HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ TẠI TP. HCM

KHOA CÔNG NGHỆ THÔNG TIN 2

BÁO CÁO MÔN KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM

Đề tài: Bài tập nhóm chương 5 & chương 6

Giảng viên phụ trách: Thầy Nguyễn Văn Hữu Hoàng

Lớp: D21CQCNPM01 – N

Sinh viên thực hiện:

1. Nguyễn Ngọc Thiên Phúc – N21DCCN066
2. Trần Thị Thùy Ngân – N21DCCN055

TP. Hồ Chí Minh, ngày 17 tháng 04 năm 2025

BẢNG PHÂN CÔNG CÔNG VIỆC

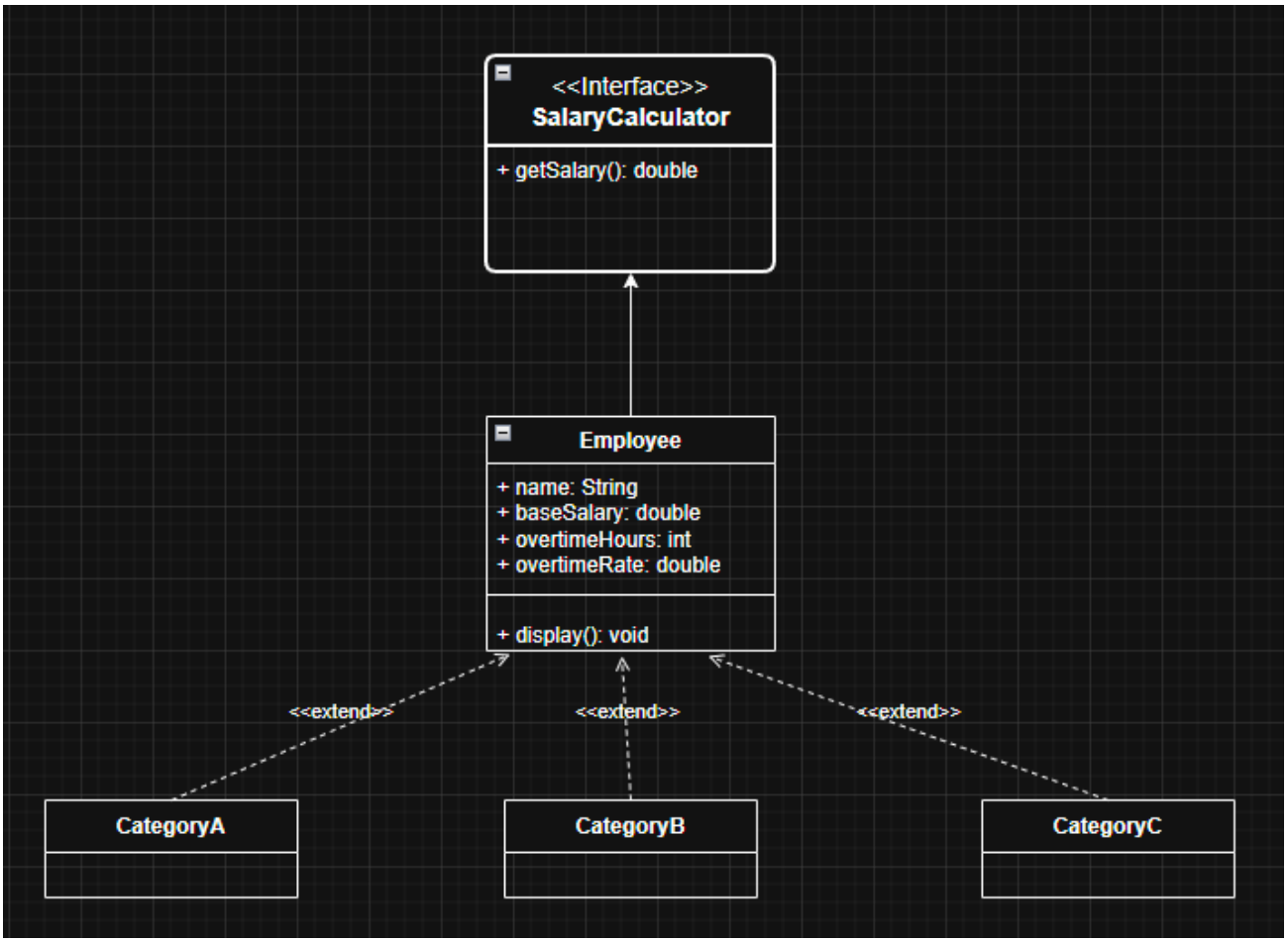
Thành viên	Nhiệm vụ
Nguyễn Ngọc Thiên Phúc – N21DCCN066	Bài tập chương 6 + Làm báo cáo
Trần Thị Thùy Ngân – N21DCCN055	Bài tập chương 5

