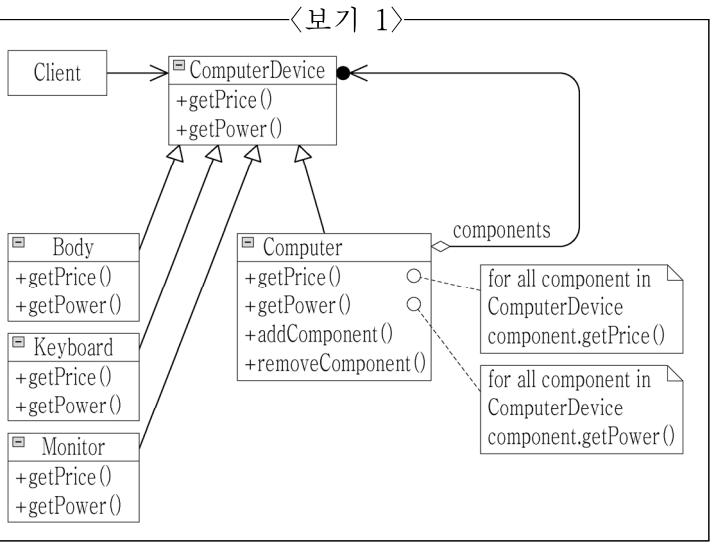
1. 〈보기 1〉은 컴포지트 디자인 패턴(composite design pattern)을 사용하여 컴퓨터의 가격 또는 전력 소모량을 계산할 목적으로 구성한 프로그램의 클래스 다이어그램이다. 〈보기 2〉는 새로운 부품 VR 헤드셋을 컴퓨터에 추가하기 위해 이 프로그램을 컴포지트 디자인 패턴에 따라 수정할 때, 최종 프로그램에 대한 설명이다. 〈보기 2〉에서 옳지 않은 것을 모두 고른 것은?

〈보기 1〉



〈보기 2〉

ㄱ. Body 클래스가 변경되었다.
ㄴ. Computer 클래스가 변경되었다.
ㄷ. ComputerDevice 클래스가 변경되었다.

① ㄱ ② ㄱ, ㄴ ③ ㄴ, ㄷ ④ ㄱ, ㄴ, ㄷ

2. 〈보기〉에서 설명하는 UP(Unified Process)의 단계 ㈎~㈐를 옳게 짝지은 것은?

〈보기〉

㈎ 소프트웨어 시스템을 개발 환경에서 사용자 환경으로 옮긴다.
㈏ 핵심 아키텍처를 구축하고 대부분의 요구사항을 명확히 정의한다.
㈐ 주요 요구사항들을 나열하고 전체적으로 10% 정도의 유스케이스(use case)를 상세히 작성한다.

	㈎	㈏	㈐
①	전이(transition)	정련(elaboration)	도입(inception)
②	도입(inception)	구축(construction)	정련(elaboration)
③	전이(transition)	구축(construction)	정련(elaboration)
④	구축(construction)	전이(transition)	도입(inception)

3. 프로젝트 계획을 세울 때 적용하는 기법들을 순서대로 바르게 나열한 것은?

① CPM – Gantt Chart – WBS
② WBS – CPM – Gantt Chart
③ CPM – WBS – Gantt Chart
④ WBS – Gantt Chart – CPM

4. 소프트웨어 요구사항 명세서(Software Requirement Specification)의 항목에 해당하지 않는 것은?

① 사용자 요구사항 정의 ② 시스템 아키텍처
③ 인터페이스 요구사항 ④ 소프트웨어 개발 일정

5. 기능적 요구사항(functional requirement)과 비기능적 요구사항(non-functional requirement)에 대한 설명으로 가장 옳지 않은 것은?

