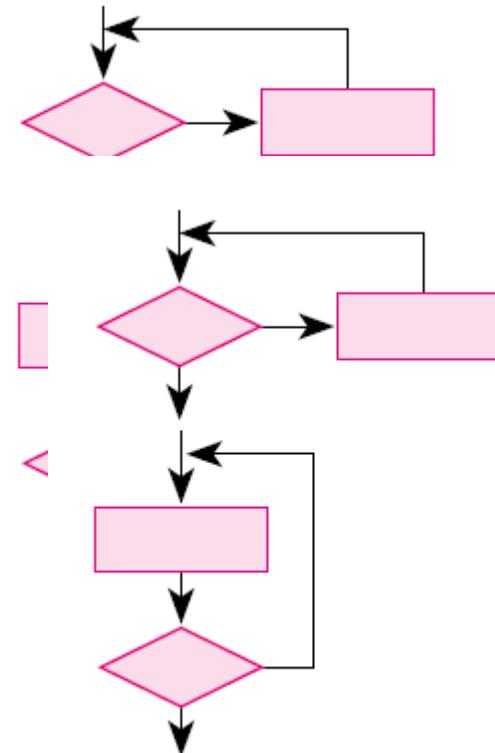
- 싸이클로매틱 복잡도 계산 예제
- 폐쇄 영역의 수+1: $3+1 = 4$
- 노드와 간선의 수(간선-노드+2): $13-11+2 = 4$
- 단일 조건의 수+1 : $3+1 = 4$
(노드 2, 노드 3, 노드 7에 참/거짓의 단일 조건)





④ 단순 반복: 경계값 분석 방법 이용

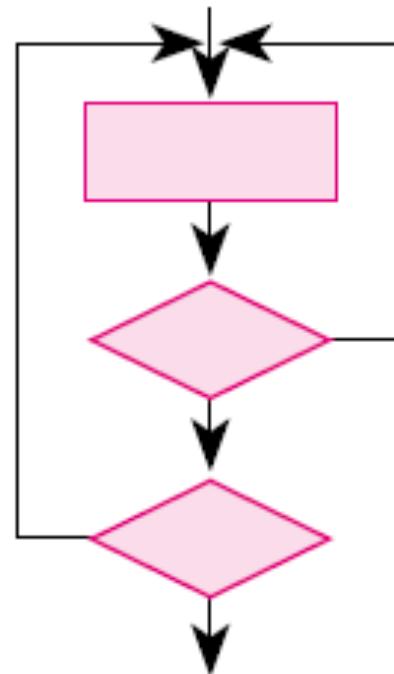
- 반복 구조를 들어가지 않고 생략
- 반복 구조 안에서 한 번 반복
- 반복 구조 안에서 두 번 반복
- 일정한 횟수의 반복
- 반복 최대 횟수 -1 만큼 반복
- 반복 최대 횟수만큼 반복
- 반복 최대 횟수 +1 만큼 반복





중첩된 반복: 가장 내부에 있는 반복 구조부터 테스트

- 최소 횟수의 반복
- 최소 횟수보다 하나 많은 반복
- 범위 내 임의의 횟수 반복
- 최대 횟수보다 하나 적은 반복
- 최대 횟수의 반복
- 외부로 향하여 다음 반복구조를 테스트





반복문의 테스팅 (3/3)

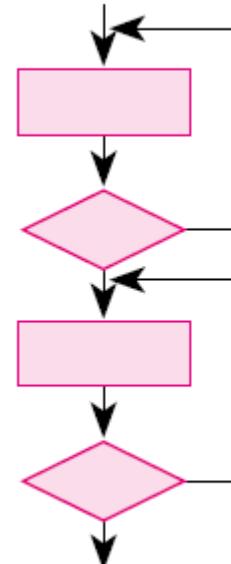
테스팅(Testing)

연속된 반복

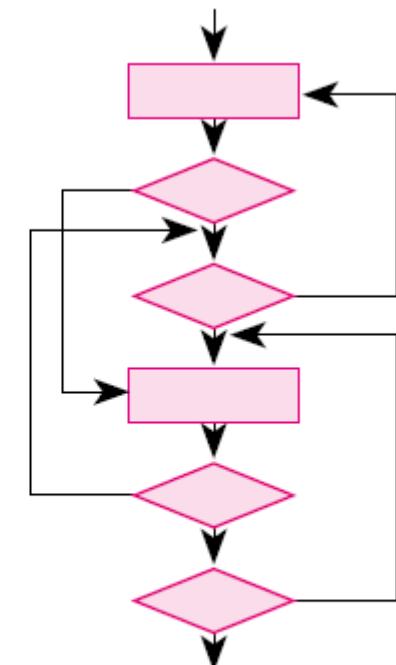
- 반복구조가 서로 독립적 → 단순반복으로 테스트
 - 어느 한 반복구조의 제어변수가 다른 반복구조에 직간접 의존
→ 중첩된 반복으로 테스트

