(01-4)DAO 설계

Gaillardia / Flo:be(플로비) - 플라워 카페 정보 제공 및 상품 주문/예약 서비스

- UserDAO.java

```
package model.dao;

import java.sql.ResultSet;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import model.Customer;
import model.Seller;

public class UserDAO {
	private JDBCUtil jdbcUtil = null;

	public UserDAO() {
		jdbcUtil = new JDBCUtil();
	}

	// Customer 회원 가입
	public int createCustomer(Customer customer) throws SQLException {
		String sql = "INSERT INTO customer VALUES (?, ?, ?, ?, ?, ?)";
		Object[] param = new Object[] {customer.getCustomerId(), customer.getPwd(), customer.getName(),
			customer.getPhone(), customer.getEmail(), customer.getAddress()};

		jdbcUtil.setSqlAndParameters(sql, param);

		try {
			int result = jdbcUtil.executeUpdate();
			return result;
		} catch (Exception ex) {
			jdbcUtil.rollback();
			ex.printStackTrace();
		} finally {
			jdbcUtil.commit();
			jdbcUtil.close();
		}
		return 0;
	}
```

```
// Seller 회원 가입
public int createSeller(Seller seller) throws SQLException {
		String sql = "INSERT INTO seller VALUES (?, ?, ?, ?, ?)";
		Object[] param = new Object[] {seller.getSellerId(), seller.getPwd(), seller.getName(),
			seller.getPhone(), seller.getEmail()};
		jdbcUtil.setSqlAndParameters(sql, param);

		try {
			int result = jdbcUtil.executeUpdate();
			return result;
		} catch (Exception ex) {
			jdbcUtil.rollback();
			ex.printStackTrace();
		} finally {
			jdbcUtil.commit();
			jdbcUtil.close();
		}
		return 0;
}

// Customer 회원 정보 수정
public int updateCustomer(Customer customer) throws SQLException {
		String sql = "UPDATE customer "
			+ "SET pwd=?, name=?, phone=?, email=?, address=? "
			+ "WHERE customerId=?";
		Object[] param = new Object[] {customer.getPwd(), customer.getName(), customer.getPhone(),
			customer.getEmail(), customer.getAddress(), customer.getCustomerId()};

		jdbcUtil.setSqlAndParameters(sql, param);

		try {
			int result = jdbcUtil.executeUpdate();
			return result;
		} catch (Exception ex) {
			jdbcUtil.rollback();
			ex.printStackTrace();
		}
		finally {
			jdbcUtil.commit();
```

```
                jdbcUtil.close();
        }
        return 0;
}

// Seller 회원 정보 수정
public int updateSeller(Seller seller) throws SQLException {
        String sql = "UPDATE seller "
                + "SET pwd=?, name=?, phone=?, email=? "
                + "WHERE sellerId=?";
        Object[] param = new Object[] {seller.getPwd(), seller.getName(), seller.getPhone(),
                seller.getEmail(), seller.getSellerId()};
        jdbcUtil.setSqlAndParameters(sql, param);

        try {
                int result = jdbcUtil.executeUpdate();
                return result;
        } catch (Exception ex) {
                jdbcUtil.rollback();
                ex.printStackTrace();
        }
        finally {
                jdbcUtil.commit();
                jdbcUtil.close();
        }
        return 0;
}

// Customer 로그인
public int loginCustomer(String customerId, String pwd) throws SQLException {
        String sql = "SELECT pwd FROM customer WHERE customerId=?";
        jdbcUtil.setSqlAndParameters(sql, new Object[] {customerId});

        try {
                int result = jdbcUtil.executeUpdate();

                if(result.next()) {
                        if(result.getString(1).equals(pwd))
                                return 1; //사용자 로그인 성공
```

```
                    else
                        return 0; //비밀번호 불일치
                }
                return -1; //존재하지 않는 아이디
            } catch (Exception ex) {
                jdbcUtil.rollback();
                ex.printStackTrace();
            }
            finally {
                jdbcUtil.commit();
                jdbcUtil.close();
            }
            return 0;
}

// Seller 로그인
public int loginSeller(String sellerId, String pwd) throws SQLException {
            String sql = "SELECT pwd FROM seller WHERE sellerId=?";
            jdbcUtil.setSqlAndParameters(sql, new Object[] {sellerId});

            try {
                int result = jdbcUtil.executeUpdate();

                if(result.next()) {
                    if(result.getString(1).equals(pwd))
                        return 1; //사용자 로그인 성공
                    else
                        return 0; //비밀번호 불일치
                }
                return -1; //존재하지 않는 아이디
            } catch (Exception ex) {
                jdbcUtil.rollback();
                ex.printStackTrace();