MỤC LỤC

CHƯƠNG 5

BÀI 1:

1.1: Biểu đồ UML:



1.2: Code:

```
interface SalaryCalculator {
    double getSalary();
}

class Employee implements SalaryCalculator {
    protected String name;
```

```
    protected double baseSalary;
    protected int overtimeHours;
    protected double overtimeRate;

    public Employee(String name, double baseSalary, int overtimeHours, double
overtimeRate) {
        this.name = name;
        this.baseSalary = baseSalary;
        this.overtimeHours = overtimeHours;
        this.overtimeRate = overtimeRate;
    }

    @Override
    public double getSalary() {
        return baseSalary + (overtimeHours * overtimeRate);
    }

    public void display() {
        System.out.println("Name: " + name);
        System.out.println("Salary: " + getSalary());
    }
}
```

```
class CategoryA extends Employee {
    public CategoryA(String name, int overtimeHours) {
        super(name, 1500, overtimeHours, 10);
    }
}

class CategoryB extends Employee {
    public CategoryB(String name, int overtimeHours) {
        super(name, 1700, overtimeHours, 12);
    }
}

class CategoryC extends Employee {
    public CategoryC(String name, int overtimeHours) {
        super(name, 600, overtimeHours, 5);
    }
}

public class MainApp {
    public static void main(String[] args) {
        Employee employee1 = new CategoryA("Phuc", 10);
```

```
        Employee employee2 = new CategoryB("Ngan", 20);
        Employee employee3 = new CategoryC("Ruffa", 15);

        System.out.println("Giam doc: ");
        employee1.display();

        System.out.println("\nQuan ly ban hang: ");
        employee2.display();

        System.out.println("\nNhan vien ban hang: ");
        employee3.display();
    }
}
```

BÀI 2:

```
interface Search {
  int search(Book[] books, String title);
}
class Book {
  private String title;

  public Book(String title) {
    this.title = title;
```

```
}

    public String getTitle() {
        return title;
    }
}

class LinearSearch implements Search {
    @Override
    public int search(Book[] books, String title) {
        for (int i = 0; i < books.length; i++) {
            if (books[i].getTitle().equals(title)) {
                return i;
            }
        }
        return -1;
    }
}

class BinarySearch implements Search {
    @Override
    public int search(Book[] books, String title) {
```

```
        int left = 0, right = books.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            int comparison = books[mid].getTitle().compareTo(title);
            if (comparison == 0) {
                return mid;
            } else if (comparison < 0) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1;
    }
}

public class MainApp {
    public static void main(String[] args) {
        Book[] books = {
                new Book("Data Structures and Algorithms"),
                new Book("Object Oriented Programming"),
                new Book("Developing Java Applications"),
```

```
            new Book("Design Patterns"),
            new Book("Storytelling with Data"),
        };

        java.util.Arrays.sort(books, (a, b) -> a.getTitle().compareTo(b.getTitle()));

        Search linearSearch = new LinearSearch();
        int linearResult = linearSearch.search(books, "Data Structures and Algorithms");
        System.out.println(
            "Linear Search: "
                + (linearResult != -1 ? "Position: " + linearResult : "Cannot find"));

        Search binarySearch = new BinarySearch();
        int binaryResult = binarySearch.search(books, "Design Patterns");
        System.out.println(
            "Binary Search: "
                + (binaryResult != -1 ? "Position: " + binaryResult : "Cannot find"));
    }
}
```

