제06장

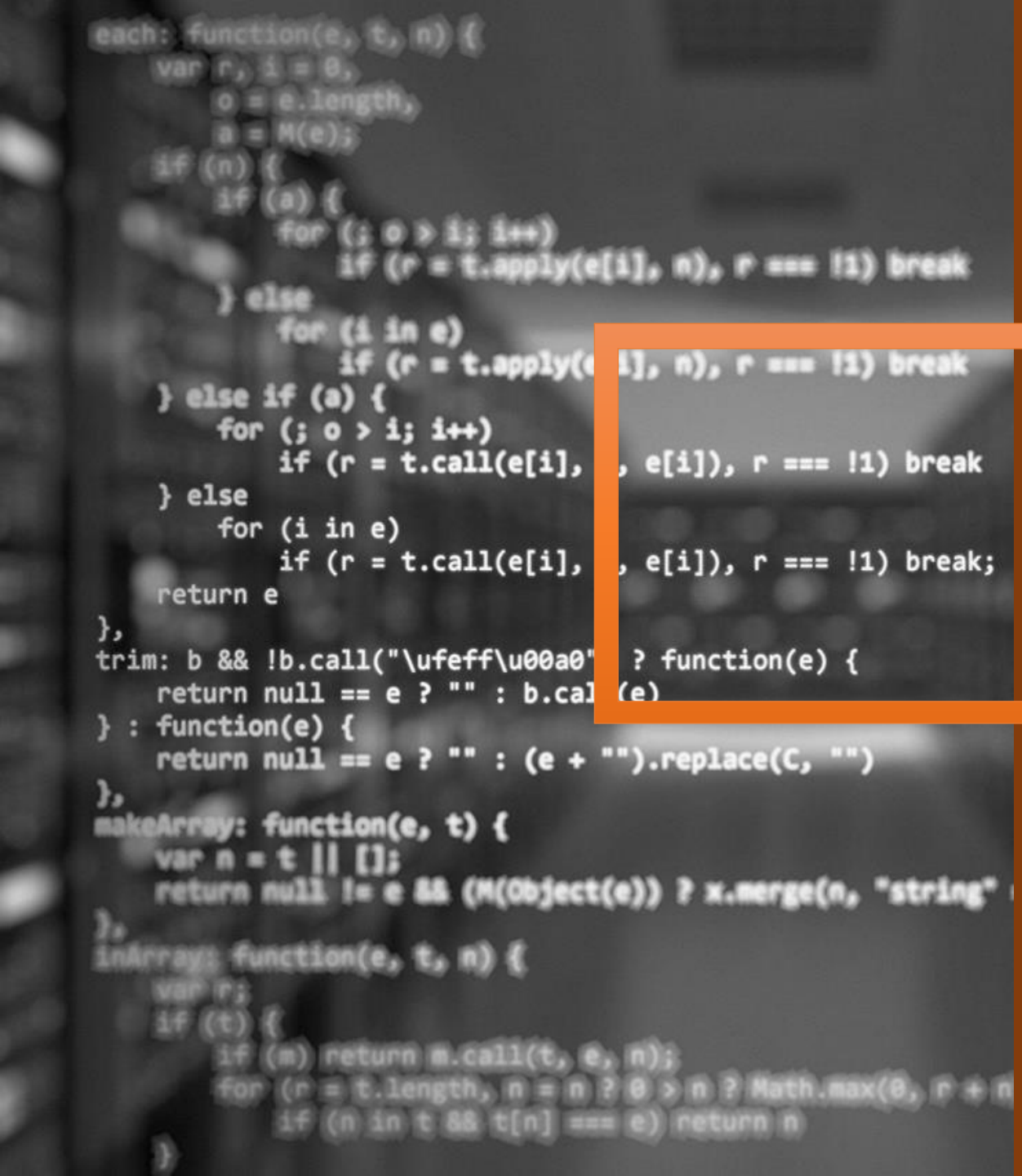
JDBC

Spring

학습목표

1. JDBC의 개념에 대해서 알 수 있다.
2. 스프링에서 JDBC를 활용할 수 있다.

목차



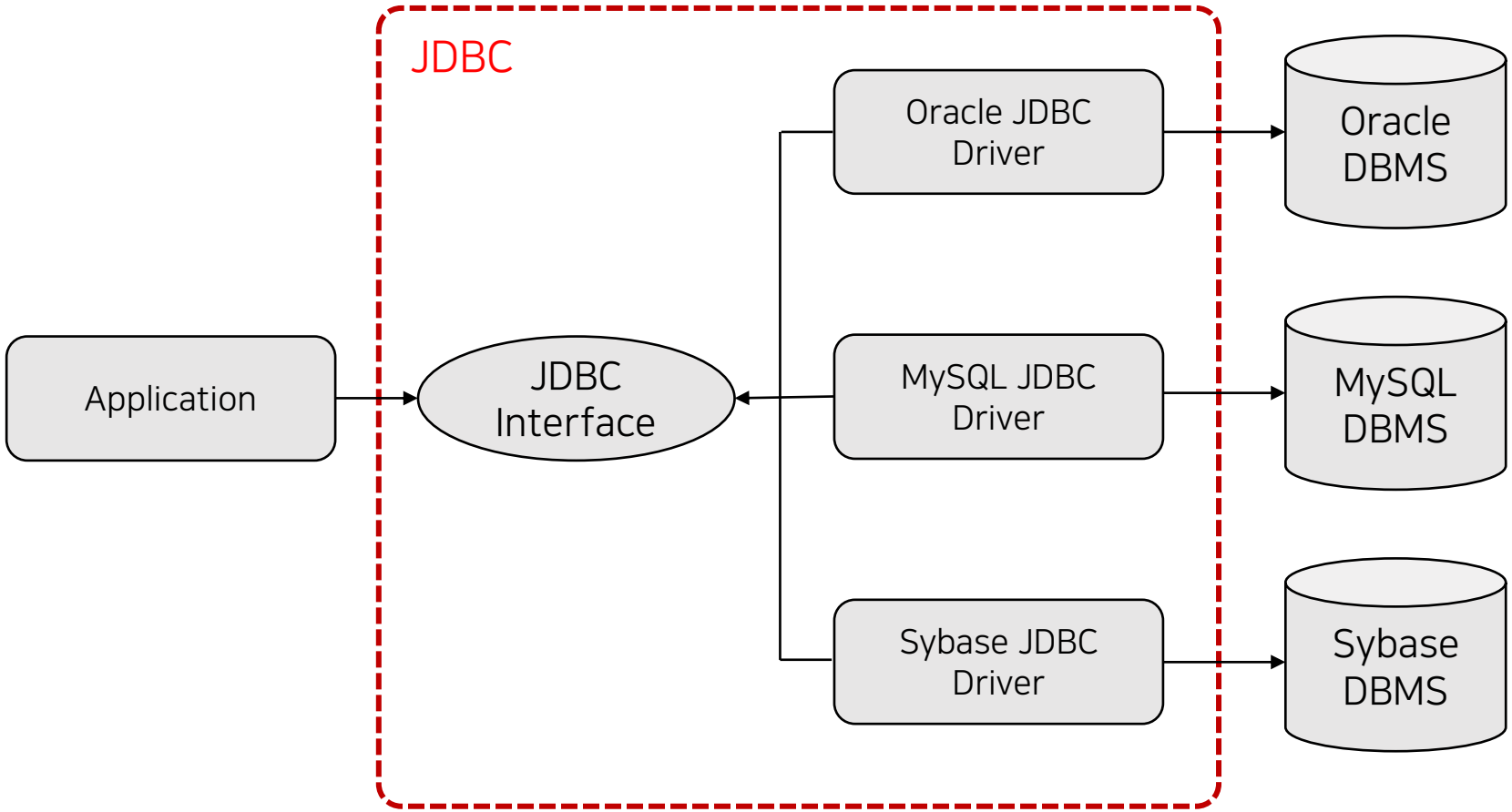
1. JDBC 개념

JDBC

- JDBC
  - Java DataBase Connectivity
  - 자바에서 데이터베이스에 접근할 수 있게 해주는 Programming API
  - JDBC API 이용 시 DBMS의 종류에 상관 없이 하나의 방법으로 작업 진행 가능
  - DBMS마다 서로 다른 SQL 문법으로 인한 문제 해결
  - java.sql 패키지를 통해서 JDBC API를 제공함