            }
            finally {
                jdbcUtil.commit();
                jdbcUtil.close();
            }
            return 0;
```

```
}

// Customer 회원 탈퇴
public int removeCustomer(String customerId) throws SQLException {
		String sql = "DELETE FROM customer WHERE customerId=?";
		jdbcUtil.setSqlAndParameters(sql, new Object[] {customerId});

		try {
			int result = jdbcUtil.executeUpdate();
			return result;
		} catch (Exception ex) {
			jdbcUtil.rollback();
			ex.printStackTrace();
		}
		finally {
			jdbcUtil.commit();
			jdbcUtil.close();
		}
		return 0;
}

// Customer 회원 상세 조회
public Customer findCustomer(String customerId) throws SQLException {
		String sql = "SELECT * FROM customer WHERE customerId=?";
		jdbcUtil.setSqlAndParameters(sql, new Object[] {customerId});

		try {
			ResultSet rs = jdbcUtil.executeQuery();
			if (rs.next()) {
				Customer customer = new Customer(
					customerId,
					rs.getString("pwd"),
					rs.getString("name"),
					rs.getString("phone"),
					rs.getString("email"),
					rs.getString("address"));
				return customer;
			}
		} catch (Exception ex) {
```

```
                ex.printStackTrace();
        } finally {
                jdbcUtil.close();
        }
        return null;
}

// Customer 회원 목록 조회 - 테이블 전체를 List로 반환
public List<Customer> findCustomerList() throws SQLException {
        String sql = "SELECT * FROM customer ORDER BY customerId";
        jdbcUtil.setSqlAndParameters(sql, null);

        try {
                ResultSet rs = jdbcUtil.executeQuery();
                List<Customer> customerList = new ArrayList<Customer>();
                while (rs.next()) {
                        Customer customer = new Customer(
                                rs.getString("customerId"),
                                rs.getString("name"),
                                rs.getString("phone"),
                        customerList.add(customer);
                }
                return customerList;

        } catch (Exception ex) {
                ex.printStackTrace();
        } finally {
                jdbcUtil.close();
        }
        return null;
}

// Customer 회원 목록 조회 - 테이블 전체를 List로 반환 (현재 페이지, 페이지당 출력할 사용자 수)
public List<Customer> findCustomerList(int currentPage, int countPerPage) throws SQLException {
        String sql = "SELECT * FROM customer ORDER BY customerId";
        jdbcUtil.setSqlAndParameters(sql, null, ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);

        try {
```

```
                ResultSet rs = jdbcUtil.executeQuery();
                int start = ((currentPage-1) * countPerPage) + 1;

                if ((start >= 0) && rs.absolute(start)) {
                        List<Customer> customerList = new ArrayList<Customer>();

                        do {
                                Customer customer = new Customer(
                                        rs.getString("customerId"),
                                        rs.getString("name"),
                                        rs.getString("phone"),
                                customerList.add(customer);
                        } while ((rs.next()) && (--countPerPage > 0));

                        return customerList;

                }
        } catch (Exception ex) {
                ex.printStackTrace();
        } finally {
                jdbcUtil.close();
        }
        return null;
}

// Customer 아이디 중복체크 함수
public boolean existingCustomer(String customerId) throws SQLException {
        String sql = "SELECT count(*) FROM customer WHERE customerId=?";
        jdbcUtil.setSqlAndParameters(sql, new Object[] {customerId});

        try {
                ResultSet rs = jdbcUtil.executeQuery();
                if (rs.next()) {
                        int count = rs.getInt(1);
                        return (count == 1 ? true : false);
                }
        } catch (Exception ex) {
                ex.printStackTrace();
        } finally {
```

```
                    jdbcUtil.close();
                }
                return false;
        }

}
```

- ProductDAO.java

```
package model.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import model.Product;

public class ProductDAO {
        private JDBCUtil jdbcUtil = null;

        public ProductDAO() {
                jdbcUtil = new JDBCUtil();
        }

        // 상품 추가
        public Product add(Product product) throws SQLException{
