CONNECT DB

```
package DAO;
import Entity._Member;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class ConnectDB {
    private SessionFactory factory;

    public ConnectDB() {
        try {
            Configuration configuration = new Configuration();

            configuration.addAnnotatedClass(_Member.class);
            configuration.configure("hibernate.cfg.xml");

            factory = configuration.buildSessionFactory();

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public SessionFactory getFactory() {
        return factory;
    }
}
```

READ

```
public List<_Device> getAllDevices() {
    List<_Device> results = new ArrayList<>();
      try {
         results = session.createQuery("FROM _Device").getResultList();
      } catch (Exception e) {
         e.printStackTrace();
      } finally {
         session.close();
      }
      return results;
  }
```

CREATE

```
public void addMember(_Member member) throws Exception {
    Transaction transaction = null;
    try {
        transaction = session.beginTransaction();
        session.save(member);
        transaction.commit();
    } catch (Exception e) {
        if (transaction != null && transaction.isActive()) {
            transaction.rollback();
        }
        e.printStackTrace();
    } finally {
        session.close();
    }
}
```

UPDATE

```
public void updateMember(_Member member) throws Exception {
    Transaction transaction = null;
    try {
        transaction = session.beginTransaction();
        session.update(member);
        transaction.commit();
    } catch (Exception e) {
        if (transaction != null) {
            transaction.rollback();
        }
        e.printStackTrace();
        throw e;
    } finally {
        session.close();
    }
}
```

DELETE

```
public boolean deleteMember(int memberId) {
        Transaction transaction = null;
        try {
            transaction = session.beginTransaction();


            Query query = session.createQuery("DELETE FROM _Member WHERE maTV
= :maTV");
            query.setParameter("maTV", memberId);


            int rowsAffected = query.executeUpdate();
            transaction.commit();
            return rowsAffected >= 1;
        } catch (Exception e) {
            if (transaction != null) {
                transaction.rollback();
            }
            e.printStackTrace();
            return false;
        } finally {
            session.close();
        }
    }
```