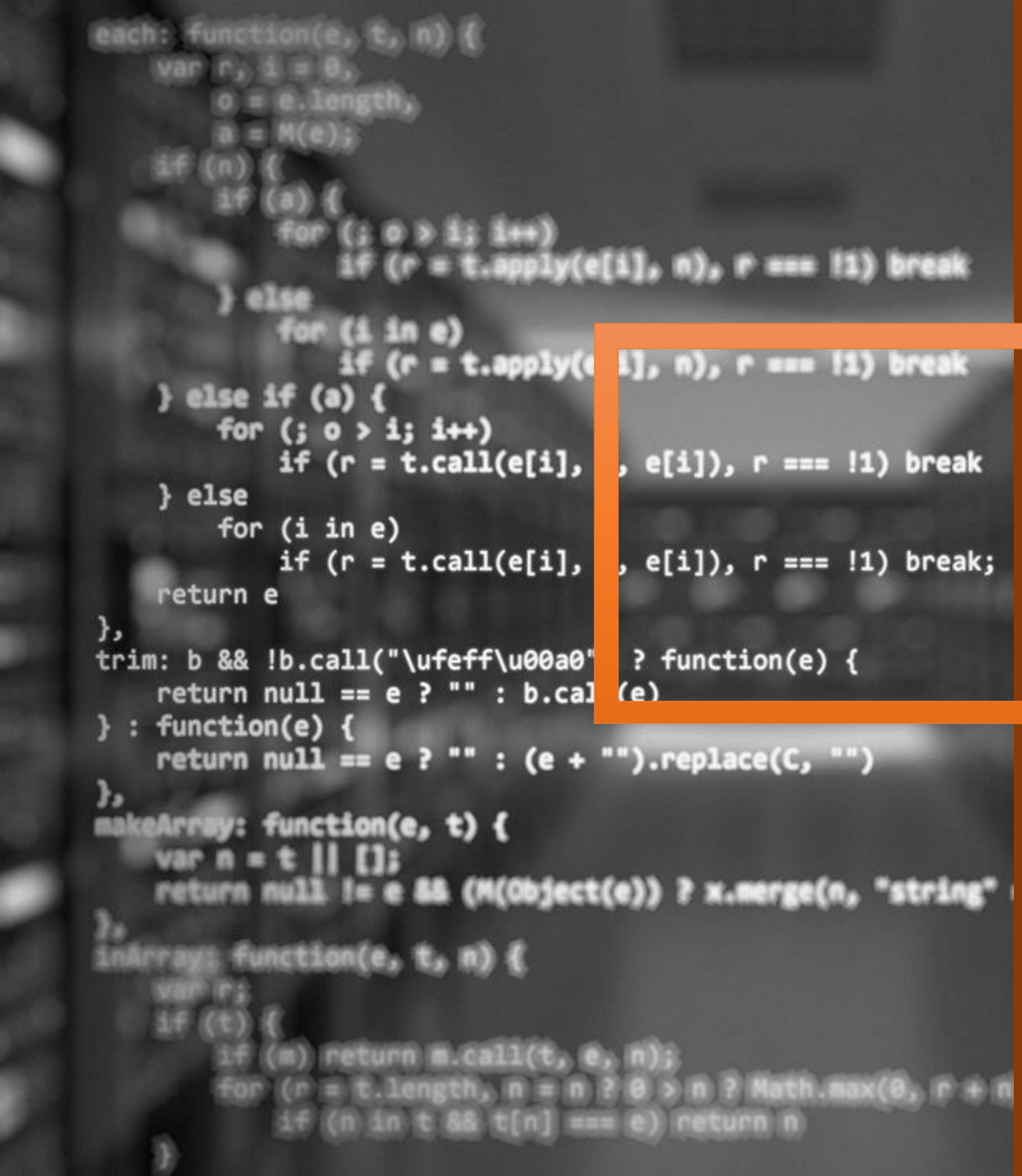
JDBC

- JDBC 개념도



JDBC Driver

- JDBC Driver
  - Java와 DataBase 연동 시 사용하는 라이브러리
  - DBMS에 따라서 서로 다른 Driver를 사용함
  - 기본적으로 Driver는 DB Vendor에서 제공함
  - 주요 종류
    - Oracle : ojdbc
    - MySQL : mysql-connector-j



