Exercise 1: Basic

Trong bài này sẽ sử dụng file jdbcUltis để thao tác với cơ sở dữ liệu, `jdbcUltis sẽ được khai báo như 1 biến private trong Class sử dụng và được khởi tạo trong hàm tạo của class`

Question 1: (Sử dụng Database Testing System đã xây dựng ở SQL)
Tạo connection tới database Testing System
In ra "Connect success!" khi kết nối thành công.

```
public void question1() throws ClassNotFoundException, SQLException {
		System.out.println("Test kết nối.");
		jdbc.connnectionTestting();
	}
```

Question 2:
Tạo method để in ra các thông tin của position gồm có id, name

```
	public void question2() throws ClassNotFoundException, SQLException {
		String sql = "SELECT * FROM position;";
		ResultSet posResult = jdbc.executeQuery(sql);
		System.out.println("Thông tin Position đang có trên hệ thống: ");
		String leftAlignFormat = "| %-6d | %-21s |%n";

		System.out.format("+--------+----------------------+%n");
		System.out.format("|   ID   | PositionName         |%n");
		System.out.format("+--------+----------------------+%n");
		while (posResult.next()) {
			System.out.format(leftAlignFormat, posResult.getInt(1), posResult.getString(2));
		}
		System.out.format("+--------+----------------------+%n");
	}
```

```
Thông tin Position đang có trên hệ thống:
+--------+----------------------+
|   ID   | PositionName         |
+--------+----------------------+
| 1      | Dev                  |
| 2      | Test                 |
| 3      | Scrum Master         |
| 4      | PM                   |
+--------+----------------------+
```

Question 3:
Tạo method để tạo position, user sẽ nhập vào name.

```
public void question3() throws ClassNotFoundException, SQLException {
		System.out.println("Tạo Position mới.");
		String sql = "INSERT INTO position (PositionName) VALUES (?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		System.out.println("Chọn Positon cần tạo 1.Dev, 2.Test, 3.Scrum Master, 4.PM: ");
		String name = getName();
		preStatement.setString(1, name);
		if (preStatement.executeUpdate()==1) {
			System.out.println("Tạo thành công");
```

```
                        question2();
                }else {
                        System.out.println("Đã có lỗi xảy ra");
                }
        }
```

Phương thức getName để lấy ra tên từ số chọn của Position, trong phần này cần phải tạo phương thức getName riêng để thoát khỏi vòng lặp khi lựa chọn.

```
private String getName() {
                while (true) {
                        switch (ScannerUltis.inputIntPositive()) {
                        case 1:
                                return "Dev";

                        case 2:
                                return "Test";

                        case 3:
                                return "Scrum Master";

                        case 4:
                                return "PM";

                        default:
                                System.out.println("Nhập lại");
                                break;
                        }
                }
        }
```

Question 4:

Tạo method để update tên của position gồm có id = 5 thành "Dev".

```
public void question4() throws ClassNotFoundException, SQLException {
                System.out.println("Update tên của Position: ");
                System.out.println("Danh sách Position: ");
                question2();
                System.out.println("Chọn ID cần update");
                int id = ScannerUltis.inputIntPositive();
                System.out.println("Chọn tên Positon cần update: 1.Dev, 2.Test,
3.Scrum Master, 4.PM: ");
                String newName = getName();
                String sql = "UPDATE position SET PositionName = ? WHERE
(PositionID = ?);";
                PreparedStatement pre4 = jdbc.createPrepareStatement(sql);
                pre4.setString(1, newName);
                pre4.setInt(2, id);
                if (pre4.executeUpdate() == 1) {
                        System.out.println("Update thành công");
                        question2();
                } else {
                        System.out.println("Có lỗi xảy ra");
                }

        }
```

Question 5:
Tạo method để delete của position theo id và user sẽ nhập vào id

```
public void question5() throws ClassNotFoundException, SQLException {
		System.out.println("Xóa postion theo ID");
		question2();
		System.out.println("Chọn ID cần xóa:");
		int id = ScannerUltis.inputIntPositive();
		String sql = "DELETE FROM position WHERE (PositionID = ?);";
		PreparedStatement pre5 = jdbc.createPrepareStatement(sql);
		pre5.setInt(1, id);
		if (pre5.executeUpdate() == 1) {
			System.out.println("Xóa thành công");
			question2();
		} else {
			System.out.println("Xóa không thành công");
			question2();
		}
	}
```

Exercise 2: CRUD

Tạo class DepartmentDao trong package backend để cung cấp các method chuyên thao tác với table Department trong database. (các method cụ thể ở question bên dưới)

