소프트웨어공학

문 1. 애자일 선언문은 애자일 방법론이 추구하고 있는 가치를 요약하고 있다. 애자일 선언문의 내용으로 옳은 것은?

① 포괄적인 문서보다는 제대로 동작하는 소프트웨어에 더 가치를 둔다.
② 고객과의 협력보다는 계약 협상에 더 가치를 둔다.
③ 변화에 대응하는 것보다는 계획을 따르는 것에 더 가치를 둔다.
④ 개인과 상호작용보다는 도구나 프로세스에 더 가치를 둔다.

문 2. 다음에서 설명하는 아키텍처는?

> ○ 시스템은 여러 개의 실행 모듈 또는 컴포넌트로 구성되며 각 컴포넌트는 독립된 기능을 제공한다.
> ○ 시스템의 서비스는 컴포넌트들의 기능을 조합하여 제공된다.
> ○ 컴포넌트가 상호 작용하는 데이터베이스는 다른 컴포넌트의 데이터베이스와 독립적이다.
> ○ 각 컴포넌트의 실패는 격리되며 일부가 실패해도 전체 시스템이 중단되지 않는다.
> ○ 각 컴포넌트를 독립적으로 개발하고 배포할 수 있으므로 지속적 통합과 지속적 배포가 가능하다.

① 마이크로서비스(microservice) 아키텍처
② 마스터-슬레이브(master-slave) 아키텍처
③ 저장소(repository) 아키텍처
④ 피어-투-피어(peer-to-peer) 아키텍처

문 3. 다음은 무엇에 사용되는 도구인가?

> ○ JUnit
> ○ Mockito
> ○ JMeter

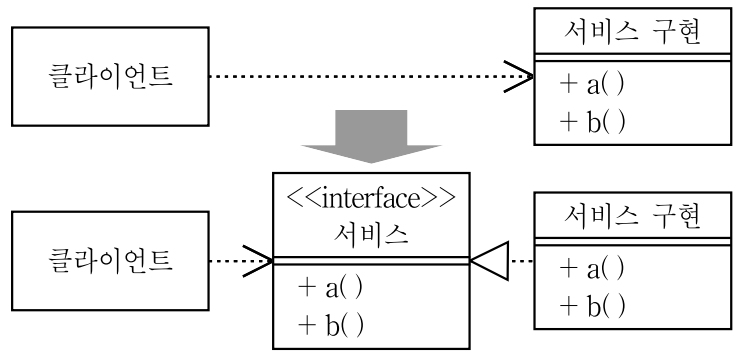
① 버전 관리
② 프로젝트 관리
③ 소프트웨어 설계
④ 소프트웨어 테스팅

문 4. CMMI 프로세스 성숙도 수준(maturity level) 중 관리(Managed) 단계의 특징으로만 묶은 것은?

> ㄱ. 기본적인 프로젝트 관리 프로세스가 정의되어 비용, 일정, 기능 등을 추적할 수 있다.
> ㄴ. 새로운 프로젝트에 대한 계획과 관리가 이전의 성공한 프로젝트에 근거하여 이루어진다.
> ㄷ. 조직의 소프트웨어 프로세스를 전담하는 소프트웨어공학 프로세스 그룹이 있다.
> ㄹ. 프로세스 개선을 지속적으로 추진하여 프로세스 능력 수준을 높인다.

① ㄱ, ㄴ
② ㄴ, ㄷ
③ ㄱ, ㄴ, ㄹ
④ ㄴ, ㄷ, ㄹ

문 5. 그림과 같이 서비스 구현 클래스의 a( ), b( ) 연산을 사용하는 클라이언트 클래스가 서비스 구현 클래스에 직접 의존하는 관계에서 클라이언트 클래스가 서비스 인터페이스에 의존하고 서비스 구현 클래스는 서비스 인터페이스를 구현하는 것으로 설계를 변경하였다. 다음 중 이와 가장 관련이 깊은 SOLID 설계 원칙은?