                String query = "insert into product(productId, sellerId, price, description, name, category, type) "
                                + "values(Sequence_product.nextVal, ?, ?, ?, ?, ?, ?)";
                Object[] param = new Object[] {product.getSellerId(), product.getPrice(), product.getDescription(),
                                product.getName(), product.getCategory(), product.getType()};
                jdbcUtil.setSqlAndParameters(query, param);
                String key[] = {"productId"};

                try {
                        jdbcUtil.executeUpdate(key);
                        ResultSet rs = jdbcUtil.getGeneratedKeys();
                        if(rs.next()) {
                                int generatedKey = rs.getInt(1);
                                product.setProductId(generatedKey);
                        }
```

```
                return product;
            } catch (Exception e) {
                jdbcUtil.rollback();
                e.printStackTrace();
            } finally {
                jdbcUtil.commit();
                jdbcUtil.close();
            }
            return null; // 추가한 뒤 해당 객체 반환
        }

        // 상품 수정
        public int update(Product product) throws SQLException{
            String query = "update product set sellerId = ?, price = ?, description = ?, name = ? where productId
= ?";
            Object[] param = new Object[] {product.getSellerId(), product.getPrice(), product.getDescription(),
                    product.getName(), product.getProductId()};
            jdbcUtil.setSqlAndParameters(query, param);

            try {
                int result = jdbcUtil.executeUpdate();
                return result;
            }catch(Exception e) {
                jdbcUtil.rollback();
                e.printStackTrace();
            }finally {
                jdbcUtil.commit();
                jdbcUtil.close();
            }
            return 0; // 성공적으로 수행된 개수 반환
        }

        // 상품 삭제
        public int remove(String productId) throws SQLException{
            String query = "delete from product where productId = ?";
            jdbcUtil.setSqlAndParameters(query, new Object[] {productId});

            try {
                int result = jdbcUtil.executeUpdate();
```

```
                return result;
        }catch(Exception e) {
                jdbcUtil.rollback();
                e.printStackTrace();
        }finally {
                jdbcUtil.commit();
                jdbcUtil.close();
        }
        return 0; // 성공적으로 수행된 개수 반환
}

// 이름으로 상품 검색
public List<Product> findProductByName(String name){
        String query = "select * from product where name like '%?%' ?";
        jdbcUtil.setSqlAndParameters(query, new Object[] {name});

        try {
                ResultSet rs = jdbcUtil.executeQuery();
                List<Product> productList = new ArrayList<>();

                while(rs.next()) {
                        Product product = new Product(
                                        rs.getInt("productId"),
                                        rs.getString("name"),
                                        rs.getInt("price"),
                                        rs.getString("description"),
                                        rs.getString("type"),
                                        rs.getString("category"));
                        productList.add(product);
                }
                return productList;
        }catch(Exception e) {
                e.printStackTrace();
        }finally {
                jdbcUtil.close();
        }
        return null;
}
```

```
// 전체 상품 목록 조회(음식/꽃 구분)
public List<Product> findProductList(String type){
        String query = "select * from product where type = ? order by productId"; // 필요한 속성만 검색하기
        jdbcUtil.setSqlAndParameters(query, new Object[] {type});

        try {
                ResultSet rs = jdbcUtil.executeQuery();
                List<Product> productList = new ArrayList<>();

                while(rs.next()) {
                        Product product = new Product(
                                        rs.getInt("productId"),
                                        rs.getString("name"),
                                        rs.getInt("price"),
                                        rs.getString("description"),
                                        type,