비구조화 반복

- 구조적 반복 형태로 변경하여 테스트



(c) 연속된 반복



(d) 비구조화 반복



In this chapter ...

테스팅(Testing)

9.1 테스팅 기초

9.2 블랙박스 테스팅

9.3 화이트박스 테스팅

9.4 객체지향 테스팅

9.5 통합 테스팅

9.6 시스템 및 인수 테스트

9.7 테스트 도구



④ 전통적 테스팅 방식

- 블랙박스 테스팅, 화이트박스 테스팅
- 객체지향 방법에서 클래스 안의 “메소드”에 적용될 수 있음

⑤ 객체지향 방식의 프로그램에 적용할 수 있는 방법 소개

- 사용 사례 기반 테스팅
- 상태 기반 테스팅



④ 사용 사례 명세로부터 테스트 케이스 추출

Step 1: 액터의 입력과 액션을 파악한다.

- 예제: 사용자 등록 사용 사례로부터 입력 요소 추출

UC1: 새 고객 등록

액터: 새 고객	시스템: 웹 애플리케이션
	0. 시스템이 사용자 등록 링크를 가진 홈페이지를 디스플레이 한다.
사용자 입력 요소	1. 사용자가 고객 등록 링크를 클릭한다. 3. 사용자가 사용자 ID, 패스워드 재입력 패스워드를 넣고 제출 버튼을 누른다.
	2. 시스템이 새 고객의 등록 양식을 디스플레이 한다. 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.



사용 사례 기반 테스팅 (2/4)

테스팅(Testing)

Step 2: 입력 값을 결정한다.

- 정상/비정상/예외 값 분류
- 예제: 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 값이나 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 (3/4)

테스팅(Testing)



Step 3: (입력 값 조합 규칙에 따라) 테스트 케이스를 생성한다.

- 테스트 조합이 프로그램 기능과 동작의 정확성을 가진다면 선택
- 테스트 조합이 오류를 발견할 가능성이 있다면 선택
- 테스트 조합이 선택된 다른 테스트 조합에 의해 포함될 수 있다면 삭제(중복제거), 유지
할지 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>



사용 사례 기반 테스팅 (4/4)

테스팅(Testing)

Step 4: 실제적인 테스트 케이스를 생성한다.

- 테스트 케이스 각각에 대해 실제 테스트를 수행할 값을 생성

테스트 케이스	로그인 ID	패스워드	재입력된 패스워드	예상 결과
1	“newuser@naver.com”	“gls123%KSL”	“gls123%KSL”	등록이 성공되었다는 페이지가 보임
2	“newuser@naver.com”	“gls123%KSL”	“xxxxxxxxxx”	오류 메시지가 보임
3	“newuser@naver.com”	“vvvv”	“gls123%KSL”	오류 메시지가 보임
5	“newuser@naver.com”	“z”	“gls123%KSL”	〈테스트 케이스 3에 포함〉
7	“olduser@naver.com”	“gls123%KSL”	“gls123%KSL”	오류 메시지가 보임
13	“new user@naver.com”	“gls123%KSL”	“gls123%KSL”	〈테스트 케이스 2, 3에 포함〉

Step 5: 테스트를 구현 및 실행한다.