① 비기능적 요구사항은 기능적 요구사항보다 구현해야 하는 우선순위가 낮은 기능에 대한 요구사항을 포함한다.
② 동일한 소프트웨어에 대해서 여러 종류의 사용자가 존재할 수 있으며, 사용자 종류에 따라 별도의 기능적 요구사항을 가질 수 있다.
③ 기능적 요구사항은 소프트웨어 시스템을 통해 사용자가 얻고자 하는 서비스에 대한 조건으로, 사용자 입력에 따른 기대 출력에 대한 조건으로 표현할 수 있다.
④ 보안 요구사항, 성능 요구사항은 비기능적 요구사항이다.

6. 추상화와 구현을 분리하여, 실행시간에 동적으로 인터페이스의 다른 구현들을 사용할 수 있는 디자인 패턴은?

① adapter 패턴 ② bridge 패턴
③ proxy 패턴 ④ facade 패턴

7. 〈보기〉는 MyIF 인터페이스를 정의하고 이를 MyImpl 클래스로 구현한 자바 코드와 이들의 관계를 UML로 표현한 클래스 다이어그램이다. ㈎, ㈏에 들어갈 내용과 ㈐에 들어갈 그림을 옳게 짝지은 것은?

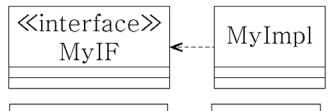
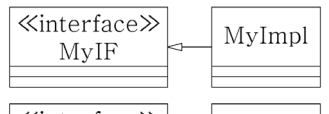
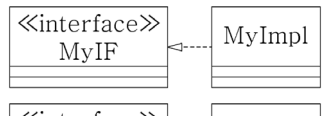
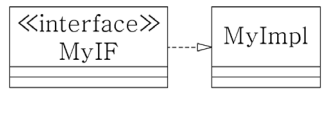
〈보기〉

• 코드

```
㈎ MyIF { … }
Class MyImpl ㈏ MyIF { … }
```

• UML 표기

㈐

	㈎	㈏	㈐
①	abstract	extends	
②	interface	extends	
③	interface	implements	
④	interface	implements	

8. 블랙박스 테스트 기법에 대한 설명으로 가장 옳은 것은?
① 경곗값 분석: 테스트 대상의 입출력값을 특정 클래스로 분할한 후 각각의 클래스로부터 대푯값을 추출하여 테스트 케이스를 생성하고 테스트 데이터의 범주를 식별한다.
② 인과 그래프: 입력값 조합의 경우의 수를 줄이기 위해 두 입력값의 조합을 통해 테스트 케이스를 도출하는 기법으로, 모든 경우의 수를 테스트하지 않고 2개 요소의 모든 조합을 확인할 수 있도록 테스트 케이스를 생성한다.
③ 페어와이즈 조합: 입력 데이터 간 관계가 출력에 영향을 미치는 상황을 체계적으로 분석하여 테스트 케이스를 생성한다.
④ 결정 테이블: 의사 결정의 여러 조합을 정하는 행위나 결과를 식별함으로써 시스템의 예상되는 행위를 파악하여 테스트 케이스를 생성한다.

9. 린(Lean) 소프트웨어 개발의 7가지 원칙에 해당하는 것을 〈보기〉에서 모두 고른 것은?

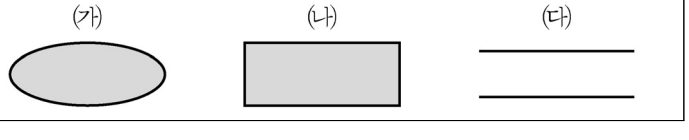
〈보기〉

ㄱ. 낭비를 제거하라
ㄴ. 학습을 확대하라
ㄷ. 최대한 빨리 배포하라
ㄹ. 최대한 빨리 결정하라
ㅁ. 팀장에게 권한을 집중시켜라

① ㄱ, ㄴ, ㄷ
② ㄱ, ㄷ, ㄹ
③ ㄴ, ㄹ, ㅁ
④ ㄱ, ㄴ, ㄷ, ㄹ

10. 〈보기〉는 데이터 흐름도에서 사용하는 기호들이다. ㈎~㈐ 기호의 의미를 반영한 용어를 옳게 짝지은 것은?