                                        rs.getString("category"));
                        productList.add(product);
                }
                return productList;
        }catch(Exception e) {
                e.printStackTrace();
        }finally {
                jdbcUtil.close();
        }
        return null;
}

// 상품 상세 조회
public Product findProduct(int productId) {
        String query = "select * from product where productId = ?";
        jdbcUtil.setSqlAndParameters(query, new Object[] {productId});
        Product product = null;

        try {
                ResultSet rs = jdbcUtil.executeQuery();

                if(rs.next()) {
                        product = new Product(
```

```
                                productId,
                                rs.getString("name"),
                                rs.getInt("price"),
                                rs.getString("description"),
                                rs.getString("type"),
                                rs.getString("category"));
                        }
                }catch (Exception e) {
                        e.printStackTrace();
                }finally {
                        jdbcUtil.close();
                }
                return product;
        }

}
```

- ClassDAO.java

```
package model.dao;

import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import model.ClassInfo; // 이후 도메인 클래스로 변경
import model.Product;

public class ClassDAO {
        private JDBCUtil jdbcUtil = null;

        public ClassDAO() {
                jdbcUtil = new JDBCUtil();
        }

        // 클래스 추가
        public ClassInfo add(ClassInfo c) {
                String query = "insert into class(classId, sellerId, name, classDate, maxNum, currentNum)
values(Sequence_class.nextVal, ?, ?, ?, ?, ?)";
                Object[] param = new Object[] {c.getSellerId(), c.getName(), c.getDate(), c.getMaxNum(),
c.getCurrentNum()};
                jdbcUtil.setSqlAndParameters(query, param);
```

```
		String key[] = {"classId"};


		try {
			jdbcUtil.executeUpdate(key);
			ResultSet rs = jdbcUtil.getGeneratedKeys();
			if(rs.next()) {
				int generatedKey = rs.getInt(1);
				c.setClassId(generatedKey);
			}
			return c;
		} catch (Exception e) {
			jdbcUtil.rollback();
			e.printStackTrace();
		} finally {
			jdbcUtil.commit();
			jdbcUtil.close();
		}
		return null; // 추가한 뒤 해당 객체 반환
	}


	// 클래스 정보 수정
	public int update(ClassInfo c) {
		String query = "update class set sellerId = ?, name = ?, date = ?, maxNum = ?, currentNum = ? where classId = ?";
		Object[] param = new Object[] {c.getSellerId(), c.getName(), c.getDate(), c.getMaxNum(), c.getCurrentNum(), c.getClassId()};
		jdbcUtil.setSqlAndParameters(query, param);


		try {
			int result = jdbcUtil.executeUpdate();
			return result;
		}catch(Exception e) {
			jdbcUtil.rollback();
			e.printStackTrace();
		}finally {
			jdbcUtil.commit();
			jdbcUtil.close();
		}
		return 0; // 성공적으로 수행된 개수 반환
	}
```

```
// 클래스 삭제
public int remove(int classId) {
		String query = "delete from class where classId = ?";
		jdbcUtil.setSqlAndParameters(query, new Object[] {classId});

		try {
			int result = jdbcUtil.executeUpdate();
			return result;
		}catch(Exception e) {
			jdbcUtil.rollback();
			e.printStackTrace();
		}finally {
			jdbcUtil.commit();
			jdbcUtil.close();
		}
		return 0; // 성공적으로 수행된 개수 반환
}

// 전체 상품 목록 조회
public List<ClassInfo> findClassList(){
		String query = "select * from class";
		jdbcUtil.setSqlAndParameters(query, null);

		try {
			ResultSet rs = jdbcUtil.executeQuery();
			List<ClassInfo> classList = new ArrayList<>();

			while(rs.next()) {
				ClassInfo c = new ClassInfo(
						rs.getInt("classId"),
						rs.getString("name"),
						rs.getString("date"),
						rs.getInt("maxNum"),
						rs.getInt("currentNum"),
						rs.getString("sellerId"));
				classList.add(c);
			}
			return classList;
```

```
                }catch(Exception e) {
                        e.printStackTrace();
                }finally {
                        jdbcUtil.close();
                }
                return null;
        }

        // 상품 상세 조회
        public ClassInfo findClass(int classId) {
                String query = "select * from class where classId = ?";
                jdbcUtil.setSqlAndParameters(query, new Object[] {classId});
                ClassInfo c = null;

                try {
