다음은 Lec08 강의자료에서 중요한 세 가지 개념에 기반한 간단한 퀴즈와 답변이다.

1. 멀티스레딩의 정의

질문: 멀티스레딩(Multithreading)이란 무엇이며, Java에서 이를 활용하면 얻을 수 있는 주요 이점은 무엇인가?

답:

멀티스레딩은 하나의 프로그램에서 여러 스레드를 생성해 동시에 작업을 실행할 수 있는 능력이다.

Java에서 멀티스레딩은 CPU 자원을 효율적으로 활용하고, 작업을 병렬로 실행함으로써 성능을 개선하며, GUI 애플리케이션에서 사용자 응답성을 높이는 데 유용하다.

2. 동시성(Concurrency)과 병렬성(Parallelism)의 차이

질문: 동시성과 병렬성의 차이를 설명하고, 각각의 적합한 사례를 제시하시오.

답:

- 동시성은 여러 작업이 시간이 겹치도록 실행되는 것을 의미하며, 단일 코어에서도 가능하다. 예: 파일 읽기와 데이터 처리 병행.

- 병렬성은 여러 작업이 실제로 동시에 실행되는 것을 의미하며, 다중 코어 프로세서에서 가능하다. 예: 서버에서 여러 클라이언트 요청을 처리.

3. 스레드 상태 전이

질문: 스레드의 라이프사이클 상태를 설명하고, New 상태에서 Terminated 상태로 전환되는 과정을 서술하시오.

답:

- New: 스레드가 생성되었으나 아직 시작되지 않은 상태.

- Runnable: start() 호출 후 실행 준비 상태.

- Running: 스케줄러가 CPU를 할당하여 실행되는 상태.

- Terminated: run() 메서드 실행이 완료되어 종료된 상태.

New에서 Runnable로 전환되려면 start() 메서드가 호출되어야 하며, Running 상태는 스케줄러에 의해 결정된다. Task가 완료되면 Terminated 상태로 이동한다.

다음은 Lec09 강의자료에서 중요한 세 가지 개념에 기반한 간단한 퀴즈와 답변이다.

1. 동기화(Synchronization)의 필요성

질문: 동기화가 필요한 이유를 설명하고, 동기화되지 않은 공유 자원 접근 시 발생할 수 있는 문제를 기술하시오.

답:

동기화는 다중 스레드 환경에서 공유 데이터를 안전하게 보호하기 위해 필요하다. 동기화되지 않은 경우 Race Condition이나 Data Inconsistency 문제가 발생할 수 있다. 예를 들어, 두 스레드가 동시에 동일한 변수의 값을 증가시키려 할 때, 데이터가 덮어써져 잘못된 결과를 초래할 수 있다.

2. Synchronized 메서드와 블록의 차이

질문: Synchronized 메서드와 Synchronized 블록의 차이를 설명하고, Synchronized 블록이 더 유리한 경우를 서술하시오.

답:

- Synchronized 메서드는 메서드 전체를 동기화하지만, Synchronized 블록은 코드의 특정 부분만 동기화한다.

- Synchronized 블록은 필요한 부분만 잠그기 때문에 성능이 더 우수하다. 예: 잔액 업데이트 시 계좌 전체에 대한 동기화 대신 특정 연산만 보호.

3. ReentrantLock의 장점

질문: ReentrantLock이 synchronized 키워드와 비교하여 제공하는 주요 장점을 두 가지 이상 서술하시오.

답:

- 명시적 잠금 제어: lock()과 unlock() 메서드를 통해 잠금을 명확히 제어할 수 있다.

- 공정성 설정: ReentrantLock은 요청 순서에 따라 쓰레드 접근을 제어하는 공정 모드를 지원한다.

예를 들어, 여러 스레드가 동일한 리소스를 사용하려 할 때 선입선출 방식으로 공정하게 접근 권한을 부여할 수 있다.

다음은 Lec10 강의자료에서 중요한 세 가지 개념에 기반한 간단한 퀴즈와 답변이다.

1. Concurrent Collections

질문: Concurrent Collections의 주요 목적을 설명하고, `ConcurrentHashMap`과 `CopyOnWriteArrayList`의 차이점을 기술하시오.

답:

Concurrent Collections은 멀티스레드 환경에서 안전하게 데이터를 관리하기 위해 설계된 컬렉션이다.

- `ConcurrentHashMap`: 읽기 작업은 높은 동시성을 제공하며, 쓰기 작업은 세그먼트 단위로 동기화된다. 주로 실시간 데이터 업데이트가 필요한 환경에서 사용된다.