Class DepartmentDao

```
package com.vti.DAO;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.vti.entity.Department;
import com.vti.ultis.jdbcUltis;

public class DepartmentDao {
	private jdbcUltis jdbc;

	public DepartmentDao() throws FileNotFoundException, IOException {
		jdbc = new jdbcUltis();
	}

	public List<Department> getListDepartment() throws
ClassNotFoundException, SQLException {
		String sql = "SELECT * FROM Department";
		ResultSet resultSet = jdbc.executeQuery(sql);

		List<Department> listDep = new ArrayList<Department>();
		while (resultSet.next()) {
```

```
                    Department dep = new
Department(resultSet.getInt("DepartmentID"),
resultSet.getString("DepartmentName"));
                    listDep.add(dep);
             }
             jdbc.disConnection();
             return listDep;
      }

      public Department getDepByID(int id) throws SQLException,
ClassNotFoundException {
             String sql = "SELECT * FROM Department WHERE DepartmentID = ?";
             PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
             preStatement.setInt(1, id);
             ResultSet result = preStatement.executeQuery();
             if (result.next()) {
                    Department dep = new
Department(result.getInt("DepartmentID"), result.getNString("DepartmentName"));
                    return dep;
             } else {
                    System.out.println("Khong tim duoc");
                    jdbc.disConnection();
                    return null;
             }

      }

      public Boolean isDepartmentNameExists(String name) throws SQLException,
ClassNotFoundException {
             String sql = "SELECT * FROM Department WHERE DepartmentName = ?";
             PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
             preStatement.setNString(1, name);
             ResultSet result = preStatement.executeQuery();

             if (result.next()) {
                    jdbc.disConnection();
                    return true;
             } else {
                    jdbc.disConnection();
                    return false;
             }
      }

      public boolean createDep(String name) throws SQLException,
ClassNotFoundException {
             String sql = "INSERT INTO Department (DepartmentName) VALUES
(?);";
             PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
             preStatement.setNString(1, name);
             int result = preStatement.executeUpdate();
             if (result == 1) {
                    System.out.println("Tạo phòng thành công");
                    jdbc.disConnection();
                    return true;
             } else {
                    System.out.println("Đã có lỗi xảy ra");
                    jdbc.disConnection();
                    return false;
             }
```

```
	}

	public boolean updateDepartmentName(int id, String newName) throws
ClassNotFoundException, SQLException {
		Department depID = getDepByID(id);
		if (depID == null) {
			return false;
		} else {

			String sql = "UPDATE Department SET DepartmentName = ? WHERE
(DepartmentID = ?);";
			PreparedStatement preStatement =
jdbc.createPrepareStatement(sql);
			preStatement.setNString(1, newName);
			preStatement.setInt(2, id);
			int result = preStatement.executeUpdate();
			if (result == 1) {
				jdbc.disConnection();
				return true;
			} else {
				jdbc.disConnection();
				return false;
			}

		}

	}
}
```


Trong bài này sẽ sử dụng file jdbcUltis để thao tác với cơ sở dữ liệu.

Question 1: read data – get list departments

Tạo method để lấy ra danh sách tất cả các Department
Gợi ý:
Viết method **getDepartments()** và return ra
List<Department>, Nếu có lỗi sẽ throw Exception lên frontend để in ra
Trên front-end sẽ gọi class DepartmentDao và demo method
này

```
	public void question1() throws ClassNotFoundException, SQLException {
		List<Department> listDep1 = depDAQ.getListDepartment();
		String leftAlignFormat = "| %-6d | %-21s |%n";

		System.out.format("+--------+-----------------------+%n");
		System.out.format("|   ID   | Department Name       |%n");
		System.out.format("+--------+-----------------------+%n");

		for (Department department : listDep1) {
			System.out.format(leftAlignFormat, department.getId(),
department.getName());
		}
		System.out.format("+--------+-----------------------+%n");
	}
```

Question 2: read data – get department by id
Tạo method để lấy ra Department có id = 5
Gợi ý: Làm tương tự câu trên
Nếu không có department nào có id = 5 thì sẽ in ra là
"Cannot find department which has id = 5"
Nếu có lỗi sẽ throw Exception lên front-end để in ra
Trên front-end sẽ gọi class DepartmentDao và demo method
này

```
      public void question2() throws ClassNotFoundException, SQLException {
            System.out.println("Tìm thông tin phòng có ID =5");
            Department depQues2 = depDAQ.getDepByID(5);
            if (depQues2 != null) {
                  String leftAlignFormat = "| %-6d | %-21s |%n";
                  System.out.format("+--------+-----------------------+%n");
                  System.out.format("|   ID   | Department Name       |%n");
                  System.out.format("+--------+-----------------------+%n");
                  System.out.format(leftAlignFormat, depQues2.getId(),
depQues2.getName());
                  System.out.format("+--------+-----------------------+%n");
            } else {
                  System.out.println("Không tồn tại phòng này trên HT");
            }

                                                                  }
```

Question 3: Tiếp tục Question 2 (read data – get department by id)
Không fix cứng id nữa mà sẽ dùng scanner để yêu cầu người dùng
nhập vào id, sau đó trả về thông tin department có id như người dùng
nhập vào
Gợi ý:
Trên backend sẽ viết method **getDepartmentById(int id)** và
return ra **Department**
Nếu tìm thấy department có id = parameter thì sẽ return về department đó
Nếu không tìm thấy thì sẽ throw ra với message "Cannot find department which has id = " +
id
Nếu có lỗi sẽ throw Exception lên front-end để in ra
Trên front-end sẽ gọi class DepartmentDao và demo method
này

```
public void question3() throws ClassNotFoundException, SQLException {
            System.out.println("Tìm kiếm phòng theo ID: ");
            System.out.println("Nhập vào ID cần tìm kiếm: ");
            int idFind = ScannerUltis.inputIntPositive();
            Department depQues3 = depDAQ.getDepByID(idFind);
            if (depQues3 != null) {
                  String leftAlignFormat = "| %-6d | %-21s |%n";
                  System.out.format("+--------+-----------------------+%n");
                  System.out.format("|   ID   | Department Name       |%n");
                  System.out.format("+--------+-----------------------+%n");
                  System.out.format(leftAlignFormat, depQues3.getId(),
depQues3.getName());
```

```
                System.out.format("+--------+----------------------+%n");
            } else {
                System.out.println("Không tồn tại phòng này trên HT");
            }

                                                                    }
```