BÀI 3:

```
interface AddressValidator {
```

```
    boolean validateStreet(String street);

    boolean validateCity(String city);

    boolean validatePostalCode(String postalCode);

    boolean validateCountry(String country);
}

class USAAddress implements AddressValidator {
    @Override
    public boolean validateStreet(String street) {
        return street != null && !street.isEmpty();
    }

    @Override
    public boolean validateCity(String city) {
        return city != null && !city.isEmpty();
    }

    @Override
    public boolean validatePostalCode(String postalCode) {
```

```
        return postalCode != null && postalCode.matches("\\d{5}(-\\d{4})?");
    }

    @Override
    public boolean validateCountry(String country) {
        return "USA".equalsIgnoreCase(country);
    }
}

class VNAddress implements AddressValidator {
    @Override
    public boolean validateStreet(String street) {
        return street != null && !street.isEmpty();
    }

    @Override
    public boolean validateCity(String city) {
        return city != null && !city.isEmpty();
    }

    @Override
    public boolean validatePostalCode(String postalCode) {
```

```
        return postalCode != null && postalCode.matches("\\d{6}");
    }

    @Override
    public boolean validateCountry(String country) {
        return "Vietnam".equalsIgnoreCase(country) || "VN".equalsIgnoreCase(country);
    }
}

public class MainApp {
    public static void main(String[] args) {
        AddressValidator usaAddress = new USAAddress();
        System.out.println("USA Address Validation:");
        System.out.println("Street valid: " + usaAddress.validateStreet("911 Main Street"));
        System.out.println("City valid: " + usaAddress.validateCity("New York"));
        System.out.println("Postal Code valid: " + usaAddress.validatePostalCode("12345-6789"));
        System.out.println("Country valid: " + usaAddress.validateCountry("USA"));

        AddressValidator vnAddress = new VNAddress();
        System.out.println("\nVietnam Address Validation:");
        System.out.println("Street valid: " + vnAddress.validateStreet("97 Man Thien"));
```

```
        System.out.println("City valid: " + vnAddress.validateCity("Hồ Chí Minh"));
        System.out.println("Postal Code valid: " + vnAddress.validatePostalCode("700000"));
        System.out.println("Country valid: " + vnAddress.validateCountry("Vietnam"));
    }
}
```

BÀI 4:

4.1: Abstract:

```
abstract class Employee {
  private String name;
  private int id;
  public Employee(String name, int id) {
     this.name = name;
    this.id = id;
    }
    public void displayData() {
        System.out.println("Staff id: " + id + ", Name: " + name + ", Monthly salary: " +
calculateMonthlyIncome() + "vnd");
     }
     public String getName() {
        return name;
     }

     public int getId() {
```

```
        return id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setId(int id) {
        this.id = id;
    }

    public abstract double calculateMonthlyIncome();
}

class SalesRep extends Employee {
    private double baseSalary;
    private double commission;

    public SalesRep(String name, int id, double baseSalary, double commission) {
        super(name, id);
        this.baseSalary = baseSalary;
        this.commission = commission;
```

```
    }

    @Override
    public double calculateMonthlyIncome() {
        return baseSalary + commission;
    }
}

class Consultant extends Employee {
    private double hourlyRate;
    private int hoursWorked;

    public Consultant(String name, int id, double hourlyRate, int hoursWorked) {
        super(name, id);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

    @Override
    public double calculateMonthlyIncome() {
        return hourlyRate * hoursWorked;
    }
```

```
}

public class MainApp {
    public static void main(String[] args) {
        Employee salesRep = new SalesRep("Thien Phuc", 168, 3200, 600);
        salesRep.displayData();

        Employee consultant = new Consultant("Thuy Ngan", 412, 60, 180);
        consultant.displayData();
    }
}
```

4.2: Interface:

```
interface IEmployee {
    void displayData();

    double calculateMonthlyIncome();
}

class SalesRep implements IEmployee {
    private String name;
    private int id;
    private double baseSalary;
```

```
    private double commission;

    public SalesRep(String name, int id, double baseSalary, double commission) {
        this.name = name;
        this.id = id;
        this.baseSalary = baseSalary;
        this.commission = commission;
    }

    @Override
    public void displayData() {
        System.out.println(
                "Staff id: " + id + ", Name: " + name + ", Monthly salary: " +
calculateMonthlyIncome() + "đ");
    }

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
```

```
    }

    @Override
    public double calculateMonthlyIncome() {
        return baseSalary + commission;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setId(int id) {
        this.id = id;
    }
}

class Consultant implements IEmployee {
    private String name;
    private int id;
    private double hourlyRate;
    private int hoursWorked;
```

```
    public Consultant(String name, int id, double hourlyRate, int hoursWorked) {
        this.name = name;
        this.id = id;
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

    @Override
    public void displayData() {
        System.out.println(
                "Staff id: " + id + ", Name: " + name + ", Monthly salary: " +
calculateMonthlyIncome() + "đ");
    }

    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public void setId(int id) {
        this.id = id;
    }

    @Override
    public double calculateMonthlyIncome() {
        return hourlyRate * hoursWorked;
    }
}

public class MainApp {
    public static void main(String[] args) {
        IEmployee salesRep = new SalesRep("Phuc", 168, 3000, 300);
        salesRep.displayData();
        IEmployee consultant = new Consultant("Ngan", 412, 50, 100);
        consultant.displayData();
    }
}
```