2. JDBC 활용

Dependency

- 주요 JDBC Driver Dependency
  - Oracle (ojdbc11)

```
<dependency>
  <groupId>com.oracle.database.jdbc</groupId>
  <artifactId>ojdbc11</artifactId>
  <version>23.3.0.23.09</version>
</dependency>
```

  - MySQL (mysql-connector-j)

```
<dependency>
  <groupId>com.mysql</groupId>
  <artifactId>mysql-connector-j</artifactId>
  <version>8.0.33</version>
</dependency>
```

JDBC Driver

- JDBC Driver 로드
  - JDBC Driver 사용을 위해서 메모리에 올리는 작업을 의미함
  - 사용하려는 Driver 클래스의 이름을 알아야 로드 가능

- 주요 Driver 클래스 이름
  - Oracle : `oracle.jdbc.driver.OracleDriver`
  - MySQL : `com.mysql.cj.jdbc.Driver`
  - MSSQL : `com.microsoft.sqlserver.jdbc.Driver`
  - MariaDB : `org.mariadb.jdbc.Driver`

- Driver 로드 코드 예시 (ClassNotFountException 예외 핸들링 필요)

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

접속 URL

- 접속 URL
  - Java에서 DataBase에 접속하기 위해서 필요한 URL
  - DataBase에 따라 서로 다른 접속 URL을 가지고 있음

- 주요 DataBase 접속 URL
  - Oracle
    - `jdbc:oracle:thin:@127.0.0.1:1521:xe`
  - MySQL
    - `jdbc:mysql://127.0.0.1:3306/db_name`
  - MariaDB
    - `jdbc:mariadb://127.0.0.1:3306/db_name`

DriverManager

- DriverManager
  - 패키지 : java.sql
  - DataBase에 접속하기 위한 Connection 객체를 만드는 클래스
  - 3가지 접속 정보가 필요함
    - 접속 URL
    - 계정명
    - 비밀번호

Connection

- Connection
  - 패키지 : java.sql
  - DataBase와 연결된 커넥션을 의미함
  - DataBase 작업을 할 때 가장 먼저 Connection 객체를 생성해야 함
  - 사용이 완료된 후에는 닫아주어야 함

Connection

- Connection 생성
  - DriverManager 클래스의 getConnection() 메소드 호출을 이용함

```
Connection conn = DriverManager.getConnection(
                  "jdbc:mysql://127.0.0.1:3306/db_name?serverTimezone=Asia/Seoul"
                , "root"
                , "1234");
```

PreparedStatement

- PreparedStatement
  - 패키지 : java.sql
  - 쿼리문을 실행하는 인터페이스
  - PreparedStatement 객체 생성 이전에 반드시 쿼리문이 먼저 만들어져 있어야 함
  - 쿼리문에 인자 값을 전달해야 하는 경우 place holder(? 마크)를 이용하여 쿼리문을 작성함
  - Connection 인터페이스의 prepareStatement() 메소드를 호출할 때 쿼리문을 전달하면 PreparedStatement 객체를 생성할 수 있음
  - 사용이 완료된 후에는 닫아주어야 함