Question 4: check data exists – check department name exists
Tạo method để check department name có tồn tại hay không?
Gợi ý:
Trên backend sẽ viết method
isDepartmentNameExists(String name) và return ra **boolean**
Nếu tìm thấy department đã có name = name parameter
thì sẽ return true
Nếu không tìm thấy thì sẽ return ra false
Nếu có lỗi sẽ throw Exception lên front-end để in ra
Trên front-end sẽ gọi class DepartmentDao và demo method
này

```
    public void question4() throws ClassNotFoundException, SQLException {
            System.out.println("Kiểm tra tên phòng đã có trên hệ thông? ");
            System.out.println("Nhập vào tên cần kiểm tra: ");
            String nameCheck = ScannerUltis.inputString();
            Boolean checkResult = depDAQ.isDepartmentNameExists(nameCheck);
            if (checkResult) {
                System.out.println("Tên đã có trên hệ thống.");
                question1();
            } else {
                System.out.println("Tên chưa có trên hệ thống.");
            }
                                                                    }
```

Question 5: create data – create department
Tạo method để người dùng có thể tạo được department
Gợi ý:
Trên backend sẽ viết method **void createDepartment (String name)**
Check xem đã có department nào có tên như parameter
chưa (sử dụng method ở Question 6 để check)
Nếu tìm đã có department có tên như parameter thì throw ra Exception "Department Name is Exists!"
Nếu không tìm thấy thì sẽ create department
Nếu có lỗi sẽ throw Exception lên front-end để in ra
Trên front-end sẽ gọi class DepartmentDao và demo method này (dùng scanner để nhập thông tin của department muốn create)

```
public void question5() throws ClassNotFoundException, SQLException {
            String newNameDep = getNewName();
            if (depDAQ.createDep(newNameDep)) {
                System.out.println("Tạo thành công.");
                question1();
            } else {
                System.out.println("Đã có lỗi xảy ra");
```

```
            }

                                                        }
```

➔Hàm getNewName()

```
private String getNewName() throws ClassNotFoundException, SQLException {
            while (true) {
                  System.out.println("Nhập vào tên phòng cần tạo: ");
                  String newName = ScannerUltis.inputString();
                  if (depDAQ.isDepartmentNameExists(newName)) {
                        System.out.println("Đã có phòng trên hệ thống");
                  } else {
                        return newName;
                  }

            }

                                                        }
```

Question 6: update data – update department
Tạo method để người dùng có thể update được department name
Gợi ý:
Trên backend sẽ viết method **void updateDepartmentName (int id, String newName)**
Check xem đã có department nào có id như id parameter chưa (Viết thêm method như Question 6 để check, tên method là **isDepartmentIdExists(int id)**)
Nếu không tìm thấy department có id = id parameter thì sẽ throw ra Exception có message "Cannot find department which has id = " + id
Nếu tìm thấy department có id = id parameter thì sẽ check xem tên mới của department có bị trùng không (sử dụng method ở Question 6 để check)
Nếu tìm đã có department có tên như parameter thì throw ra Exception "Department Name is Exists!"
Nếu không tìm thấy thì sẽ update department
Nếu có lỗi sẽ throw Exception lên front-end để in ra
Trên front-end sẽ gọi class DepartmentDao và demo method này(dùng scanner để nhập thông tin của department muốn update)

```
public void question6() throws ClassNotFoundException, SQLException {
            question1();
            int updateID = getIdUpdate();
            System.out.println("Nhập vào tên cần Updare: ");
            String newName = ScannerUltis.inputString();
            if (depDAQ.updateDepartmentName(updateID, newName)) {
                  System.out.println("Update tên phòng thành công: ");
                  question1();
            } else {
                  System.out.println("Đã có lỗi xảy ra");
            }

                                                        }
```

➔Hàm getIdUpdate()

```
private int getIdUpdate() throws ClassNotFoundException, SQLException {
            while (true) {
                  System.out.println("Nhập ID phòng cần Update: ");
```

```
                int id = ScannerUltis.inputIntPositive();
                Department dep = depDAQ.getDepByID(id);
                if (dep == null) {
                        System.out.println("Không có ID này trên HT");
                } else {
                        return id;
                }
        }
                                                                }
```

Question 7:
Tạo method để người dùng có thể xóa được department theo id mà user nhập vào
Gợi ý:
Trên backend sẽ viết method **void deleteDepartment (int id)**
Check xem đã có department nào có id như id parameter chưa (sử dụng method ở Question 6 để check)
Nếu không tìm thấy department có id = id parameter thì sẽ throw ra Exception có message "Cannot find department which has id = " + id
Nếu tìm thấy thì sẽ delete department đó
Nếu có lỗi sẽ throw Exception lên front-end để in ra
Trên front-end sẽ gọi class DepartmentDao và demo method này (dùng scanner để nhập vào id của department muốn delete)

```
public void question7() throws ClassNotFoundException, SQLException {
        question1();
        int updateID = getIdUpdate();
        if (depDAQ.delDepByID(updateID)) {
                System.out.println("Xóa phòng thành công");
                question1();
        } else {
                System.out.println("Đã có lỗi xảy ra");
        }
                                                                }
```