BÀI TẬP QUẢN LÝ KHÓA HỌC

GUI

```
package GUI;

import Entity.Khoahoc;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import java.text.ParseException;
import java.util.Date;
import java.util.List;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.text.SimpleDateFormat;

public class Khoahoc_GUI {

    private JFrame frame;
    private JTable khoahocTable;
    private JTable thongkedonviTable;
    private JTextField ngayBatDauTextField;
    private JTextField searchNameTextField;
    private DefaultTableModel model;

    private JComboBox<String> orderCombobox;
```

```
    public Khoahoc_GUI() {
        try {
            frame = new JFrame("Danh Sach Khoa Hoc");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(1000, 600);

            showFilter();
            showTable();
            showThongKeDonVi();

            JScrollPane scrollPane1 = new JScrollPane(khoahocTable);
            JScrollPane scrollPane2 = new JScrollPane(thongkedonviTable);
            JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT,
scrollPane1, scrollPane2);
            splitPane.setDividerLocation(300);
            frame.getContentPane().add(splitPane, BorderLayout.CENTER);

            frame.setVisible(true);
        } catch (Exception e) {
        }
    }

    private void showFilter() {
        JPanel topPanel = new JPanel();
        topPanel.setLayout(new FlowLayout());

        JLabel searchLabel = new JLabel("Tìm kiếm tên:");
```

```
    searchNameTextField = new JTextField(10);

    JLabel ngayBatDauLabel = new JLabel("Ngày bắt đầu:");
    ngayBatDauTextField = new JTextField(10);

    JLabel ordeByNameLabel = new JLabel("Sắp xếp tên:");
    orderCombobox = new JComboBox<>();
    orderCombobox.addItem("Giảm dần");
    orderCombobox.addItem("Tăng dần");

    JButton filterButton = new JButton("Lọc");
    filterButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            showTable();
        }
    });
    topPanel.add(searchLabel);
    topPanel.add(searchNameTextField);
    topPanel.add(ngayBatDauLabel);
    topPanel.add(ngayBatDauTextField);
    topPanel.add(ordeByNameLabel);
    topPanel.add(orderCombobox);
    topPanel.add(filterButton);
    frame.add(topPanel, BorderLayout.NORTH);
}
```

```
    private void setUpTableModel() {
        model = new DefaultTableModel();
        model.addColumn("ID");
        model.addColumn("Tên");
        model.addColumn("Đơn vị");
        model.addColumn("Học phí");
        model.addColumn("Ngày bắt đầu");
        khoahocTable = new JTable(model);
    }

    private void showTable() {
        try {
            String order = (String) orderCombobox.getSelectedItem();
            // orderCombobox.getSelectedIndex()
            if (model == null) {
                setUpTableModel();
            }
            SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy HH:mm");

            // Date date = formatter.parse(ngayBatDauTextField.getText());
            // List<Khoahoc> dsKhoaHoc = new
BUS.Khoahoc_BUS().readByNgayBatDau(date);
            // Date start = formatter.parse("19/12/2022");
            // Date end = formatter.parse("20/12/2023");
            // dsKhoaHoc = new
BUS.Khoahoc_BUS().readByNgayBatDauVoiKhoangThoiGian(start,
            // end);
```

```
        List<Khoahoc> dsKhoaHoc = new
BUS.Khoahoc_BUS().read(searchNameTextField.getText(), order);


        model.setRowCount(0);
        for (int i = 0; i < dsKhoaHoc.size(); i++) {
            Object[] row = new Object[5];
            row[0] = dsKhoaHoc.get(i).getId();
            row[1] = dsKhoaHoc.get(i).getTen();
            row[2] = dsKhoaHoc.get(i).getDonvi();
            row[3] = dsKhoaHoc.get(i).getHocphi();
            row[4] = formatter.format(dsKhoaHoc.get(i).getNgaybatdau());
            model.addRow(row);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

  private void showThongKeDonVi() {
    try {
        DefaultTableModel thongkedonviModel = new DefaultTableModel();
        thongkedonviModel.addColumn("ID");
        thongkedonviModel.addColumn("Tên");
        thongkedonviModel.addColumn("Đơn vị");
        thongkedonviModel.addColumn("Tổng khóa học");
        thongkedonviModel.addColumn("Tổng học phí");
        thongkedonviTable = new JTable(thongkedonviModel);
```

```
            List<Object[]> ketquathongke = new BUS.Khoahoc_BUS().thongke();

            thongkedonviModel.setRowCount(0);
            for (int i = 0; i < ketquathongke.size(); i++) {
                Object[] row = new Object[5];
                Khoahoc khoahoc = (Khoahoc) ketquathongke.get(i)[0];
                row[0] = khoahoc.getId();
                row[1] = khoahoc.getTen();
                row[2] = khoahoc.getDonvi();
                row[3] = ketquathongke.get(i)[1];
                row[4] = ketquathongke.get(i)[2];
                thongkedonviModel.addRow(row);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        new Khoahoc_GUI();
    }
}
```

```
package BUS;

import DAO.DAO_KHOAHOC;
import Entity.Khoahoc;
import java.util.Date;
import java.util.List;

public class Khoahoc_BUS {

    DAO_KHOAHOC dao_khoahoc;

    public Khoahoc_BUS() {
        dao_khoahoc = new DAO_KHOAHOC();
    }

    public List<Khoahoc> read(String ten, String order) {
        return dao_khoahoc.read(ten, order);
    }

    public List<Object[]> thongke() {
        return dao_khoahoc.thongke();
    }

    public List<Khoahoc> readByDonVi(String donvi) {
        return dao_khoahoc.readByDonVi(donvi);