                        ResultSet rs = jdbcUtil.executeQuery();

                        if(rs.next()) {
                                c = new ClassInfo(
                                                classId,
                                                rs.getString("name"),
                                                rs.getString("date"),
                                                rs.getInt("maxNum"),
                                                rs.getInt("currentNum"),
                                                rs.getString("sellerId"));
                        }
                }catch (Exception e) {
                        e.printStackTrace();
                }finally {
                        jdbcUtil.close();
                }
                return c;
        }
}
```

- OrderDAO.java

```
package model.dao;

import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import model.Order;

public class OrderDAO {
    private JDBCUtil jdbcUtil = null;

    public OrderDAO() {
        jdbcUtil = new JDBCUtil();
    }

    //주문 목록 조회
    public List<Order> findOrderList() throws SQLException{
        String query = "select * from order where customerId = ? order by customerId";
        jdbcUtil.setSqlAndParameters(query, new Object[] {customerId});

        try {
            ResultSet rs = jdbcUtil.executeQuery();
            List<Order> orderList = new ArrayList<Order>();

            while(rs.next()) {
                Order order = new Order(
                        rs.getString("orderId"),
                        rs.getString("productId"),
                        rs.getInt("quantity"),
                        rs.getInt("price"));
                orderList.add(order);
            }
            return orderList;
        }catch(Exception e) {
            e.printStackTrace();
        }finally {
            jdbcUtil.close();
        }
        return null;
    }
```

```
//주문자 정보 입력
public int ConsumerInfo (ConsumerInfo consumerInfo) throws SQLException{
    int result = 0;
    String query = "insert into consumer (name, phone, email, address) value (?, ?, ?, ?)";

    Object[] param = new Object[]
        { consumerInfo.getName(), consumerInfo.getPhone(), consumerInfo.getEmail(), consumerInfo.getAddress() };
    jdbcUtil.setSqlAndParameters(query, param);

    try {
        result = jdbcUtil.executeUpdate();
        return result;
    } catch(Exception ex) {
        jdbcUtil.rollback();
        ex.printStackTrace();
    } finally {
        jdbcUtil.commit();
        jdbcUtil.close();
    }
    return result;
}

//배송 정보 입력
public int DeliverInfo (DeliverInfo deliverInfo) throws SQLException{
    int result = 0;
    String query = "insert into consumer (name, phone, email, address) value (?, ?, ?, ?)";

    Object[] param = new Object[]
        { deliverInfo.getName(), deliverInfo.getPhone(), deliverInfo.getEmail(), deliverInfo.getAddress() };
    jdbcUtil.setSqlAndParameters(query, param);

    try {
        result = jdbcUtil.executeUpdate();
        return result;
    } catch(Exception ex) {
        jdbcUtil.rollback();
        ex.printStackTrace();
    } finally {
        jdbcUtil.commit();
```

```
            jdbcUtil.close();
        }
        return result;
    }

    //상품 예약 정보 입력
    public int ReservationInfo (ReservationInfo resInfo) throws SQLException{
        int result = 0;
        String query = "insert into consumer (name, phone, date, memo) value (?, ?, ?, ?)";

        Object[] param = new Object[] { resInfo.getName(), resInfo.getPhone(), resInfo.getDate(), resInfo.getMemo() };
        jdbcUtil.setSqlAndParameters(query, param);

        try {
            result = jdbcUtil.executeUpdate();
            return result;
        } catch(Exception ex) {
            jdbcUtil.rollback();
            ex.printStackTrace();
        } finally {
            jdbcUtil.commit();
            jdbcUtil.close();
        }
        return result;
    }

}
```

- CartDAO.java

```
package model.dao;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import model.CartItem;
import util.JDBCUtil;
```

```
public class CartDAO {
	private JDBCUtil jdbcUtil = null;

	public CartDAO() {	// 생성자
		jdbcUtil = new JDBCUtil();
	}

	// 장바구니 항목 조회(장바구니 페이지 첫 화면에 나오는 품목 리스트)
	public List<CartItem> getCartItemList(String customerId) {
		String query = "SELECT cartItemId, quantity, productId FROM CartItem where customerId = ?
