3주차

진행 상태	시작 전
완료 확인	☐

▼ Java 컴파일 과정을 설명하세요.

Java 컴파일 과정은 크게 3단계로 진행됩니다.

첫째, 자바 소스코드(.java 파일)를 javac 컴파일러가 바이트코드(.class 파일)로 변환합니다.

둘째, 클래스 로더가 이 바이트코드를 JVM 메모리에 로드합니다.

셋째, JIT 컴파일러가 바이트코드를 실제 기계어로 변환하여 실행합니다.

이런 과정을 거치기 때문에 Java는 "한 번 작성하면 어디서든 실행된다"는 플랫폼 독립성을 가질 수 있습니다.

▼ 교착상태가 무엇이며, 4가지 조건은?

교착상태는 두 개 이상의 프로세스가 서로 상대방이 가진 자원을 기다리면서
무한정 대기하게 되는 상황을 말합니다.

교착상태가 발생하는 4가지 조건은 다음과 같습니다.

첫째, 상호 배제입니다. 한 번에 하나의 프로세스만 자원을 사용할 수 있어야 합니다.

둘째, 점유와 대기입니다. 프로세스가 자원을 보유한 상태에서 다른 자원을 기다려야 합니다.

셋째, 비선점입니다. 다른 프로세스의 자원을 강제로 빼앗을 수 없어야 합니다.

넷째, 순환 대기입니다. 프로세스들이 원형으로 서로의 자원을 기다리는 상황이어야 합니다.

이 네 가지 조건이 모두 만족될 때 교착상태가 발생합니다.

▼ 인덱스란 무엇인가요? 꼭 사용해야하나요?

인덱스는 데이터베이스에서 검색 속도를 빠르게 하기 위한 자료구조입니다.
책의 목차처럼 데이터의 위치를 빠르게 찾을 수 있게 도와줍니다.

장점으로는 SELECT 쿼리의 성능이 크게 향상되고, WHERE절이나 ORDER BY절 처리가 빨라집니다.

단점으로는 추가 저장공간이 필요하고,
INSERT, UPDATE, DELETE 시 인덱스도 함께 수정해야 해서 성능이 저하될 수 있습

니다.

꼭 사용해야 하는지에 대해서는 상황에 따라 다릅니다.
자주 검색하는 컬럼에는 인덱스를 생성하는 것이 좋지만, 데이터가 적거나 자주 변경되는 테이블에서는 오히려 성능에 악영향을 줄 수 있습니다.

▼ 정규화에 대해 설명해주세요.

정규화는 데이터베이스에서 데이터의 중복을 최소화하고 데이터 무결성을 보장하기 위해 테이블을 분해하는 과정입니다.

주요 정규화 단계를 말씀드리면,

1차 정규화는 테이블의 모든 속성이 원자값, 즉 더 이상 나눌 수 없는 값만 가지도록 하는 것입니다.

2차 정규화는 부분적 함수 종속을 제거하는 것으로, 기본키가 아닌 속성이 기본키 전체에 종속되도록 하는 것입니다.

3차 정규화는 이행적 함수 종속을 제거하는 것으로, 기본키가 아닌 속성들 간의 종속성을 없애는 것입니다.

정규화의 장점은 데이터 중복을 최소화하고 일관성을 보장하는 것이고, 단점은 테이블 간의 조인 연산이 많아져서 성능이 저하될 수 있다는 점입니다.

실무에서는 보통 3차 정규화까지 적용하는 것이 일반적입니다.

▼ TCP 3-way handshaking이란?

TCP 3-way handshaking은 TCP 연결을 설정하기 위한 3단계 과정입니다.

첫 번째 단계에서 클라이언트가 서버에게 SYN 패킷을 보내며 연결을 요청합니다.
"연결하고 싶습니다"라는 의미입니다.

두 번째 단계에서 서버가 클라이언트에게 SYN+ACK 패킷을 보내며 연결 요청을 수락하고 자신도 연결을 요청합니다.
"알겠습니다, 저도 연결하고 싶습니다"라는 의미입니다.

세 번째 단계에서 클라이언트가 서버에게 ACK 패킷을 보내며 최종 연결을 확인합니다.
"확인했습니다, 이제 통신을 시작합시다"라는 의미입니다.

이 과정을 통해서 양방향 통신이 가능한 안정적인 연결이 설정됩니다.

▼ GET와 POST의 차이는?

GET과 POST는 HTTP 메서드로, 사용 목적과 특징이 다릅니다.

GET 방식은 주로 데이터를 조회할 때 사용합니다.
데이터가 URL에 쿼리 파라미터로 노출되고, 브라우저에서 캐싱이 가능하며 히스토리에

도 남습니다.
다만 URL 길이 제한이 있어서 대용량 데이터 전송에는 적합하지 않습니다.