    }
```

```
    public List<Khoahoc> readByHocPhi(double hocphi) {
        return dao_khoahoc.readByHocPhi(hocphi);
    }

    public List<Khoahoc> readByNam(int nam) {
        return dao_khoahoc.readByNam(nam);
    }

    public List<Khoahoc> readByThang(int thang) {
        return dao_khoahoc.readByThang(thang);
    }

    public List<Khoahoc> readByNgay(int ngay) {
        return dao_khoahoc.readByNgay(ngay);
    }

    public List<Khoahoc> readByNgayBatDauVoiKhoangThoiGian(Date batdau,
Date ketthuc) {
        return dao_khoahoc.readByNgayBatDauVoiKhoangThoiGian(batdau, ketthuc);
    }

    public List<Khoahoc> readByNgayBatDau(Date ngaybatdau) {
        return dao_khoahoc.readByNgayBatDau(ngaybatdau);
    }
}
```

```
package DAO;

import Entity.Khoahoc;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import java.util.Calendar;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.query.Query;

public class DAO_KHOAHOC {

    private SessionFactory factory;
    Session session;

    public DAO_KHOAHOC() {
        factory = new ConnectDB().getFactory();
        session = factory.openSession();
    }

    public List<Khoahoc> read(String ten, String order) {
        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            String hql = "FROM Khoahoc";
```

```
        boolean hasTen = !"".equals(ten);
        if (hasTen) {
            hql += " WHERE ten LIKE :ten";
        }

        hql += " ORDER BY ten";
        if (order.equals("Giảm dần")) {
            hql += " DESC";
        } else {
            hql += " ASC";
        }

        Query query = session.createQuery(hql);
        if (hasTen) {
            query.setParameter("ten", "%" + ten + "%");
        }
        dsKhoaHoc = query.getResultList();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        session.close();
    }
    return dsKhoaHoc;
}

public List<Object[]> thongke() {
    List<Object[]> ketqua = new ArrayList<>();
```

```
        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            String hql = "SELECT kh,COUNT(kh.donvi),SUM(kh.hocphi) FROM Khoahoc
kh GROUP BY donvi";
            Query query = session.createQuery(hql);
            ketqua = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
        return ketqua;
    }

    public List<Khoahoc> readByDonVi(String donvi) {
        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            Query query = session.createQuery("FROM Khoahoc WHERE donvi = :donvi");
            query.setParameter("donvi", donvi);
            dsKhoaHoc = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
        return dsKhoaHoc;
    }
```

```
    public List<Khoahoc> readByHocPhi(double hocphi) {
        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            Query query = session.createQuery("FROM Khoahoc WHERE hocphi >=
:hocphi");
            query.setParameter("hocphi", hocphi);
            dsKhoaHoc = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
        return dsKhoaHoc;
    }

    public List<Khoahoc> readByNam(int nam) {
        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            Query query = session.createQuery("FROM Khoahoc WHERE
YEAR(ngaybatdau) = :nam");
            query.setParameter("nam", nam);
            dsKhoaHoc = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
```

```
        return dsKhoaHoc;
    }

    public List<Khoahoc> readByThang(int thang) {
        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            Query query = session.createQuery("FROM Khoahoc WHERE
MONTH(ngaybatdau) = :thang");
            query.setParameter("thang", thang);
            dsKhoaHoc = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
        return dsKhoaHoc;
    }

    public List<Khoahoc> readByNgay(int ngay) {
        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            Query query = session.createQuery("FROM Khoahoc WHERE DAY(ngaybatdau)
= :ngay");
            query.setParameter("ngay", ngay);
            dsKhoaHoc = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
```

```
            session.close();
        }
        return dsKhoaHoc;
    }

    public List<Khoahoc> readByNgayBatDau(Date ngaybatdau) {
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(ngaybatdau);

        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            Query query = session.createQuery("FROM Khoahoc WHERE DAY(ngaybatdau)
= :ngay "
                    + "AND MONTH(ngaybatdau) = :thang "
                    + "AND YEAR(ngaybatdau) = :nam");
            int ngay = calendar.get(Calendar.DAY_OF_MONTH);
            int thang = calendar.get(Calendar.MONTH) + 1;
            int nam = calendar.get(Calendar.YEAR);
            query.setParameter("ngay", ngay);
            query.setParameter("thang", thang);
            query.setParameter("nam", nam);
            dsKhoaHoc = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
```

```
        return dsKhoaHoc;
    }

    public List<Khoahoc> readByNgayBatDauVoiKhoangThoiGian(Date batdau,
Date ketthuc) {
        List<Khoahoc> dsKhoaHoc = new ArrayList<>();
        try {
            Query query = session.createQuery("FROM Khoahoc WHERE ngaybatdau >=
:batdau AND ngaybatdau <= :ketthuc");
            query.setParameter("batdau", batdau);
            query.setParameter("ketthuc", ketthuc);
            dsKhoaHoc = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            session.close();
        }
        return dsKhoaHoc;
    }
}
```