State-less 시스템: 같은 입력에 대해 같은 동작/동일한 결과를 생성

- 배치 처리 시스템
- 계산 중심 시스템
- 하드웨어로 구성된 회로

하지만, 많은 시스템의 경우, 그 동작은 시스템의 상태에 의해 좌우됨

상태 모델 구성요소

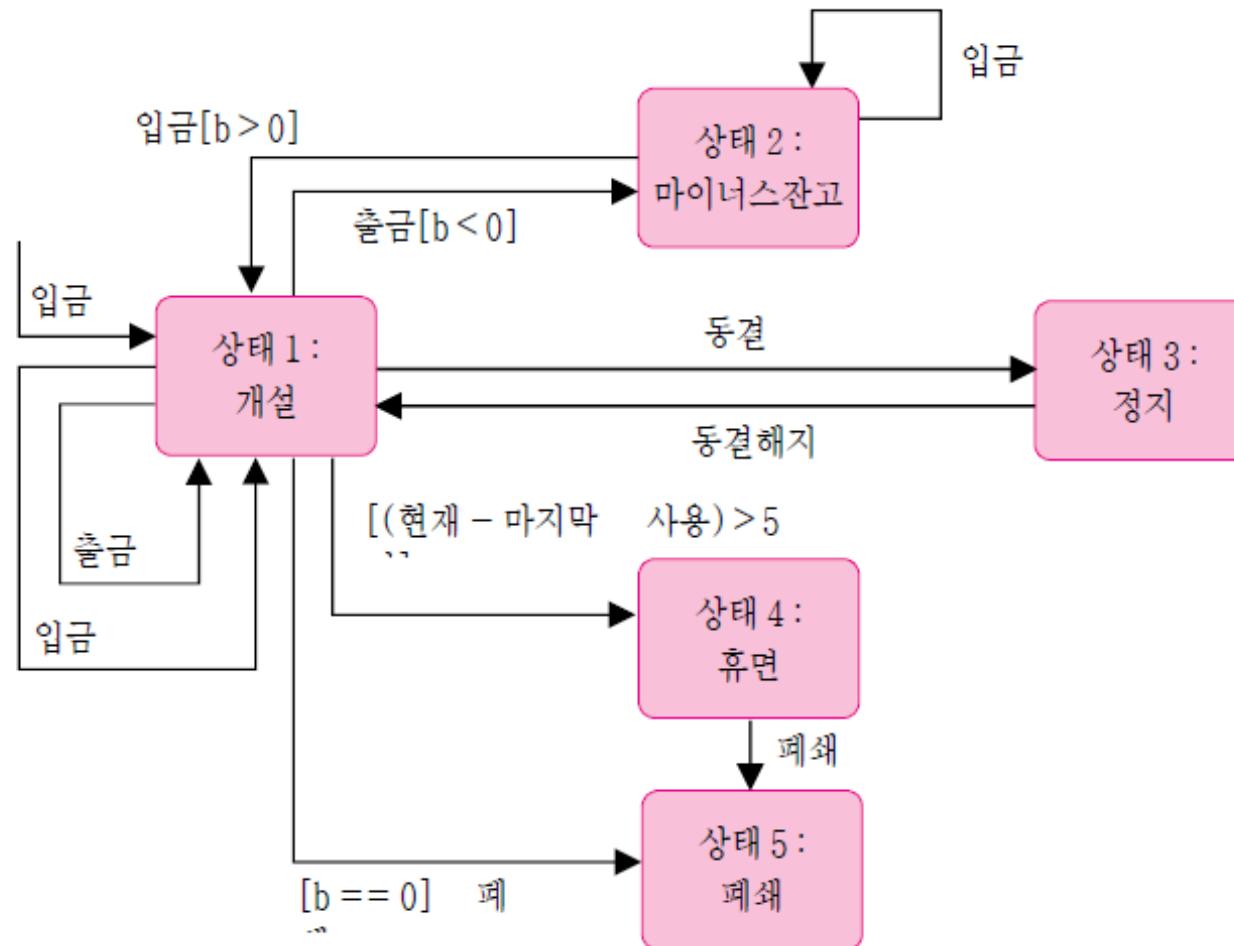
- 상태: 시스템의 과거 입력에 대한 영향을 표시
- 트랜지션: 이벤트에 대한 반응으로 시스템이 하나의 상태에서 다른 상태로 어떻게 변해 가는지를 나타냄
- 이벤트: 시스템에 대한 입력
- 액션: 이벤트에 대한 출력



상태 기반 테스팅 (2/3)

테스팅(Testing)

예제: 예금 계좌의 상태 모델





상태 모델에 대한 검증 기준

- 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)
→ 모든 트랜지션에 비해 훨씬 많은 테스트 케이스 생성
- 트랜지션 트리:** 테스트 케이스 집합이 모든 단순 경로를 만족시키는 기준

 모든 트랜지션에 대한
테스트 케이스 예제

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



In this chapter ...