Viết hàm `getIdUpdate`

```
private int getIdUpdate() throws ClassNotFoundException, SQLException {
        while (true) {
                System.out.println("Nhập ID phòng cần xóa: ");
                int id = ScannerUltis.inputIntPositive();
                Department dep = depDAQ.getDepByID(id);
                if (dep == null) {
                        System.out.println("Không có ID này trên HT");
                } else {
                        return id;
                }
        }
                                                                }
```

Question 8:
Làm các chức năng CRUD tương tự với table Account
(Chú ý: trong chức năng getListAccount, getById thì phải in ra cả thông tin tên của department

Tạo class PositionDao:

```
package com.vti.DAO;
```

```java
import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.vti.entity.Department;
import com.vti.entity.Position;
import com.vti.ultis.jdbcUltis;

public class PositionDao {
	private jdbcUltis jdbc;

	public PositionDao() throws FileNotFoundException, IOException {
		jdbc = new jdbcUltis();
	}

	public List<Position> getListPosition() throws ClassNotFoundException,
SQLException {
		String sql = "SELECT * FROM position";
		ResultSet resultSet = jdbc.executeQuery(sql);

		List<Position> listPosition = new ArrayList<Position>();
		while (resultSet.next()) {
			Position pos = new Position(resultSet.getInt(1),
resultSet.getString(2));
			listPosition.add(pos);
		}
		jdbc.disConnection();
		return listPosition;
	}

	public Position getPosByID(int id) throws SQLException,
ClassNotFoundException {
		String sql = "SELECT * FROM position WHERE PositionID = ?";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setInt(1, id);
		ResultSet result = preStatement.executeQuery();
		if (result.next()) {
			Position pos = new Position(result.getInt(1),
result.getString(2));
			return pos;
		} else {
			jdbc.disConnection();
			return null;
		}

	}
}
```

Tạo class DepartmentDao:

```java
package com.vti.DAO;

import java.io.FileNotFoundException;
import java.io.IOException;
```

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

import com.vti.entity.Department;
import com.vti.ultis.jdbcUltis;

public class DepartmentDao {
	private jdbcUltis jdbc;

	public DepartmentDao() throws FileNotFoundException, IOException {
		jdbc = new jdbcUltis();
	}

	public List<Department> getListDepartment() throws
ClassNotFoundException, SQLException {
		String sql = "SELECT * FROM Department ORDER BY DepartmentID";
		ResultSet resultSet = jdbc.executeQuery(sql);

		List<Department> listDep = new ArrayList<Department>();
		while (resultSet.next()) {
			Department dep = new
Department(resultSet.getInt("DepartmentID"),
resultSet.getString("DepartmentName"));
			listDep.add(dep);
		}
		jdbc.disConnection();
		return listDep;
	}

	public Department getDepByID(int id) throws SQLException,
ClassNotFoundException {
		String sql = "SELECT * FROM Department WHERE DepartmentID = ?";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setInt(1, id);
		ResultSet result = preStatement.executeQuery();
		if (result.next()) {
			Department dep = new
Department(result.getInt("DepartmentID"), result.getNString("DepartmentName"));
			return dep;
		} else {
			jdbc.disConnection();
			return null;
		}

	}

	public Boolean isDepartmentNameExists(String name) throws SQLException,
ClassNotFoundException {
		String sql = "SELECT * FROM Department WHERE DepartmentName = ?";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setNString(1, name);
		ResultSet result = preStatement.executeQuery();

		if (result.next()) {
			jdbc.disConnection();
			return true;
```

```
		} else {
			jdbc.disConnection();
			return false;
		}
	}

	public boolean createDep(String name) throws SQLException,
ClassNotFoundException {
		String sql = "INSERT INTO Department (DepartmentName) VALUES
(?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setNString(1, name);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}

	}

	public boolean updateDepartmentName(int id, String newName) throws
ClassNotFoundException, SQLException {
		Department depID = getDepByID(id);
		if (depID == null) {
			return false;
		} else {

			String sql = "UPDATE Department SET DepartmentName = ? WHERE
(DepartmentID = ?);";
			PreparedStatement preStatement =
jdbc.createPrepareStatement(sql);
			preStatement.setNString(1, newName);
			preStatement.setInt(2, id);
			int result = preStatement.executeUpdate();
			if (result == 1) {
				jdbc.disConnection();
				return true;
			} else {
				jdbc.disConnection();
				return false;
			}

		}

	}

	public boolean delDepByID(int id) throws ClassNotFoundException,
SQLException {
		Department depID = getDepByID(id);
		if (depID == null) {
			return false;
		} else {
			String sql = "DELETE FROM department WHERE (DepartmentID =
?);";
			PreparedStatement preStatement =
jdbc.createPrepareStatement(sql);
```

```
                        preStatement.setInt(1, id);
                        int result = preStatement.executeUpdate();
                        if (result == 1) {
                                jdbc.disConnection();
                                return true;
                        } else {
                                jdbc.disConnection();
                                return false;
                        }

                }

        }

}
```

Tạo class AccountDao:

```
package com.vti.DAO;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

import com.vti.entity.Account;
import com.vti.entity.Department;
import com.vti.entity.Position;
import com.vti.ultis.jdbcUltis;

public class AccountDao {
        private jdbcUltis jdbc;

        public AccountDao() throws FileNotFoundException, IOException {
                jdbc = new jdbcUltis();
        }

        public List<Account> getListAccount()
                        throws ClassNotFoundException, SQLException,
FileNotFoundException, IOException {
                String sql = "SELECT * FROM account ORDER BY AccountID";
                ResultSet resultSet = jdbc.executeQuery(sql);
                List<Account> listAcc = new ArrayList<Account>();
                while (resultSet.next()) {
                        Account acc = new Account();
                        acc.setID(resultSet.getInt(1));
                        acc.setEmail(resultSet.getString(2));
                        acc.setUsername(resultSet.getString(3));
                        acc.setFullName(resultSet.getString(4));

                        DepartmentDao depDao = new DepartmentDao();
                        Department dep = depDao.getDepByID(resultSet.getInt(5));
                        acc.setDepartment(dep);
```

```
                    PositionDao posDao = new PositionDao();
                    Position pos = posDao.getPosByID(resultSet.getInt(6));
                    acc.setPosition(pos);

                    LocalDate lcd = resultSet.getDate(7).toLocalDate();
                    acc.setCreateDate(lcd);

                    listAcc.add(acc);
             }
             return listAcc;
      }

      public Account getAccByID(int id) throws SQLException,
ClassNotFoundException, FileNotFoundException, IOException {
             String sql = "SELECT * FROM account WHERE AccountID = ?";
             PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
             preStatement.setInt(1, id);
             ResultSet resultSet = preStatement.executeQuery();
             if (resultSet.next()) {
                    Account acc = new Account();
                    acc.setID(resultSet.getInt(1));
                    acc.setEmail(resultSet.getString(2));
                    acc.setUsername(resultSet.getString(3));
                    acc.setFullName(resultSet.getString(4));
                    DepartmentDao depDao = new DepartmentDao();
                    Department dep = depDao.getDepByID(resultSet.getInt(5));
                    acc.setDepartment(dep);
                    PositionDao posDao = new PositionDao();
                    acc.setPosition(posDao.getPosByID(resultSet.getInt(6)));
                    LocalDate lcd =
Date.valueOf(resultSet.getDate(7).toString()).toLocalDate();
                    acc.setCreateDate(lcd);
                    return acc;
             } else {
                    jdbc.disConnection();
                    return null;
             }

      }

      public Boolean isAccNameExists(String name) throws SQLException,
ClassNotFoundException {
             String sql = "SELECT * FROM account WHERE Username = ?";
             PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
             preStatement.setNString(1, name);
             ResultSet result = preStatement.executeQuery();

             if (result.next()) {
                    jdbc.disConnection();
                    return true;
             } else {
                    jdbc.disConnection();
                    return false;
             }
      }

      public boolean createAccount(Account acc, int depId, int posId) throws
SQLException, ClassNotFoundException {
```

```
            String sql = "INSERT INTO account (Email, Username, FullName,
DepartmentID, PositionID, CreateDate) VALUES (?, ?, ?,?,?,now());";

            PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
            preStatement.setNString(1, acc.getEmail());
            preStatement.setNString(2, acc.getUsername());
            preStatement.setNString(3, acc.getFullName());
            preStatement.setInt(4, depId);
            preStatement.setInt(5, posId);
//          preStatement.setDate(6,
java.util.Date.parse(LocalDate.now().toString()) );
            int result = preStatement.executeUpdate();
            if (result == 1) {
                  jdbc.disConnection();
                  return true;
            } else {
                  jdbc.disConnection();
                  return false;
            }

      }

      public boolean delAccByID(int ID) throws ClassNotFoundException,
SQLException {
            String sql = "DELETE FROM account WHERE (AccountID = ?);";
            PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
            preStatement.setInt(1, ID);
            int result = preStatement.executeUpdate();
            if (result == 1) {
                  jdbc.disConnection();
                  return true;
            } else {
                  jdbc.disConnection();
                  return false;
            }
      }

      public boolean updateByEmai(int id, String newEmail) throws
ClassNotFoundException, SQLException {
            String sql = "UPDATE account SET Email = ? WHERE (AccountID =
?);";
            PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
            preStatement.setString(1, newEmail);
            preStatement.setInt(2, id);
            int result = preStatement.executeUpdate();
            if (result == 1) {
                  jdbc.disConnection();
                  return true;
            } else {
                  jdbc.disConnection();
                  return false;
            }

      }
      public boolean updateByUserName(int id, String newUserName) throws
ClassNotFoundException, SQLException {
            String sql = "UPDATE account SET Username = ? WHERE (AccountID =
?);";
            PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
```

```
			preStatement.setString(1, newUserName);
			preStatement.setInt(2, id);
			int result = preStatement.executeUpdate();
			if (result == 1) {
				jdbc.disConnection();
				return true;
			} else {
				jdbc.disConnection();
				return false;
			}

		}

		public boolean updateByFullName(int id, String newFullName) throws
ClassNotFoundException, SQLException {
			String sql = "UPDATE account SET FullName = ? WHERE (AccountID =
?);";
			PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
			preStatement.setString(1, newFullName);
			preStatement.setInt(2, id);
			int result = preStatement.executeUpdate();
			if (result == 1) {
				jdbc.disConnection();
				return true;
			} else {
				jdbc.disConnection();
				return false;
			}