				ORDER BY cartItenId";

		Object[] param = new Object[] { customerId };			// customerId = ? 의 매개변수 설정
		jdbcUtil.setSqlAndParameters(query, param);

		try {
			ResultSet rs = jdbcUtil.executeQuery();			// query 문 실행
			List<CartItem> list = new ArrayList<CartItem>();			// list 객체 생성
			while (rs.next()) {
				CartItem dto = new CartItem();	// 하나의 CartItem 객체 생성 후 정보 설정
				dto.setCartItemId(rs.getInt("cartItemId"));
				dto.setQuantity(rs.getInt("quantity"));
				dto.setProductId(rs.getInt("productId"));
				list.add(dto);		// list 객체에 정보를 설정한 CartItem 객체 저장
			}
			return list;		// dto 들의 목록을 반환
		} catch (Exception ex) {
			ex.printStackTrace();
		} finally {
			jdbcUtil.close();
		}
		return null;
	}

	// 장바구니에 등록
	public int addItem(CartItem cartItem) {
		int result = 0;
		String query = "INSERT INTO CartItem (productId, quantity) VALUES(?, ?)";
```

```
        // query 문에 사용할 매개변수 값을 갖는 매개변수 배열 생성
        Object[] param = new Object[] { cartItem.getProductId(), cartItem.getQuantity() };
        jdbcUtil.setSqlAndParameters(query, param);

        try {
                result = jdbcUtil.executeUpdate();                    // insert 문 실행
                System.out.println(cartItem.getCartItemId() + " item이 장바구니에 추가되었습니다.");
        } catch (SQLException ex) {
                System.out.println("입력 오류 발생!!!");
        } catch (Exception ex) {
                jdbcUtil.rollback();
                ex.printStackTrace();
        } finally {
                jdbcUtil.commit();
                jdbcUtil.close();
        }
        return result;
}

// 장바구니에서 삭제 (같은 상품이라도 다른 옵션이면 따로 표시되니까 productId 대신 cartItemId로 변경)
public int removeItem(int cartItemId) {
        String query = "DELETE FROM CartItem WHERE cartItemId = ?";

        jdbcUtil.setSql(query);
        Object[] param = new Object[] { cartItemId };
        jdbcUtil.setParameters(param);

        try {
                int result = jdbcUtil.executeUpdate();                // delete 문 실행
                return result;                                        // delete 에 의해 반영된 레코드 수 반환
        } catch (Exception ex) {
                jdbcUtil.rollback();
                ex.printStackTrace();
        } finally {
                jdbcUtil.commit();
                jdbcUtil.close();
        }
        return 0;
}
```

```
	// 장바구니 항목 옵션 수정
	public int updateItem(CartItem cartItem) {
		String query = "UPDATE CartItem SET quantity = ? WHERE cartItemId = ?";

		Object[] tempParam = new Object[2];   // update 문에 사용할 매개변수를 저장할 수 있는 임시 배열

		tempParam[0] = cartItem.getQuantity();
		tempParam[1] = cartItem.getCartItemId();

		Object[] newParam = new Object[2];
		for (int i=0; i < newParam.length; i++)
			newParam[i] = tempParam[i];

		jdbcUtil.setSqlAndParameters(query, newParam);

		try {
			int result = jdbcUtil.executeUpdate();			// update 문 실행
			return result;					// update 에 의해 반영된 레코드 수 반환
		} catch (Exception ex) {
			jdbcUtil.rollback();
			ex.printStackTrace();
		}
		finally {
			jdbcUtil.commit();
			jdbcUtil.close();
		}
		return 0;
	}

}
```

- PostDAO.java

```
package model.dao;

import java.sql.ResultSet;

import java.sql.SQLException;
import java.util.ArrayList;
```

```
import java.util.List;
import model.Qna;
import model.Review;

// 게시글번호, 날짜 까지 구현
// 페이지 계산 함수 검토 필요
// create의 customerId, productId 가져오기 필요
// 조회수, 댓글 구현 필요

public class PostDAO {
	private JDBCUtil jdbcUtil = null;

	public PostDAO() {
		jdbcUtil = new JDBCUtil();
	}

	// Qna 게시글 작성
	public int createQna(Qna qna) throws SQLException {
		int maxnum = 1;
		String sql = "SELECT MAX(qnaId) FROM qna";

		try {
			int rs = jdbcUtil.executeUpdate();
			if(rs.next())
				maxnum = rs.getInt(1);
			rs.close();

			String sql = "INSERT INTO qna VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
			Object[] param = new Object[] {(maxnum+1), qna.CustomerId, qna.getProductId(),
				qna.getTitle(), qna.getContent(), SYSDATE(), 0, qna.getPwd()};

			jdbcUtil.setSqlAndParameters(sql, param);

			int result = jdbcUtil.executeUpdate();
			return result;
		} catch (Exception ex) {
			jdbcUtil.rollback();
			ex.printStackTrace();
		} finally {
```

```
                jdbcUtil.commit();
                jdbcUtil.close();
        }
        return 0;
}

// Review 게시글 작성
public int createReview(Review review) throws SQLException {
        int maxnum = 1;
        String sql = "SELECT MAX(reviewId) FROM review";

        try {
                int rs = jdbcUtil.executeUpdate();
                if(rs.next())
                        maxnum = rs.getInt(1);
                rs.close();

                String sql = "INSERT INTO review VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
                Object[] param = new Object[] {(maxnum+1), review.CustomerId, review.getProductId(),
                        review.getTitle(), review.getContent(), SYSDATE(), 0, review.getRate()};

                jdbcUtil.setSqlAndParameters(sql, param);

                int result = jdbcUtil.executeUpdate();