테스팅(Testing)

9.1 테스팅 기초

9.2 블랙박스 테스팅

9.3 화이트박스 테스팅

9.4 객체지향 테스팅

9.5 통합 테스팅

9.6 시스템 및 인수 테스트

9.7 테스트 도구



통합 테스팅의 목적

- 시스템을 구성하는 모듈의 인터페이스와 결합을 테스트
- 시스템 전체의 기능과 성능을 테스트



통합 순서에 따라

- 빅뱅(big-bang) → 단번에 해치워...
- 하향식(top-down) → 위에서부터 아래로...
- 상향식(bottom-up) → 아래서부터 위로...
- 연쇄식(threads) → 중요한 것을 중심으로 동시에...



통합 시 필요한 소프트웨어

- **테스트 하니스(harness)**: 부분적인 테스트를 위하여 코드에 삽입하는 프로그램
→ 추후 삭제됨
- **스텝(stub)**: 시험 대상 모듈이 호출하는 모듈 대신에 만들어진 모의 서브루틴
- **드라이버(driver)**: 시험 대상 모듈을 호출하는 모의 모듈



- 모든 모듈을 한꺼번에 통합하여 테스트
- 단위 테스트에 많은 시간이 필요
- 시스템의 중요 부분과 부수적인 부분을 구별하지 않음
- 일정 계획에 융통성이 없음
- 오류가 있을 경우 어떤 모듈이 변경되어야 하는지 파악하기 어려움

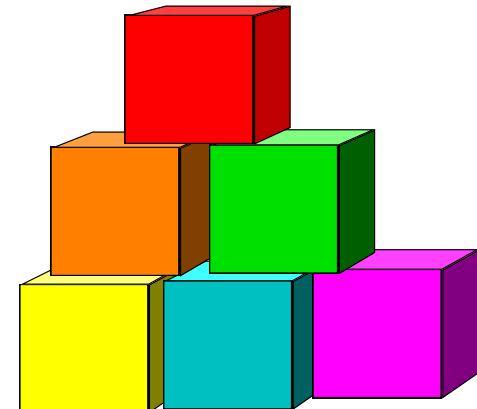




- ④ 시스템 구조도의 위에 있는 모듈부터 아래 층 모듈로 내려 오면서 통합
- ④ 점증적 통합이므로 하드웨어의 사용이 분산되고 오류의 원인을 찾아내기 쉬움
- ④ 상위층의 중요한 모듈을 먼저 시험
→ 시스템의 골격을 조기에 테스트 함
- ④ 스텝의 사용으로 시스템의 모양을 사용자에게 조기에 보여줄 수 있음
- ④ 프로그래머가 시스템의 작동에 대한 확신을 유지시킬 수 있음



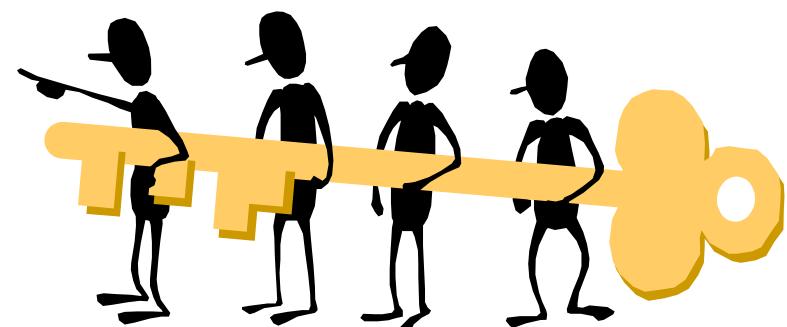
- ④ 최하위 모듈을 먼저 시험
- ④ 드라이버가 필요 (특성상, 스텝은 거의 필요치 않음)
- ④ 오류 발견이 쉽고 하드웨어의 사용을 분산
- ④ 하위층의 모듈을 상위층 보다 더 많이 시험
- ④ 테스트의 초기에 빼대가 갖추어 지지 않음





최선의 통합 방법

- ▣ 어느 정도의 기본 기능을 수행하는 모듈(thread)로부터 통합
 - ▣ 시스템의 중요한 기능을 담당하는 모듈부터 통합
 - ▣ 초기에 시스템을 골격을 보여주고 사용자의 의견을 받아 수정 가능
- 주요 테스트 작업을 동시에 진행할 수도 있음 (independent한 경우)





In this chapter ...