① 인터페이스 분리 원칙(Interface Segregation Principle)
② 의존관계 역전 원칙(Dependency Inversion Principle)
③ 리스코프 치환 원칙(Liskov Substitution Principle)
④ 단일 책임 원칙(Single Responsibility Principle)

문 6. 어떤 웹 서비스 시스템은 다음과 같은 특징을 가지고 있다. 이 시스템과 관련하여 ISO/IEC 9126 품질 특성 중에서 개선할 필요가 있는 것은?

> ○ 온라인/오프라인 도움말을 제공하지 않는다.
> ○ 시스템이 제공하는 기능을 메뉴명으로 이해하기 어렵다.
> ○ 모든 웹 페이지에서 홈페이지로 바로 가는 '홈 버튼'이 제공되지 않아 이전 페이지로 이동하는 '뒤로 가기 버튼'을 이용하여 여러 단계를 거쳐 홈페이지로 갈 수 밖에 없다.

① 효율성(efficiency)
② 사용성(usability)
③ 이식성(portability)
④ 유지보수성(maintainability)

문 7. 스크럼(Scrum)의 제품 백로그(product backlog)에 대한 설명으로 옳지 않은 것은?

① 제품 백로그에 있는 업무 목록은 프로젝트를 수행하는 동안 수정되고 정제된다.
② 제품 백로그의 업무 중 높은 우선순위를 갖는 항목부터 개발한다.
③ 제품 백로그에 있는 업무의 우선순위를 결정한 후에는 변경하지 않는다.
④ 제품 책임자(product owner)가 제품 백로그를 관리한다.

문 8. 개발자 A는 <명세>에 따라 <코드>를 작성한 후 테스팅을 수행하였다. A는 100% 문장 커버리지를 달성하면서 동시에 프로그램의 오류를 발견할 수 있었다. A가 사용한 테스트 입력은? (단, 단축 연산(short-circuit evaluation)은 수행하지 않는다)

<명 세>

> 두 정수를 입력 받아 두 정수 중 적어도 하나가 음수이면 두 정수의 곱을 반환하고 그렇지 않다면 두 정수의 합을 반환한다.

<코 드>

```
int foo(int v1, int v2) {
    int v3 = v1*v2;
    if (v1 >= 0 || v2 >= 0)
        v3 = v1+v2;
    return v3;
}
```

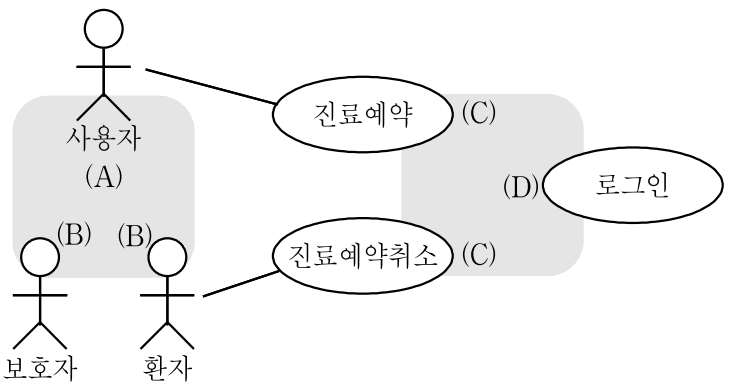
① (v1 = 0, v2 = 0)
② (v1 = −2, v2 = −2)
③ (v1 = 2, v2 = 0)
④ (v1 = −2, v2 = 2)

문 9. <그림>은 <보기>에서 기술하고 있는 병원 진료 시스템의 일부 명세에 해당하는 유스케이스 다이어그램이다. <그림>에서 '사용자'와 '보호자', '사용자'와 '환자' 사이에 들어갈 관계와 '진료예약'과 '로그인', '진료예약취소'와 '로그인' 사이에 들어갈 관계로 옳은 것은? (단, A, B, C, D는 다이어그램의 구성 요소가 아니며 관계의 방향을 지시한다)

<보 기>

병원 진료 시스템 사용자로 환자와 보호자가 있다. 환자는 진료예약이나 진료예약취소를 모두 할 수 있는 반면에 보호자는 진료예약만이 가능하다. 또한 진료예약이나 진료예약취소를 하기 위해서는 반드시 로그인 절차가 필요하다.