3 LAYERS AND MVC

- MVC có lớp controller chuyên về xử lý request của người dùng nên thích hợp với web trong khi 3 layers ko có lớp nào chuyên xử lý request của user
- MVC tách phần xử lý Request và Giao diện thành Controller và View giúp khi thay đổi controller mà vẫn giữ nguyên View và ngược lại
- Nếu dự án lớn, với nhiều logic nghiệp vụ phức tạp, thì việc tách BL và DAL để chúng ta có thể thay đổi mã BL mà không thay đổi hợp ngữ DAL trong khi đó thì MVC gộp BL và DAL vào trong Model
- Có thể sử dụng một tổ hợp DAL khác cho một cơ sở dữ liệu khác với cùng một BL.

MVC	Three Layers
focuses on the separation of concerns within the user interface	focuses on the overall organization of an application into distinct layers based on functionality
MVC is a triangle architecture.	3-layer is a linear architecture.
In MVC architecture, user interacts with the controller with the help of view	In 3-layer architecture, user interacts with the Presentation layer.
Model: Đây là thành phần chứa tất cả các nghiệp vụ logic, phương thức xử lý, truy xuất database, đối tượng mô tả dữ liệu như các Class, hàm xử lý…	Tại BLL, các thông tin sẽ được nhào nặn, tính toán theo đúng yêu cầu đã gửi, nếu không cần đến Database thì BLL sẽ gửi trả kết quả về GUI, ngược lại nó sẽ đẩy dữ liệu (thông tin đã xử lý) xuống Data Access Layer (DAL).

TRẮC NGHIỆM

1. Thiết kế cho một kiến trúc ứng dụng có thể có bao nhiêu thiết kế khác nhau?

a. 1 b. 2 c. 3 **d. Nhiều**

2. Chọn thứ tự sắp xếp của Project Life Cycle đúng?

1. Project Initiation; 2. Planning and Prototyping; 3. Project Construction; 4. Project Trasition and Release

a. **1-2-3-4** b. 2-1-3-4 c. 3-2-1-4 d. Tất cả đều đúng

3. Thiết kế các use case nằm trong giai đoạn nào của Project Life Cycle?

a. Project Initiation

b. Planning and Prototyping

c. Project Construction

d. Project Transition and Release

4. Chọn phát biểu sai của quy trình SCRUM?

a. Mỗi lần lặp có quy định số ngày cụ thể

b. Mỗi lần lặp đều phải đủ các bước: planning, requirements, design, coding, testing, and documentation

c. Chỉ cho tester kiểm tra mà không cho khách hàng sử dụng thử

d. Mục tiêu tạo sản phẩm cuối mỗi lần lặp không có lỗi

5. Một Windows-based project theo kiến trúc tier thì thường có các lớp nào?

a. Windows form, Bussiness Layer, Data Access Layer

b. Windows form, Controller, Model

c. Windows form, Bussiness Layer, Data Layer

d. Windows form, Data Access Layer

6. Bussiness Layer không có chức năng nào sau đây?

a. Chuyển đổi dữ liệu

b. Lưu cập nhật dữ liệu sau khi áp dụng các luật

c. Hiển thị CSDL

d. Kiểm soát lỗi dữ liệu nhập

7. Cách chuyển từ mô hình 1 tier – Single layer thành 1 tier – Three layer như thế nào?

a. Tách UI, BL, DAL thành 3 class khác nhau

b. Tách UI vào 01 namespace, còn BL và DAL vào cùng 01 namespace

c. Tách UI, BL, DAL vào 03 namespcae khác nhau

d. Tách UI, BL, DAL, Database vào 04 namespace khác nhau

8. Chọn thứ tự sắp xếp đúng của các layer trong mô hình 1 tier – Three layer?

a. Viewer → UI ↔ DAL ↔ BL

b. Viewer → BL ↔ UI ↔ DAL

c. Viewer → UI ↔ BL ↔ DAL

d. Viewer → UI ↔ DAL

9. Xác định kiểu quan hệ giữa sinh viên và thời khoá biểu?

a. Dependency b. Aggregation **c. Composition** d. Không có

10. Xác định kiểu quan hệ giữa sản phẩm và hoá đơn?

a. Dependency b. Aggregation c. Composition **d. Không có**

11. Code cho lớp DAL như thế nào là đúng trong mô hình Three Layer?

a. Xử lý dữ liệu nhập vào từ UI

b. Trả kết quả cho UI như các dialog

c. Chỉ viết các hàm truy vấn CSDL

d. Chuyển đổi kiểu dữ liệu cho BL

12. Trong mô hình MVC, lớp nhận sự kiện là?

a. Receiver **b. Controller** c. View d. Transmitter

13. Route mặc định trong MVC?

a. "/{action}/{controller}/{id}

b. "{controller}/{id}

c. "{controller}/{action}/{id}

d. {controller}/{action}

14. Trong mô hình MVC, model được định nghĩa là lớp?

a. Lớp data access

b. Lớp Bussiness logic

c. Lớp Presentation

d. Lớp Interface

15. Quan hệ giữa 2 thực thể Person và Student

a. Dependency b. Aggregation c. Composition **d. Inheritance**

16. Ưu điểm của mô hình MVC?