		}

		public boolean updateByDepId(int id, int idDep) throws
ClassNotFoundException, SQLException {
			String sql = "UPDATE account SET DepartmentID = ? WHERE (AccountID
= ?);";
			PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
			preStatement.setInt(1, idDep);
			preStatement.setInt(2, id);
			int result = preStatement.executeUpdate();
			if (result == 1) {
				jdbc.disConnection();
				return true;
			} else {
				jdbc.disConnection();
				return false;
			}

		}

		public boolean updateByPosId(int id, int idPos) throws
ClassNotFoundException, SQLException {
			String sql = "UPDATE account SET PositionID = ? WHERE (AccountID =
?);";
			PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
			preStatement.setInt(1, idPos);
			preStatement.setInt(2, id);
			int result = preStatement.executeUpdate();
			if (result == 1) {
				jdbc.disConnection();
```

```
                return true;
            } else {
                jdbc.disConnection();
                return false;
            }

        }
}
```

1. **Question1:**

```
public void question1() throws ClassNotFoundException,
FileNotFoundException, SQLException, IOException {
            List<Account> listAcc1 = accDAO.getListAccount();
            String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s | %-
14s | %-16s | %-16s | %n";
            System.out.format(
                        "+----+-----------------------+-----------------+-
----------------------+----------------+-----------------+-----------------
-+%n");
            System.out.format(
                        "|ID  | Email                 | Username        |
FullName            | Department     | Position        | Create Date
|%n");
            System.out.format(
                        "+----+-----------------------+-----------------+-
----------------------+----------------+-----------------+-----------------
-+%n");

            for (Account acc : listAcc1) {
                System.out.format(leftAlignFormat, acc.getId(),
acc.getEmail(), acc.getUsername(), acc.getFullName(),
                                    acc.getDepartment(), acc.getPosition(),
acc.getCreateDate());
            }
            System.out.format(
                        "+----+-----------------------+-----------------+-
----------------------+----------------+-----------------+-----------------
-+%n");

                                                                    }
```

2. **Question2:**

```
public void question2() throws ClassNotFoundException,
FileNotFoundException, SQLException, IOException {
            System.out.println("Tìm thông tin Account có ID =5");
            Account acc2 = accDAO.getAccByID(5);
            if (acc2 != null) {
                String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s
| %-14s | %-16s | %-16s | %n";
                System.out.format(
                                    "+----+-----------------------+----------
----+-----------------------+----------------+-----------------+----------
-------+%n");
```

```
                    System.out.format(
                                    "|ID  | Email                | Username
|   FullName           | Department     | Position         | Create Date
|%n");
                    System.out.format(
                                    "+----+----------------------+------------
----+----------------------+---------------+------------------+-----------
-------+%n");

                    System.out.format(leftAlignFormat, acc2.getId(),
acc2.getEmail(), acc2.getUsername(), acc2.getFullName(),
                                    acc2.getDepartment(), acc2.getPosition(),
acc2.getCreateDate());

                    System.out.format(
                                    "+----+----------------------+------------
----+----------------------+---------------+------------------+-----------
-------+%n");

            } else {
                    System.out.println("Không tồn tại phòng này trên HT");
            }

                                                                    }
```

3. Question3:

```
public void question3() throws ClassNotFoundException, FileNotFoundException,
SQLException, IOException {
            System.out.println("Tìm kiếm Account theo ID: ");
            System.out.println("Nhập vào ID cần tìm kiếm: ");
            int idFind = ScannerUltis.inputIntPositive();
            Account acc2 = accDAO.getAccByID(idFind);
            if (acc2 != null) {
                    String leftAlignFormat = "| %-2d | %-21s | %-15s | %-21s |
%-14s | %-16s | %-16s | %n";
                    System.out.format(
                                    "+----+----------------------+------------
-+----------------------+---------------+------------------+-----------------
+%n");
                    System.out.format(
                                    "|ID  | Email                | Username
|   FullName           | Department     | Position         | Create Date
|%n");
                    System.out.format(
                                    "+----+----------------------+------------
-+----------------------+---------------+------------------+-----------------
+%n");

                    System.out.format(leftAlignFormat, acc2.getId(),
acc2.getEmail(), acc2.getUsername(), acc2.getFullName(),
                                    acc2.getDepartment(), acc2.getPosition(),
acc2.getCreateDate());

                    System.out.format(
                                    "+----+----------------------+------------
-+----------------------+---------------+------------------+-----------------
+%n");
```

```
            } else {
                  System.out.println("Không tồn tại phòng này trên HT");
            }

                                                                  }
```

4. Question4:

```
public void question4() throws ClassNotFoundException, SQLException,
FileNotFoundException, IOException {
            System.out.println("Kiểm tra tên account đã có trên hệ thông? ");
            System.out.println("Nhập vào tên cần kiểm tra: ");
            String nameCheck = ScannerUltis.inputString();
            Boolean checkResult = accDAO.isAccNameExists(nameCheck);
            if (checkResult) {
                  System.out.println("Tên đã có trên hệ thống.");
                  question1();
            } else {
                  System.out.println("Tên chưa có trên hệ thống.");
            }

                                                                  }
```

5. Question5:

```
public void question5() throws FileNotFoundException, ClassNotFoundException,
IOException, SQLException {
            Account acc = new Account();
            System.out.println("Nhập vào Email: ");
            acc.setEmail(ScannerUltis.inputString());
            System.out.println("Nhập vào UserName: ");
            acc.setUsername(ScannerUltis.inputString());
            System.out.println("Nhập vào FullName: : ");
            acc.setFullName(ScannerUltis.inputString());
            System.out.println("Hãy chọn phòng nhân viên: ");
            int depid = getDep();
            System.out.println("Hãy chọn Position nhân viên: ");
            int posid = getPos();
            if (accDAO.createAccount(acc, depid, posid)) {
                  System.out.println("Tạo thành công: ");
                  question1();
            } else {
                  System.out.println("Tạo không thành công, hãy kiểm tra
lại");
            }

                                                                  }
```

➔Hàm getpos();

```
private int getPos() throws ClassNotFoundException, SQLException,
FileNotFoundException, IOException {
            while (true) {
                  PositionDao posDAO = new PositionDao();
                  List<Position> listpos = posDAO.getListPosition();
                  String leftAlignFormat = "| %-6d | %-21s |%n";
```

```
                    System.out.format("+--------+----------------------+%n");
                    System.out.format("|   ID   | Position Name        |%n");
                    System.out.format("+--------+----------------------+%n");
                    for (Position position : listpos) {
                          System.out.format(leftAlignFormat, position.getId(),
position.getName());
                    }
                    System.out.format("+--------+----------------------+%n");
                    System.out.println("Chọn Position theo ID:");
                    int choosePos = ScannerUltis.inputIntPositive();
                    ;
                    if (posDAO.getPosByID(choosePos) != null) {
                          return choosePos;
                    } else {
                          System.out.println("Không có Position này, hãy chọn
lại: ");
                    }
              }
                                                                  }
```

Hàm getDep

```
      private int getDep() throws FileNotFoundException, IOException,
ClassNotFoundException, SQLException {
              while (true) {
                    DepartmentDao depDAO = new DepartmentDao();
                    List<Department> listDep = depDAO.getListDepartment();
                    String leftAlignFormat = "| %-6d | %-21s |%n";

                    System.out.format("+--------+----------------------+%n");
                    System.out.format("|   ID   | Depament Name        |%n");
                    System.out.format("+--------+----------------------+%n");
                    for (Department department : listDep) {
                          System.out.format(leftAlignFormat,
department.getId(), department.getName());
                    }
                    System.out.format("+--------+----------------------+%n");
                    System.out.println("Chọn phòng theo ID:");
                    int chooseDep = ScannerUltis.inputIntPositive();
                    ;
                    if (depDAO.getDepByID(chooseDep) != null) {
                          return chooseDep;
                    } else {
                          System.out.println("Không có phòng này, hãy chọn lại:
");
                    }
              }

                                                                  }
```

6. Question6:

MenuQuestion 6:

```
      private int getMenuQues6() {
              while (true) {
                    System.out.println("Bạn muốn update trường nào??");
```

```
                    System.out.println("1.Email, 2.UserName, 3.FullName,
4.Department, 5.Position, 6.Exit ");
                    int i = ScannerUltis.inputIntPositive();
                    if (i == 1 || i == 2 || i == 3 || i == 4 || i == 5 || i ==
6) {
                          return i;
                    } else {
                          System.out.println("Chọn lại: ");
                    }
             }

                                                                  }
```

```
public void question6() throws ClassNotFoundException, FileNotFoundException,
SQLException, IOException {
             while (true) {
                    switch (getMenuQues6()) {
                    case 1:
                          int id = getidCase1();
                          System.out.println("Nhập vào New Email: ");
                          String newEmail = ScannerUltis.inputString();
                          if (accDAO.updateByEmai(id, newEmail)) {
                                System.out.println("Update thành công.");
                                question1();
                          } else {
                                System.out.println("update không thành công,
kiểm tra lại.");
                          }
                          break;
                    case 2:
                          int id2 = getidCase1();
                          System.out.println("Nhập vào New UserName: ");
                          String newUserName = ScannerUltis.inputString();
                          if (accDAO.updateByUserName(id2, newUserName)) {
                                System.out.println("Update thành công.");
                                question1();
                          } else {
                                System.out.println("update không thành công,
kiểm tra lại.");
                          }
                          break;
                    case 3:
                          int id3 = getidCase1();
                          System.out.println("Nhập vào New FullName: ");
                          String newFullName = ScannerUltis.inputString();
                          if (accDAO.updateByFullName(id3, newFullName)) {
                                System.out.println("Update thành công.");
                                question1();
                          } else {
                                System.out.println("update không thành công,
kiểm tra lại.");
                          }
                          break;
                    case 4:
                          int id4 = getidCase1();
                          int idDep = getNewIDDep();
                          if (accDAO.updateByDepId(id4, idDep)) {
                                System.out.println("Update thành công.");
```

```
                                question1();
                        } else {
                                System.out.println("Có lỗi xảy ra, Hãy kiểm tra
lại");
                        }
                        break;
                case 5:
                        int id5 = getidCase1();
                        int idPos = getNewIDPos();
                        if (accDAO.updateByPosId(id5, idPos)) {
                                System.out.println("Update thành công.");
                                question1();
                        } else {
                                System.out.println("Có lỗi xảy ra, Hãy kiểm tra
lại");
                        }
                        break;
                case 6:
                        return;
                }
        }
                                                                        }
```

DAO