- `CopyOnWriteArrayList`: 데이터를 수정할 때 전체 배열의 복사본을 생성하여 안전한 반복을 보장한다. 읽기가 쓰기보다 훨씬 많은 경우에 적합하다.

2. CountDownLatch의 역할

질문: CountDownLatch가 무엇이며, 멀티스레드 환경에서 어떻게 사용되는지 설명하시오.

답:

CountDownLatch는 특정 작업이 완료될 때까지 다른 스레드가 대기하도록 동기화하는 도구이다.

예: 자원 로딩이 완료되기 전까지 메인 스레드가 대기하도록 설정할 수 있다. 각 작업 스레드가 `countDown()`을 호출해 카운트를 줄이며, 카운트가 0이 되면 대기가 해제된다.

3. Semaphore의 정의와 사용 사례

질문: Semaphore가 무엇이며, 어떤 상황에서 유용한지 설명하시오.

답:

Semaphore는 공유 자원에 대한 접근을 제한하는 동기화 도구로, 일정 수의 스레드만 자원에 접근하도록 허용한다.

사용 사례: 데이터베이스 연결 풀 관리, 제한된 개수의 프린터 사용 제어.

Semaphore semaphore = new Semaphore(2);

semaphore.acquire(); // 자원 접근 시작

semaphore.release(); // 자원 접근 종료

다음은 Lec11 강의자료에서 중요한 세 가지 개념에 기반한 간단한 퀴즈와 답변이다.

1. Java 네트워킹의 정의

질문: Java 네트워킹이란 무엇이며, Java의 네트워킹 API가 제공하는 주요 기능은 무엇인가?

답:

Java 네트워킹은 장치나 애플리케이션 간의 데이터 교환을 가능하게 하는 연결을 생성하는 것을 의미한다.

Java의 네트워킹 API는 `java.net` 패키지를 통해 제공되며, 클라이언트-서버 통신과 P2P 데이터 교환을 지원한다. 이를 통해 낮은 수준의 네트워크 코드를 작성하지 않고도 네트워크 애플리케이션을 개발할 수 있다.

2. Socket과 ServerSocket의 차이

질문: Socket과 ServerSocket의 역할과 차이를 설명하시오.

답:

- Socket은 네트워크 상의 두 장치 간의 데이터를 송수신하기 위한 클라이언트 측 통신 종단점이다.

- ServerSocket은 서버 측에서 클라이언트 연결 요청을 대기하고 수락하는 데 사용된다.

Socket은 주로 클라이언트 애플리케이션에서, ServerSocket은 서버 애플리케이션에서 사용된다.

3. HttpURLConnection의 주요 메서드

질문: HttpURLConnection 클래스의 주요 메서드와 그 역할을 설명하시오.

답:

- `connect()`: URL에 연결을 생성한다.

- `getResponseCode()`: HTTP 응답 코드를 반환한다 (예: 200, 404).

- `getInputStream()`: 서버의 응답 데이터를 읽을 수 있는 입력 스트림을 반환한다.

- `disconnect()`: 연결을 닫아 자원을 해제한다.

HttpURLConnection은 웹 API 호출 및 인터넷 리소스와의 통신에 사용된다.

다음은 Lec12 강의자료에서 중요한 세 가지 개념에 기반한 간단한 퀴즈와 답변이다.

1. JDBC의 정의

질문: JDBC(Java Database Connectivity)란 무엇이며, 이를 사용하는 주요 이유는 무엇인가?

JDBC는 Java 애플리케이션과 관계형 데이터베이스를 연결하고 상호작용하기 위한 API이다. JDBC를 사용하면 SQL 쿼리를 실행하고 데이터를 검색하거나 수정할 수 있어 데이터 중심 애플리케이션 개발에 필수적이다.

2. CRUD 작업의 정의

질문: CRUD 작업이란 무엇이며, 각각의 SQL 명령어와 역할을 설명하시오.

CRUD는 데이터베이스에서 수행되는 기본 작업을 나타낸다.

- Create: INSERT 명령어를 사용하여 데이터를 추가한다.

- Read: SELECT 명령어로 데이터를 검색한다.

- Update: UPDATE 명령어로 데이터를 수정한다.

- Delete: DELETE 명령어로 데이터를 제거한다.

3. JDBC 연결 설정

질문: JDBC를 사용하여 데이터베이스에 연결하기 위한 기본 단계를 설명하시오.