〈보기〉



	㈎	㈏	㈐
①	프로세스	데이터 저장소	외부 개체
②	프로세스	외부 개체	데이터 저장소
③	외부 개체	데이터 저장소	프로세스
④	외부 개체	프로세스	데이터 저장소

11. 관점지향 프로그래밍(Aspect Oriented Programming)에서 횡단 관심 모듈의 기능이 삽입되어 동작할 수 있는 실행 가능한 특정 위치를 의미하는 용어는?
① 어드바이스(advice)
② 위빙(weaving)
③ 조인 포인트(join point)
④ 포인트 컷(point-cut)

12. 상태 디자인 패턴(state design pattern)을 적용하여 전등(Light)의 상태를 관리하는 상태 클래스들을 〈보기 1〉과 같이 Java 프로그램으로 구현하였다. 〈보기 2〉와 같이 취침등(SLEEP) 상태를 추가하여 전등의 동작을 변경하고자 한다. 상태 디자인 패턴에 따라 작성한 최종 코드에 대한 설명으로 가장 옳지 않은 것은?

〈보기 1〉

REQ-1) 전등이 꺼진 상태에서 on 버튼을 누르면 켜진다.
REQ-2) 전등이 켜진 상태에서 off 버튼을 누르면 꺼진다.

```
public class State {
   public void on_button(Light light) {}
   public void off_button(Light light) {}
}

public class ON implements State {
   public void off_button(Light light) {
         light.setState(new OFF());
   }
}

public class OFF implements State {
   public void on_button(Light light) {
         light.setState(new ON());
   }
}
```

〈보기 2〉

REQ-3) 전등이 켜진 상태에서 on 버튼을 누르면 취침등 상태가 된다.
REQ-4) 취침등 상태에서 on 버튼을 누르면 다시 전등이 켜진다.
REQ-5) 취침등 상태에서 off 버튼을 누르면 꺼진다.

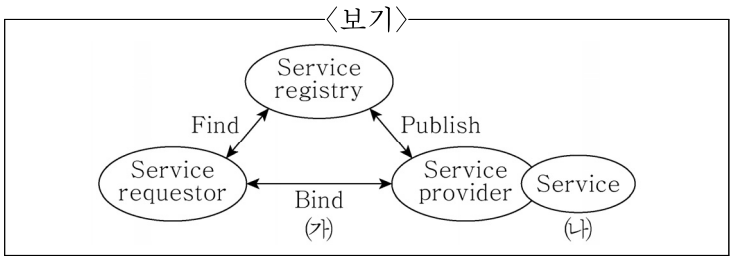
① ON 상태 클래스가 변경되었다.
② OFF 상태 클래스가 변경되었다.
③ SLEEP 상태 클래스가 추가되었다.
④ Light 클래스는 변경되지 않았다.

13. 익스트림 프로그래밍(XP)에서 좋은 사용자 스토리가 되기 위한 특성으로 가장 옳지 않은 것은?
① 각 사용자 스토리의 크기 또는 작업 소요 시간을 추정할 수 있어야 한다.
② 스토리는 테스트가 가능하도록 작성되어야 한다.
③ 스토리 간에 의존성이 강화되어야 한다.
④ 좋은 스토리는 그 크기가 작은 편이다.

14. 모듈화 설계에 대한 설명으로 가장 옳지 않은 것은?
① 모듈화가 잘 된 소프트웨어일수록 재사용 가능성이 높다.
② 모듈 사이의 불필요한 상호 교류를 최소화함으로써 결합력(coupling)을 낮출 수 있다.
③ 모듈 사이의 기능적 관련성이 강할수록 응집력(cohesion)이 높다.
④ 기능 응집력(functional cohesion)이 높은 모듈에서는 모든 요소가 하나의 기능 구현을 위해 구성되어 있다.

15. 〈보기〉는 웹 기반의 서비스 지향 아키텍처의 구성으로 (가)는 메시지를 교환하기 위한 프로토콜이고, (나)는 서비스 정의 언어이다. (가), (나)에 알맞은 내용을 순서대로 나열한 것은?