                return result;
        } catch (Exception ex) {
                jdbcUtil.rollback();
                ex.printStackTrace();
        } finally {
                jdbcUtil.commit();
                jdbcUtil.close();
        }
        return 0;
}

// Qna 게시글 수정
public int updateQna(Qna qna) throws SQLException {
        String sql = "UPDATE qna "
        + "SET customerId=?, productId=?, title=?, content=?, creationDate=?, viewCount=?, pwd=? "
```

```
            + "WHERE qnaId=?";
            Object[] param = new Object[] {qna.getCustomerId(), qna.getProductId(), qna.getTitle(),
                        qna.getContent(), qna.getCreationDate(), qna.getViewCount(), qna.getPwd(), qna.getQnaId()};

            jdbcUtil.setSqlAndParameters(sql, param);


            try {
                        int result = jdbcUtil.executeUpdate();
                        return result;
            } catch (Exception ex) {
                        jdbcUtil.rollback();
                        ex.printStackTrace();
            }
            finally {
                        jdbcUtil.commit();
                        jdbcUtil.close();
            }
            return 0;
}

// Review 게시글 수정
public int updateReview(Review review) throws SQLException {
            String sql = "UPDATE review "
                        + "SET customerId=?, productId=?, title=?, content=?, creationDate=?, viewCount=?, rate=? "
                        + "WHERE reviewId=?";
            Object[] param = new Object[] {review.getCustomerId(), review.getProductId(), review.getTitle(),
                        review.getContent(), review.getCreationDate(), review.getViewCount(), review.getRate(),
                        review.getReviewId()};
            jdbcUtil.setSqlAndParameters(sql, param);


            try {
                        int result = jdbcUtil.executeUpdate();
                        return result;
            } catch (Exception ex) {
                        jdbcUtil.rollback();
                        ex.printStackTrace();
            }
            finally {
                        jdbcUtil.commit();
                        jdbcUtil.close();
```

```
        }
        return 0;
}

// Qna 게시글 삭제
public int removeQna(String qnaId) throws SQLException {
        String sql = "DELETE FROM qna WHERE qnaId=?";
        jdbcUtil.setSqlAndParameters(sql, new Object[] {qnaId});

        try {
                int result = jdbcUtil.executeUpdate();
                return result;
        } catch (Exception ex) {
                jdbcUtil.rollback();
                ex.printStackTrace();
        }
        finally {
                jdbcUtil.commit();
                jdbcUtil.close();
        }
        return 0;
}

// Review 게시글 삭제
public int removeReview(String reviewId) throws SQLException {
        String sql = "DELETE FROM review WHERE reviewId=?";
        jdbcUtil.setSqlAndParameters(sql, new Object[] {reviewId});

        try {
                int result = jdbcUtil.executeUpdate();
                return result;
        } catch (Exception ex) {
                jdbcUtil.rollback();
                ex.printStackTrace();
        }
        finally {
                jdbcUtil.commit();
                jdbcUtil.close();
        }
```

```
        return 0;
}

// Qna 게시글 상세 조회
public Qna findQna(String qnaId) throws SQLException {
        String sql = "SELECT * FROM qna WHERE qnaId=?";
        jdbcUtil.setSqlAndParameters(sql, new Object[] {qnaId});

        try {
                ResultSet rs = jdbcUtil.executeQuery();
                if (rs.next()) {
                        Qna qna = new Qna(
                                rs.getInt("productId"),
                                rs.getString("title"),
                                rs.getString("customerId"),
                                rs.getDate("creationDate")),
                                rs.getInt("viewCount")),
                                rs.getString("content"),

                        return qna;
                }
        } catch (Exception ex) {
                ex.printStackTrace();
        } finally {
                jdbcUtil.close();
        }
        return null;
}

// Review 게시글 상세 조회
public Review findReview(String reviewId) throws SQLException {
        String sql = "SELECT * FROM review WHERE reviewId=?";
        jdbcUtil.setSqlAndParameters(sql, new Object[] {reviewId});

        try {
                ResultSet rs = jdbcUtil.executeQuery();
                if (rs.next()) {
                        Review review = new Review(
                                rs.getInt("productId"),
```

```
                                        rs.getInt("rate"),
                                        rs.getString("title"),
                                        rs.getString("customerId"),
                                        rs.getDate("creationDate")),
                                        rs.getInt("viewCount")),
                                        rs.getString("content"),