a. Tách dự án thành nhiều phần, giúp nhà phát triển làm việc dễ dàng. Có thể chỉnh sửa, thay đổi một số phần, giúp tiết kiệm chi phí phát triển và bảo trì

b. Phù hợp cho các dự án nhỏ

c. Phù hợp cho lập trình viên ít kinh nghiệm

d. Tất cả ý trên

17. Thiết kế giao diện UI nằm trong giai đoạn nào của PLC?

a. Project Initiation

b. Planning and Prototyping

c. Project Construction

d. Project Transition and Release

18. Quan hệ nào giữa 2 entity mà khi xoá entity 1 thì sẽ ảnh hưởng đến entity 2?

a. Dependency b. Aggregation **c. Composition** d. Không có

19. Quan hệ "is a" là kiểu quan hệ?

a. Dependency b. Aggregation c. Composition **d. Inheritance**

20. Code cho lớp BL như thế nào là đúng trong mô hình Three Layer?

a. Gọi hàm của lớp UI

b. Gọi hàm của lớp DAL

c. Trả về thông báo qua Dialog

d. Có tham số là các textbox

Hibernate

2. Phát biểu nào sau đây đúng về Hibernate?
a. Hệ quản trị CSDL quan hệ
b. Công cụ đặc tả CSDL
c. Chương trình hướng đối tượng của Java
d. Công cụ phát triển web front-end
3. Mục tiêu của SessionFactory trong Hibernate?
a. Tạo và quản lý kết nối CSDL
b. Định nghĩa CSDL

c. Cung cấp dịch vụ ánh xạ giữa class và bảng của CSDL

d. Để quản lý các transaction của CSDL

4. Tập tin cấu hình trong Hibernate?

a. hibernate.cfg.xml

b. web.xml

c. persistence.xml

d. application.properties

5. Mục tiêu của đối tượng Session trong Hibernate?

a. Thực thi một truy vấn

b. Thực hiện một xử lý CRUD

c. Để quản lý các transaction

d. Để định nghĩa thực thể ánh xạ

6. Loại khoá chính được sinh mặc định bởi Hibernate?

a. AUTO b. IDENTITY c. SEQUENCE d.TABLE

7. Kí hiệu nào dùng để đặc tả khoá chính trong lớp thực thể?

a. @PrimaryKey **b. @Id** c. @GeneratedValue d. @Key

8. Mục tiêu của HQL (Hibernate Query Language)?

a. To define database schemas

b. To create Java objects from database tables

c. To perform databasen CRUD operations

d. It allows developers to write database queries using entity and property names

9. Kiểu tải lazy trong Hibernate là gì?

a. Loading data from the database only when needed

b. Loading all data eagerly during initialization

c. Loading data in a separate thread

d. Loading data from a cache

10. Kí hiệu nào dùng để ánh xạ cho kiểu quan hệ kết hợp 1-n?

a. @OneToMany

b. @ManyToOne

c. @OneToOne

d. @ManyToMany

11. Mục tiêu của EntityManager Hibernate?

a. Quản lý kết nối CSDL

b. Định nghĩa schemas cho CSDL

c. Quản lý các transaction

d. To persits and retrieve entities

12. Các xử lý nào được hỗ trợ bởi EntityManager?

a. Create và update

b. Read và delete

c. Create, read, update và delete

d. Read và update

13. Kí hiệu nào dùng để ánh xạ class với bảng trong CSDL của Hibernate?

a. @Entity b. @Table c. @Column d. @Id

15. Vai trò của đối tượng Transaction trong Hibernate?

a. To execute SQL queries

b. To manage database connections

c. To handle database transactions

d. To define Hibernate entity mappings

16. Kí hiệu nào dùng để ánh xạ kiểu quan hệ many-to-many trong Hibernate?

a. @OneToMany

b. @ManyToOne

c. @OneToOne

d. @ManyToMany

17. Vai trò của Criteria API trong Hibernate?

a. Định nghĩa schemas của CSDL

b. Thực hiện một nghiệp vụ CRUD

c. Thực thi truy vấn SQL

d. Xây dựng quy vấn an toàn

18. Kí hiệu nào dùng để xác định ánh xạ cột của bảng trong lớp entity?

a. @PrimaryKey b. @Id c. @GeneratedValue **d. @Column**