POST 방식은 주로 데이터를 생성하거나 수정할 때 사용합니다.
데이터가 HTTP Body에 포함되어 전송되므로 보안성이 더 좋고, 캐싱되지 않으며 히스토리에도 남지 않습니다. 또한 데이터 크기에 제한이 없습니다.

예를 들어, 게시글을 조회할 때는 GET을 사용하고, 게시글을 작성하거나 수정할 때는 POST를 사용합니다.

▼ MVC 패턴이란?

MVC 패턴은 Model-View-Controller의 줄임말로, 애플리케이션을 세 가지 역할로 분리하는 아키텍처 패턴입니다.

Model은 데이터와 비즈니스 로직을 담당합니다. 데이터베이스와의 연동이나 데이터 처리 작업을 수행합니다.

View는 사용자 인터페이스를 담당합니다. 사용자에게 화면을 보여주고 입력을 받는 역할을 합니다.

Controller는 Model과 View 사이의 중재자 역할을 합니다. 사용자의 요청을 받아서 처리하고, 적절한 Model과 View를 연결해줍니다.

MVC 패턴의 장점은 각 역할이 분리되어 있어서 유지보수가 쉽고, 코드의 재사용성이 높으며, 여러 개발자가 협업하기에도 좋습니다.

Spring MVC가 대표적인 예시입니다.

▼ 객체지향이란 무엇인가요?

객체지향은 프로그램을 객체들의 집합으로 보고 설계하는 프로그래밍 패러다임입니다.

객체지향의 4대 특징을 말씀드리면,

첫째, 캡슐화입니다. 데이터와 그 데이터를 처리하는 메서드를 하나로 묶고, 외부에서의 직접적인 접근을 제한하는 것입니다.

둘째, 상속입니다. 기존 클래스의 속성과 메서드를 새로운 클래스가 물려받아서 재사용할 수 있게 하는 것입니다.

셋째, 다형성입니다. 같은 인터페이스나 메서드 호출로 서로 다른 동작을 수행할 수 있게 하는 것입니다.

넷째, 추상화입니다. 복잡한 내부 구현을 숨기고 필요한 기능만 외부에 노출하는 것입니다.

객체지향의 장점은 코드의 재사용성이 높고, 유지보수가 용이하며, 모듈화를 통해 협업 효율성이 증대된다는 점입니다.

▼ OAuth란 무엇인가요?

OAuth는 제3자 서비스를 통한 인증을 위한 개방형 표준 프로토콜입니다.
사용자가 직접 아이디와 비밀번호를 입력하지 않고도 다른 서비스의 계정을 통해 로그인할 수 있게 해줍니다.

동작 과정을 간단히 설명드리면, 먼저 사용자가 웹사이트에서 "구글로 로그인" 버튼을 클릭합니다.

그러면 구글의 인증 페이지로 리다이렉트됩니다.

사용자가 구글에서 로그인을 완료하면, 구글이 인증 코드를 원래 웹사이트로 전달합니다.

웹사이트는 이 인증 코드로 구글에서 액세스 토큰을 받아서 사용자 정보에 접근할 수 있게 됩니다.

OAuth의 장점은 사용자가 여러 사이트마다 비밀번호를 관리할 필요가 없고, 보안성이 향상되며, 사용자 편의성이 크게 증대된다는 점입니다.

카카오톡, 네이버, 구글 로그인이 대표적인 예시입니다.

▼ CSRF와 XSS의 차이는 무엇인가요?

CSRF와 XSS는 모두 웹 보안 취약점이지만 공격 방식이 다릅니다.

CSRF는 **사용자가 의도하지 않은 요청**을 보내도록 하는 공격입니다.
예를 들어, 사용자가 은행 사이트에 로그인한 상태에서 악성 링크를 클릭하면 본인 모르게 계좌이체가 실행되는 경우입니다.
CSRF는 CSRF 토큰을 사용하거나 Referer 헤더를 검증해서 방어할 수 있습니다.

XSS는 **악성 스크립트를 웹페이지에 삽입**하는 공격입니다.
예를 들어, 게시판에 댓글을 달 때 스크립트 태그를 포함시켜서 다른 사용자들의 쿠키 정보를 탈취하는 경우입니다.
XSS는 사용자 입력값을 검증하고 출력할 때 인코딩해서 방어할 수 있습니다.

핵심 차이점은 CSRF는 '요청을 조작'하는 공격이고, XSS는 '스크립트를 삽입'하는 공격이라는 점입니다.