                                return review;
                        }
                } catch (Exception ex) {
                        ex.printStackTrace();
                } finally {
                        jdbcUtil.close();
                }
                return null;
        }

        // Qna 게시물 목록 조회 - 테이블 전체를 List로 반환
        public List<Qna> findQnaList() throws SQLException {
                String sql = "SELECT * FROM qna ORDER BY qnaId";
                jdbcUtil.setSqlAndParameters(sql, null);

                try {
                        ResultSet rs = jdbcUtil.executeQuery();
                        List<Qna> qnaList = new ArrayList<Qna>();
                        while (rs.next()) {
                                Qna qna = new Qna(
                                        qnaId,
                                        rs.getString("title"),
                                        rs.getString("customerId"),
                                        rs.getDate("creationDate")),
                                        rs.getInt("viewCount")),
                                qnaList.add(qna);
                        }
                        return qnaList;

                } catch (Exception ex) {
                        ex.printStackTrace();
                } finally {
```

```
                jdbcUtil.close();
        }
        return null;
}

// Review 게시물 목록 조회 - 테이블 전체를 List로 반환
public List<Review> findReviewList() throws SQLException {
        String sql = "SELECT * FROM review ORDER BY reviewId";
        jdbcUtil.setSqlAndParameters(sql, null);

        try {
                ResultSet rs = jdbcUtil.executeQuery();
                List<Review> reviewList = new ArrayList<Review>();
                while (rs.next()) {
                        Review review = new Review(
                                reviewId,
                                rs.getString("title"),
                                rs.getString("customerId"),
                                rs.getDate("creationDate")),
                                rs.getInt("viewCount")),
                        reviewList.add(review);
                }
                return reviewList;

        } catch (Exception ex) {
                ex.printStackTrace();
        } finally {
                jdbcUtil.close();
        }
        return null;
}

// Qna 게시물 목록 조회 - 테이블 전체를 List로 반환 (현재 페이지, 페이지당 출력할 사용자 수)
public List<Qna> findQnaList(int currentPage, int countPerPage) throws SQLException {
        String sql = "SELECT * FROM qna ORDER BY qnaId";
        jdbcUtil.setSqlAndParameters(sql, null, ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);

        try {
```

```
                ResultSet rs = jdbcUtil.executeQuery();
                int start = ((currentPage-1) * countPerPage) + 1;

                if ((start >= 0) && rs.absolute(start)) {
                        List<Qna> qnaList = new ArrayList<Qna>();

                        do {
                                Qna qna = new Qna(
                                        qnaId,
                                        rs.getString("title"),
                                        rs.getString("customerId"),

                                        rs.getDate("creationDate")),
                                        rs.getInt("viewCount")),
                                qnaList.add(qna);
                        } while ((rs.next()) && (--countPerPage > 0));

                        return qnaList;

                }
        } catch (Exception ex) {
                ex.printStackTrace();
        } finally {
                jdbcUtil.close();
        }
        return null;
}

// Review 게시물 목록 조회 - 테이블 전체를 List로 반환 (현재 페이지, 페이지당 출력할 사용자 수)
public List<Review> findReviewList(int currentPage, int countPerPage) throws SQLException {
        String sql = "SELECT * FROM review ORDER BY reviewId";
        jdbcUtil.setSqlAndParameters(sql, null, ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);

        try {
                ResultSet rs = jdbcUtil.executeQuery();
                int start = ((currentPage-1) * countPerPage) + 1;

                if ((start >= 0) && rs.absolute(start)) {
                        List<Review> reviewList = new ArrayList<Review>();
```

```
                    do {
                        Review review = new Review(
                            reviewId,
                            rs.getString("title"),
                            rs.getString("customerId"),

                            rs.getDate("creationDate")),
                            rs.getInt("viewCount")),
                        reviewList.add(review);
                    } while ((rs.next()) && (--countPerPage > 0));


                    return reviewList;

                }
            } catch (Exception ex) {
                ex.printStackTrace();
            } finally {
                jdbcUtil.close();
            }
            return null;
        }
}
```