PreparedStatement

- PreparedStatement 사용 예시

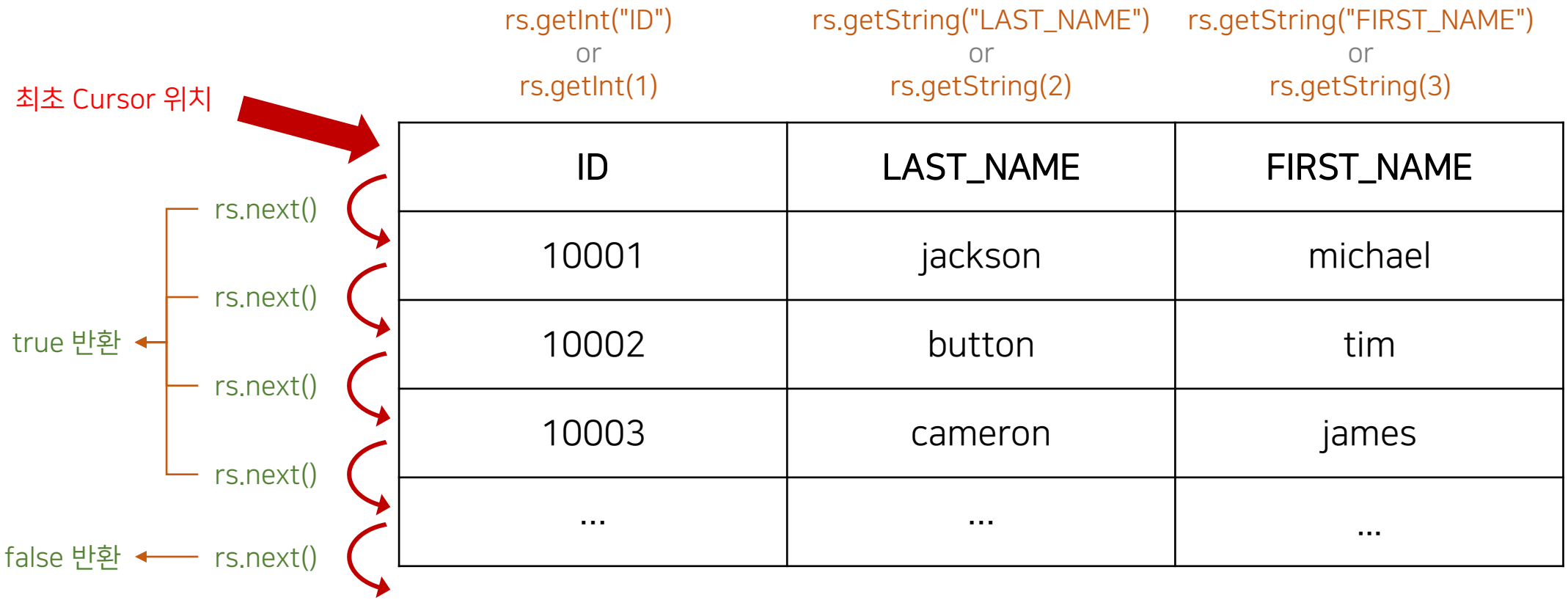
```
try {
    String sql = "INSERT INTO tbl_member VALUES (null, ?, ?)";
    ps = conn.prepareStatement(sql);
    ps.setString(1, id);
    ps.setString(2, password);
} catch (SQLException e) {
    e.printStackTrace();
}
```

ResultSet

- ResultSet
  - 패키지 : java.sql
  - SELECT문을 사용한 질의에서 사용되는 인터페이스
  - SELECT문에 의해 생성된 결과 집합(테이블)에 대한 정보를 담고 있음
  - 커서(Cursor)를 이용해 순차적으로 테이블의 데이터를 행(Row) 단위로 조회할 수 있음
  - 커서(Cursor) 이동을 위해서는 next() 메소드를 반드시 호출해야 함
  - 사용이 완료된 후에는 닫아주어야 함

ResultSet

- Cursor 이동의 이해



ResultSet

- ResultSet 사용 예시

```
try {
    String sql = "SELECT no, id, password, join_dt FROM tbl_member";
    ps = conn.prepareStatement(sql);
    rs = ps.executeQuery();
    while(rs.next()) {
        int no = rs.getInt("no");
        String id = rs.getString("id");
        String password = rs.getString("password");
        Date join_dt = rs.getDate("join_dt");
    }
} catch (Exception e) {
    e.printStackTrace();
}
```