<그 림>



① (A) ◁—— (B), (C) ·····<<include>>·····▷ (D)

② (A) ◁—— (B), (C) ·····<<extend>>·····◁ (D)

③ (A) ——▷ (B), (C) ·····<<include>>·····▷ (D)

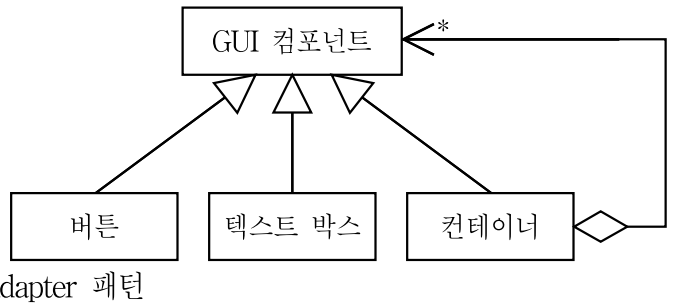
④ (A) ——▷ (B), (C) ·····<<extend>>·····◁ (D)

문 10. 다음은 어떤 시스템의 유지보수를 위한 요구사항의 일부이다. 아래 요구사항에 따라 수행한 유지보수 활동이 기존 기능에 영향을 끼쳤는지 알아보기 위해 수행하는 테스팅은?

Req-01	윈도우 환경에서 동작하는 시스템을 리눅스 환경에서도 동작하도록 한다.
Req-02	SMS 문자 발송 기능의 오류를 수정한다.
Req-03	논리 흐름을 보다 이해하기 쉽도록 코드 구조를 개선한다.

① 회귀 테스팅(regression testing)
② 사용성 테스팅(usability testing)
③ 성능 테스팅(performance testing)
④ 보안성 테스팅(security testing)

문 11. 다음 클래스 다이어그램으로 표현한 설계에서 사용한 디자인 패턴은?



① Adapter 패턴
② Composite 패턴
③ Observer 패턴
④ Factory Method 패턴

문 12. 어떤 소프트웨어의 고장 간 평균 시간(MTBF; Mean Time Between Failures)이 2,500시간이고 평균 수리 시간(MTTR; Mean Time To Repair)이 100시간일 때 평균 가동 시간(MTTF; Mean Time To Failure)과 가용성(availability)은? (단, 소수점 이하에서 반올림한다)

① 2,600시간, 96 %
② 2,600시간, 92 %
③ 2,400시간, 96 %
④ 2,400시간, 92 %

문 13. 형상관리의 형상 제어(configuration control) 활동에서 수행하는 작업으로만 묶은 것은?

ㄱ. 형상 항목과 형상 식별자 선정
ㄴ. 변경 요청사항에 대한 심사 및 변경 실시
ㄷ. 변경 내용을 확인하고 베이스라인 수립
ㄹ. 형상 관리 계획서대로 형상관리가 진행되고 있는지 검증

① ㄱ, ㄴ
② ㄴ, ㄷ
③ ㄴ, ㄹ
④ ㄷ, ㄹ

문 14. 다음은 리팩토링 전후의 코드 변화이다. 적용된 리팩토링 기법에 해당하는 것은?

```
public void myMethod(int n) {
    if (isEven(n))
        System.out.println("Even");
    else
        System.out.println("Odd");
}
private boolean isEven(int number) {
    return number % 2 ==  0;
}
```

⇩

```
public void myMethod(int n) {
    if (n % 2 == 0)
        System.out.println("Even");
    else
        System.out.println("Odd");
}
```

① 메소드 은폐(Hide Method)
② 메소드 추출(Extract Method)
③ 메소드 내용 직접 삽입(Inline Method)
④ 메소드를 매개변수로 전환(Parameterize Method)

문 15. 소프트웨어 테스팅 문서에 관한 국제 표준은?