테스팅(Testing)

9.1 테스팅 기초

9.2 블랙박스 테스팅

9.3 화이트박스 테스팅

9.4 객체지향 테스팅

9.5 통합 테스팅

9.6 시스템 및 인수 테스트

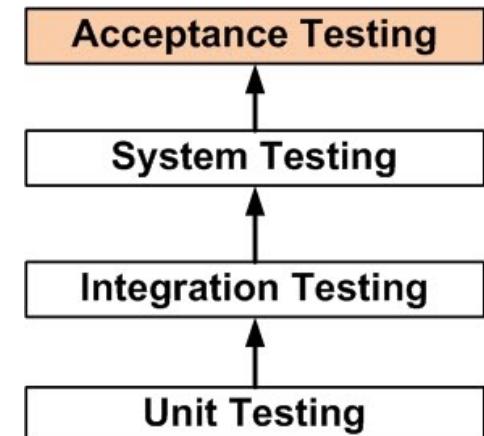
9.7 테스트 도구



컴포넌트 통합 후 수행하는 테스트 기법

테스트 종류

- 기능 테스트
- 성능 테스트
- 보안 테스트
- 사용성 테스트
- 인수 테스트
- 설치 테스트





기능 테스트 (1/2)

테스팅(Testing)

- 기능적 요구와 시스템의 차이를 발견하기 위한 테스트
 - 사용자와 관련되어 있으며 오류를 유발할 가능성이 많은 테스트를 선정
 - 사용사례 모델을 검토하고 오류 가능성이 높은 사용사례 인스턴스를 찾아냄
- 테스트 케이스
- 일반적인 사례
 - 예외적인 사례

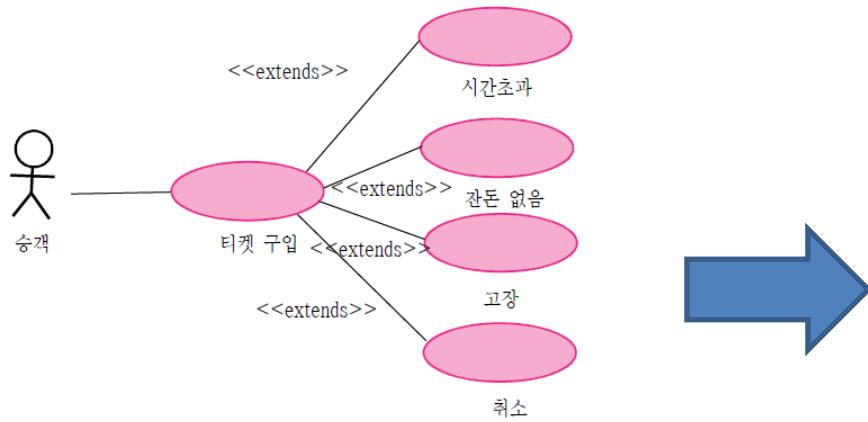




기능 테스트 (2/2)

테스팅(Testing)

기능 테스트 케이스 작성 과정



테스트 케이스 이름	티켓 정상 구입
시작 조건	<p>승객이 티켓 판매기 앞에 선다. 승객이 1000원을 가지고 있다.</p>
사건의 흐름	<ol style="list-style-type: none"> 승객이 여행할 목적지의 구간 2, 3, 1, 2를 선택한다. 티켓판매기가 800, 1000, 600, 800원을 차례로 표시한다. 승객이 1000원 지폐를 넣는다. 승객이 100원 동전 두 개를 받고 2구간의 표를 받는다. 승객이 또 한 번 1000원 지폐를 넣고 1~4까지의 과정을 반복한다. 승객이 500원 동전 2개를 넣고 1~4를 반복한다. 승객이 500원 동전과 100원 동전 세 개를 넣고 1~2를 반복한다. 승객이 1구간을 누르고 1000원을 넣는다. 티켓 판매기는 1구간 표를 발행하고 400원의 거스름돈을 배출한다. 승객이 3구간을 누르고 1000원을 넣는다. 티켓 판매기는 3구간 표를 발행한다.
종료조건	승객이 선택한 표를 받는다.

사용자례 이름	티켓구입
시작 조건	<p>승객이 티켓 판매기 앞에 선다. 승객이 티켓을 살만한 충분한 돈이 있다.</p>
사건의 흐름	<ol style="list-style-type: none"> 승객이 여행할 목적지의 구간을 선택한다. 승객이 여러 구간을 입력하였다 면 마지막 누른 버튼만을 티켓판매기가 인식한다. 티켓판매기가 총 금액을 표시한다. 승객이 돈을 넣는다. 승객이 충분한 돈을 넣기 전에 새로운 구간을 선택하였다면 티켓판매기는 승객이 넣은 모든 동전과 지폐를 돌려준다. 승객이 총액보다 많은 동전을 넣었다면 티켓판매기 초과분을 돌려준다. 티켓판매기가 표를 발행한다. 승객이 거스름돈과 표를 받는다.
종료조건	승객이 선택한 표를 받는다.