- 데이터베이스 서버 설치: MySQL, PostgreSQL 등 데이터베이스를 설치하고 실행한다.

- JDBC 드라이버 다운로드: 사용 중인 데이터베이스에 적합한 JDBC 드라이버를 다운로드한다.

- 연결 설정: `DriverManager.getConnection()` 메서드를 사용해 URL, 사용자 이름, 비밀번호로 데이터베이스에 연결한다.

String url = "jdbc:mysql://localhost:3306/mydatabase";

String username = "root";

String password = "password";

Connection connection = DriverManager.getConnection(url, username, password);

다음은 Lec13 강의자료에서 중요한 세 가지 개념에 기반한 간단한 퀴즈와 답변이다.

1. JOIN의 역할

질문: SQL에서 JOIN이란 무엇이며, INNER JOIN과 LEFT JOIN의 차이를 설명하시오.

JOIN은 두 개 이상의 테이블을 관련된 열을 기준으로 결합하여 데이터를 조회하는 SQL 연산이다.

- INNER JOIN: 두 테이블에서 매칭되는 행만 결과에 포함된다.

- LEFT JOIN: 왼쪽 테이블의 모든 행과 오른쪽 테이블에서 매칭되는 행을 포함하며, 매칭되지 않는 오른쪽 테이블 값은 NULL로 표시된다.

2. GROUP BY 절의 목적

질문: SQL에서 GROUP BY 절의 역할을 설명하고, 이를 활용한 예를 작성하시오.

GROUP BY는 특정 열을 기준으로 데이터를 그룹화하여 집계 함수(SUM, AVG 등)를 적용할 수 있도록 한다.

예: 각 부서의 평균 급여를 계산하는 쿼리.

SELECT department, AVG(salary) as avg\_salary

FROM employees

GROUP BY department

HAVING avg\_salary > 5000

ORDER BY avg\_salary DESC;

3. LIMIT 절의 역할

질문: LIMIT 절이 SQL에서 제공하는 기능을 설명하고, 이를 사용하는 상황을 제시하시오.

LIMIT 절은 결과에서 반환되는 행 수를 제한한다. 대규모 데이터셋에서 페이지네이션이나 상위 n개의 데이터를 조회할 때 유용하다.

예: 최근 입사한 직원 5명을 조회하는 쿼리.

SELECT \* FROM employees

ORDER BY hire\_date DESC

LIMIT 5;

다음은 Lec14 강의자료에서 중요한 세 가지 개념에 기반한 간단한 퀴즈와 답변이다.

1. PreparedStatement의 역할

질문: PreparedStatement가 무엇이며, SQL Injection을 방지할 수 있는 이유를 설명하시오.

PreparedStatement는 미리 컴파일된 SQL 문으로, 파라미터화된 쿼리를 지원하여 효율성과 보안을 제공한다.

SQL Injection을 방지하는 이유는, 데이터 값과 SQL 구조를 분리하여 입력값이 실행 가능한 SQL 명령어로 해석되지 않도록 하기 때문이다.

2. 트랜잭션(Transaction)의 ACID 특성

질문: 트랜잭션의 ACID 특성을 설명하고 각각의 역할을 서술하시오.

- Atomicity: 모든 작업이 완료되거나 모두 취소되어야 한다.

- Consistency: 트랜잭션 실행 전후 데이터베이스의 일관성을 유지한다.

- Isolation: 트랜잭션 간 간섭을 방지하여 독립적으로 실행되도록 보장한다.

- Durability: 트랜잭션이 성공적으로 완료되면 결과가 영구적으로 저장된다.

3. Manual Transaction Management

질문: JDBC에서 수동 트랜잭션 관리 방법과 주요 메서드를 설명하시오.

- `setAutoCommit(false)`: 자동 커밋을 비활성화하여 수동 트랜잭션을 시작한다.

- `commit()`: 트랜잭션을 완료하고 변경 내용을 데이터베이스에 영구적으로 반영한다.

- `rollback()`: 오류 발생 시 트랜잭션의 변경 내용을 취소하고 이전 상태로 복구한다.

connection.setAutoCommit(false);

try {

statement.executeUpdate("UPDATE accounts SET balance = balance - 100 WHERE id = 1");

statement.executeUpdate("UPDATE accounts SET balance = balance + 100 WHERE id = 2");

connection.commit(); // 성공적으로 완료

} catch (SQLException e) {

connection.rollback(); // 오류 발생 시 롤백

}