① IEEE 1042
② IEEE 829
③ IEEE 828
④ ISO/IEC 9899

문 16. 다음은 두 정수를 입력받아 큰 값을 출력하는 프로그램이다. 이 프로그램에서 caller 함수와 max 함수 간의 결합도와 max 함수의 응집도를 바르게 나열한 것은?

```
void caller(int x, int y) {
    int maxVal = max(x, y);
    printf("%d", maxVal);
}

int max(int v1, int v2) {
    if (v1 > v2) return v1;
    else return v2;
}
```

① 제어(control) 결합도, 논리적(logical) 응집도
② 제어(control) 결합도, 기능적(functional) 응집도
③ 데이터(data) 결합도, 논리적(logical) 응집도
④ 데이터(data) 결합도, 기능적(functional) 응집도

문 17. 다음 용어에 대한 설명으로 옳은 것은?

① 사용자 스토리(User Story)는 개발자 관점에서 프로젝트를 통해 구현하고 싶은 기능을 상세하게 표현한 알고리즘이다.
② 베이스라인(Baseline)은 공식적인 변경절차 없이 언제라도 개발자가 변경할 수 있는 상태에 있는 산출물의 집합이다.
③ 기능 점수(Function Point)는 시스템을 구현한 기술에 의존적이고 개발자에 의해 식별되는 기능에 기반하여 시스템의 크기를 측정하는 척도이다.
④ 순환 복잡도(Cyclomatic Complexity)는 원시 코드의 복잡도를 정량적으로 평가하는 척도이며 기본 경로(basis path) 테스팅과 밀접한 관련이 있다.

문 18. <명세>는 어떤 과목의 통과 여부를 결정하는 프로그램에 대한 명세이다. <코드>의 프로그램은 <명세>에 따라 작성하였지만 오류가 있다. <코드>의 오류를 검출할 수 있는 테스팅 기법과 테스트 입력을 바르게 짝 지은 것은?

<명 세>

입력 점수가 70보다 크거나 같으면 통과이고 그렇지 않으면 통과하지 못한다. 점수는 0이상 100이하 범위를 갖는 정수형이다. 프로그램의 반환 값이 0이면 통과, 1이면 통과하지 못함, −1이면 입력이 범위를 벗어났음을 나타낸다.

<코 드>

```
int passOrNot(int score) {
    if ((score > 100) || (score < 0)) return -1;
    if (score > 70) return 0;
    else return 1;
}
```

① 동등 분할 기법, 80
② 동등 분할 기법, 50
③ 경계 값 분석 기법, 100
④ 경계 값 분석 기법, 70

문 19. 유스케이스 분석 기법에서는 유스케이스를 바탕으로 세 가지 유형의 클래스들을 도출한다. 다음 중에서 이에 해당하지 않는 클래스 유형은?

① 제어(control) 클래스
② 경계(boundary) 클래스
③ 싱글톤(singleton) 클래스
④ 엔티티(entity) 클래스

문 20. 다음 Java 코드의 MyEmployeeService, EmployeeService, EmployeeDao, Employee 클래스를 UML 클래스 다이어그램으로 표현하였을 때 클래스 다이어그램에 나타나지 않는 관계는? (단, MyEmployeeService, EmployeeService, EmployeeDao, Employee 클래스는 이 외 다른 클래스와 아무 관계가 없다)

```
class MyEmployeeService extends EmployeeService{
    private EmployeeDao employeeDao;
    public void setEmployeeDao(EmployeeDao dao){
        this.employeeDao = dao;
    }
    public Employee getEmployee(String id){
        return employeeDao.get(id);
    }
    public void addEmployee(Employee emp){
        employeeDao.insert(emp);
    }
}
class EmployeeService { ... }
class EmployeeDao { ... }
class Employee { ... }
```

① 연관(association)
② 의존(dependency)
③ 일반화(generalization)
④ 실체화(realization)