시스템의 여러 측면 테스트

- 작업 부하(workload): 시스템이 처리하고 생성하는 작업의 양
- 처리량(throughput): 트랜잭션의 수, 시간 당 처리하는 메일 수
- 반응 시간(response time): 시스템 요구를 처리하는 데 걸리는 총 시간
- 효율성: 주어진 작업 처리를 위해 사용하는 CPU시간과 메모리 같은 자원량의 비율

테스트 방법

- 스트레스 테스팅: 시스템 처리능력의 몇 배의 작업부하를 처리하고 견딜 수 있는지 측정
- 성능 테스팅: 정상적인 사용 환경에서 시스템의 성능을 측정하는데 사용
(시뮬레이션을 이용한 테스팅 가능)
- 보안 테스팅: 시스템의 보안 취약점을 찾아내려는 목적



인간 공학적인 목적 → 기능, 성능, 보안 테스트와 목적이 다름

테스트 목적

- 보고 느끼는 UI에 대한 결함
- 데이터 입력과 출력 디스플레이에 대한 결함
- 액터-시스템 사이의 동작 결함
- 오류 처리에 대한 결함
- 문서와 도움말에 대한 결함

GUI Testing





목적

- 시스템이 사용할 수 있도록 모든 준비가 되어 있는지를 보이는 것
- 사용할 모든 준비가 되었다는 것을 확인시켜 주는 작업



테스트 방법은 사용자가 선택



인수 테스트 방법의 대부분은 통합 테스트와 같음.
다른 점은 (시험 환경이 아닌) 사용자 환경에서 한다는 것



알파 테스트: 사내 고객들을 대상으로 시험



베타 테스트

- 고객의 사용 환경에서 시험
- 개발자 없이, 실제 사용 환경에 소프트웨어를 설치하여 시험



In this chapter ...

테스팅(Testing)

9.1 테스팅 기초

9.2 블랙박스 테스팅

9.3 화이트박스 테스팅

9.4 객체지향 테스팅

9.5 통합 테스팅

9.6 시스템 및 인수 테스트

9.7 테스트 도구



④ 테스트 작업을 자동화

도구 종류

- 코드 분석 도구
- 테스트 케이스 생성 도구
- 테스트 케이스 실행 도구
- 단위 테스트 도구





정적 분석 도구: 프로그램을 실행하지 않고 분석

- 코드 분석 도구: 원시 코드의 문법 검사
- 구조 검사 도구: 원시 코드의 그래프를 이용한 구조적인 결함 확인
- 데이터 분석 도구: 원시 코드를 검사하여 잘못된 링크나 데이터 정의의 충돌, 잘못된 데이터의 사용을 발견
- 순서 검사 도구: 이벤트 순서가 바르게 동작하는지 체크

동적 분석 도구: 프로그램을 실행하면서 분석

- 프로그램이 수행되는 동안 이벤트의 상태 파악을 위한 특정한 변수나 조건의 스냅샷(snapshot)을 생성
- 시스템의 성능 또는 기능에 영향을 주는 분기점(breakpoint) 파악에 도움
- 모듈의 호출 횟수나 수행된 문장 번호를 리스트로 만들어 줌



- ▣ 테스트 케이스 생성 도구: 테스트 케이스를 자동으로 생성
- ▣ 테스트 케이스 실행 도구: 대규모 테스트를 케이스를 자동으로 수행
- ▣ 단위 테스트 도구: 프로그램의 단위 테스트와 리그레션 테스트 수행
 - 예제: Junit, CUnit, CppUnit, Nunit, HtmlUnit, HttpUnit 등
(각기 Java, C, C++, C#, HTML, HTTP로 작성된 프로그램의 단위 테스트 수행)

JUnit 5



In this chapter ...

테스팅(Testing)

9.1 테스팅 기초

9.2 블랙박스 테스팅

9.3 화이트박스 테스팅

9.4 객체지향 테스팅

9.5 통합 테스팅

9.6 시스템 및 인수 테스트

9.7 테스트 도구