〈보기〉

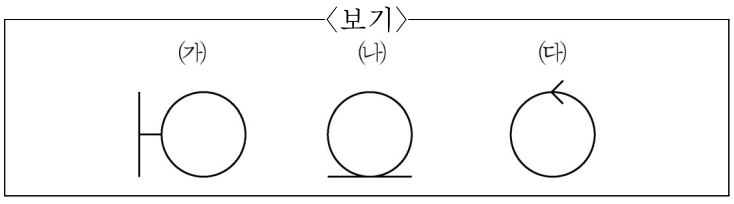


	(가)	(나)		(가)	(나)
①	SOAP	UDDI	②	WSDL	UDDI
③	UDDI	SOAP	④	SOAP	WSDL

16. 소프트웨어 프로세스 모델 중 폭포수형 모델(waterfall model)이 제시하는 개발 단계의 선후 관계에서 먼저 실행되어야 하는 단계와 나중에 실행해야 할 단계로 가장 옳지 않은 것은? (단, 화살표 왼편의 단계가 오른쪽보다 먼저 실행해야 함을 나타내며, 그 중간에 다른 단계가 있을 수 있다.)
① 기능 요구사항 분석 → 설계
② 설계 → 시스템 시험
③ 통합 → 단위 시험
④ 코딩 → 배포, 설치 및 운영

17. 〈보기〉는 UML 스테레오타입(stereotype)이 적용된 클래스를 표현한 것이다. (가)~(다)에 해당하는 스테레오타입을 옳게 짝지은 것은?

〈보기〉



	(가)	(나)	(다)
①	boundary	control	entity
②	boundary	entity	control
③	control	entity	boundary
④	control	boundary	entity

18. 객체지향 품질 척도의 설명으로 옳은 것을 〈보기〉에서 모두 고른 것은?

〈보기〉

ㄱ. 객체 클래스 사이의 결합(CBO, Coupling Between Object Classes): 해당 클래스가 의존하고 있는 클래스의 개수로, CBO가 클수록 재사용이 어렵다.
ㄴ. 클래스당 가중 메소드(WMC, Weighted Method per Class): 클래스의 메소드 개수에 그 클래스의 메소드가 호출하는 메소드의 개수를 더한 값으로, WMC가 클수록 책임이 커져 분리할 필요성이 있다.
ㄷ. 메소드의 응집 결핍(LCOM, Lack of Cohesion in Methods): 해당 클래스의 속성을 공유하지 않는 메소드 쌍의 수로, LCOM이 크면 클래스 응집도가 떨어진다.
ㄹ. 상속 트리의 깊이(DIT, Depth of Inheritance): 상속 트리의 루트로부터 해당 클래스까지 가장 깊은 상속 경로이다.

① ㄴ, ㄹ ② ㄱ, ㄴ, ㄷ
③ ㄱ, ㄷ, ㄹ ④ ㄴ, ㄷ, ㄹ

19. 리팩토링(refactoring)과 관련한 설명으로 가장 옳지 않은 것은?
① 리팩토링을 통해 성능 개선이 이루어지기도 한다.
② 소프트웨어에 대한 가독성과 이해도를 높일 수 있다.
③ 내부 구조의 변경 없이 외부 인터페이스와 기능을 변경할 수 있다.
④ 리팩토링이 필요한 코드의 징후를 코드 스멜(code smell)이라고 한다.

20. 〈보기〉와 관련한 소프트웨어 아키텍처로 가장 옳은 것은?

〈보기〉

사물인터넷의 한 가지 응용 시스템인 홈 오토메이션은 허락 없이 출입구 또는 창문이 열리는 상황을 센서가 감지하면 자동으로 알람 경보를 내거나 문자 메시지를 거주자에게 전달한다.

① 계층형 아키텍처(layered architecture)
② 이벤트 기반 아키텍처(event-based architecture)
③ 파이프 필터 아키텍처(pipe and filter architecture)
④ 저장소 아키텍처(repository architecture)

이 면은 여백입니다.