```
        public boolean updateByEmai(int id, String newEmail) throws
ClassNotFoundException, SQLException {
                String sql = "UPDATE account SET Email = ? WHERE (AccountID =
?);";
                PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
                preStatement.setString(1, newEmail);
                preStatement.setInt(2, id);
                int result = preStatement.executeUpdate();
                if (result == 1) {
                        jdbc.disConnection();
                        return true;
                } else {
                        jdbc.disConnection();
                        return false;
                }

        }
        public boolean updateByUserName(int id, String newUserName) throws
ClassNotFoundException, SQLException {
                String sql = "UPDATE account SET Username = ? WHERE (AccountID =
?);";
                PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
                preStatement.setString(1, newUserName);
                preStatement.setInt(2, id);
                int result = preStatement.executeUpdate();
                if (result == 1) {
                        jdbc.disConnection();
                        return true;
                } else {
                        jdbc.disConnection();
                        return false;
                }

        }
```

```
	public boolean updateByFullName(int id, String newFullName) throws
ClassNotFoundException, SQLException {
		String sql = "UPDATE account SET FullName = ? WHERE (AccountID =
?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setString(1, newFullName);
		preStatement.setInt(2, id);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}

	}

	public boolean updateByDepId(int id, int idDep) throws
ClassNotFoundException, SQLException {
		String sql = "UPDATE account SET DepartmentID = ? WHERE (AccountID
= ?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setInt(1, idDep);
		preStatement.setInt(2, id);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}

	}

	public boolean updateByPosId(int id, int idPos) throws
ClassNotFoundException, SQLException {
		String sql = "UPDATE account SET PositionID = ? WHERE (AccountID =
?);";
		PreparedStatement preStatement = jdbc.createPrepareStatement(sql);
		preStatement.setInt(1, idPos);
		preStatement.setInt(2, id);
		int result = preStatement.executeUpdate();
		if (result == 1) {
			jdbc.disConnection();
			return true;
		} else {
			jdbc.disConnection();
			return false;
		}
```


7. Question7:

```
public void question7() throws ClassNotFoundException, FileNotFoundException,
SQLException, IOException {
		int id = getIdQues7();
```

```
            if (accDAO.delAccByID(id)) {
                  System.out.println("Xóa thành công");
                  question1();
            } else {
                  System.out.println("Đã có lỗi xảy ra.");
            }
                                                                  }
```

Hàm getIdQues7()

```
private int getIdQues7() throws ClassNotFoundException, FileNotFoundException,
SQLException, IOException {
            while (true) {
                  System.out.println("Nhập vào ID Account cần xóa: ");
                  int id = ScannerUltis.inputIntPositive();
                  if (accDAO.getAccByID(id) != null) {
                        return id;
                  } else {
                        System.out.println("Không có Account này trên hệ
thống, Nhập lại: ");
                  }
            }

                                                                  }
```

Exercise 3 (Optional): Call Procedure

Question 1:

Tạo method để yêu cầu người dùng nhập vào id của department, sau đó sẽ xóa department đó. (sử dụng store procedure ở trong MySQL)

Gợi ý:

B1: Tạo 1 store procedure trong MySQL tên là sp_delete_department() có in parameter là id

B2: Trên backend sẽ viết method **void deleteDepartmentUsingProcedure(int id)**

Check xem đã có department nào có id như id parameter chưa (sử dụng method ở Question 6 để check)

Nếu không tìm thấy department có id = id parameter thì sẽ throw ra Exception có message "Cannot find department which has id = " + id

Nếu tìm thấy thì sẽ delete department đó

Nếu có lỗi sẽ throw Exception lên front-end để in ra

Trên front-end sẽ gọi class DepartmentDao và demo method này (dùng scanner để nhập vào id của department muốn delete)

Exercise 4 (Optional): Transaction

Question 1:

Tạo 1 method để thực thi các bước sau:

B1: xóa tất cả các nhân viên thuộc phòng ban 3

B2: xóa phòng ban 3

Với điều kiện: khi xóa B1 bị lỗi thì sẽ rollback lại

Exercise 5: Tạo JdbcUtils

Question 1:

Tạo method để test xem đã kết nối thành công tới sql chưa?

Gợi ý: Tạo method **void isConnectedForTesting()** và trong method sẽ thực hiện kết nối tới database và in ra "Connect success!", nếu không thì sẽ throw ra exception

Question 2:

Tạo method để get Connect tới database.

Gợi ý:

Tạo method **void connect()** và trong method sẽ thực hiện check đã có connect hiện tại chưa (null & closed).

Nếu chưa connect thì sẽ thực hiện connect và return về connect vừa lấy được

Nếu connect rồi thì chỉ cần return ra connect hiện tại (không cần phải connect lại tới database)

Trong khi thực hiện nếu có lỗi gì thì sẽ throw ra Exception (để message Exception cho phù hợp)

Và Các constant nên config ở trong file properties (VD như : username, password, driverName, message exception)

Question 3:
Tạo method để get disconnect tới database.
Gợi ý:
Tạo method **void disconnect()** và trong method sẽ thực hiện
check đã có connect hiện tại chưa (null & closed).
Nếu chưa connect thì sẽ không làm gì cả
Nếu đã connect rồi thì disconnect
Trong khi thực hiện nếu có lỗi gì thì sẽ throw ra Exception (để
message Exception cho phù hợp)