CHƯƠNG 6

Book.java:

```
enum BookType {
    JAVA,
    PYTHON,
    WEB_DEVELOPMENT
}


abstract class Book {
    protected int id;
    protected String title;
    protected double price;
    protected BookType type;

    public Book(int id, String title, double price, BookType type) {
        this.id = id;
        this.title = title;
        this.price = price;
        this.type = type;
    }

    public abstract void displayInfo();
```

```
public int getId() {
    return id;
}

public String getTitle() {
    return title;
}

public double getPrice() {
    return price;
}

public BookType getType() {
    return type;
}

public void setId(int id) {
    this.id = id;
}

public void setTitle(String title) {
```

```
        this.title = title;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public void setType(BookType type) {
        this.type = type;
    }
}

class JavaBook extends Book {
    public JavaBook(int id, String title, double price, BookType type) {
        super(id, title, price, type);
    }

    @Override
    public void displayInfo() {
        System.out.println("Java book: " + getTitle() + " | Price: $" + getPrice() + " | ID: " +
getId());
    }
```

```
}

class PythonBook extends Book {
    public PythonBook(int id, String title, double price, BookType type) {
        super(id, title, price, type);
    }

    @Override
    public void displayInfo() {
        System.out.println("Python book: " + getTitle() + " | Price: $" + getPrice() + " | ID: " +
getId());
    }
}

class WebDevBook extends Book {
    public WebDevBook(int id, String title, double price, BookType type) {
        super(id, title, price, type);
    }

    @Override
    public void displayInfo() {
        System.out.println("Web Development book: " + getTitle() + " | Price: $" + getPrice() +
" | ID: " + getId());
```

```
    }
}
```

BookFactory.java:

```
class BookFactory {
    public static Book createBook(int id, String title, double price, BookType type) {
        switch (type) {
            case JAVA:
                return new JavaBook(id, title, price, type);
            case PYTHON:
                return new PythonBook(id, title, price, type);
            case WEB_DEVELOPMENT:
                return new WebDevBook(id, title, price, type);
            default:
                throw new IllegalArgumentException("Invalid book: " + type);
        }
    }
}
```

Order.java:

```
public class Order {
    private int orderId;
    private String customerName;
    private int quantity;
```

```
    private double totalPrice;
    private Book book;

    public Order(int orderId, String customerName, Book book, int quantity, double totalPrice) {
        this.orderId = orderId;
        this.customerName = customerName;
        this.quantity = quantity;
        this.totalPrice = totalPrice;
        this.book = book;
    }

    public int getOrderId() {
        return orderId;
    }

    public void setOrderId(int orderId) {
        this.orderId = orderId;
    }

    public String getCustomerName() {
        return customerName;
```

```
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public int getQuantity() {
    return quantity;
}

public void setQuantity(int quantity) {
    this.quantity = quantity;
}

public double getTotalPrice() {
    return totalPrice;
}

public void setTotalPrice(double totalPrice) {
    this.totalPrice = totalPrice;
}
```

```
    public Book getBook() {
        return book;
    }

    public void setBook(Book book) {
        this.book = book;
    }
}
```

Storage.java:

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

class Storage {
    private static Storage instance;
    private List<Book> books;
    private List<Order> carts;
    private List<Order> orders;

    private Storage() {
        books = new ArrayList<>();
        orders = new ArrayList<>();
        carts = new ArrayList<>();
```

```
}

public static Storage getInstance() {
    if (instance == null) {
        instance = new Storage();
    }
    return instance;
}

public List<Book> getBooks() {
    return books;
}

public void setBooks(List<Book> books) {
    this.books = books;
}

public List<Order> getOrders() {
    return orders;
}

public void setOrders(List<Order> orders) {
```

```
        this.orders = orders;
    }

    public List<Order> getCarts() {
        return carts;
    }

    public void setCarts(List<Order> carts) {
        this.carts = carts;
    }
}
```

Main.java:

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        Book javaBook = BookFactory.createBook(1, "Mastering Java", 15.99,
BookType.JAVA);
        Book pythonBook = BookFactory.createBook(2, "Python for Beginners", 25.99,
BookType.PYTHON);
        Book webBook1 = BookFactory.createBook(3, "HTML & CSS Crash Course", 18.50,
BookType.WEB_DEVELOPMENT);
```

```
        Book webBook2 = BookFactory.createBook(4, "Full-Stack Web Dev", 20.50,
BookType.WEB_DEVELOPMENT);

        Storage.getInstance().getBooks().add(javaBook);
        Storage.getInstance().getBooks().add(pythonBook);
        Storage.getInstance().getBooks().add(webBook1);
        Storage.getInstance().getBooks().add(webBook2);

        System.out.println("Book list:");
        for (Book book : Storage.getInstance().getBooks()) {
            book.displayInfo();
        }

        Order order1 = new Order(1, "Phuc", javaBook, 2, javaBook.getPrice() * 2);
        Order order2 = new Order(2, "Phuc", pythonBook, 1, pythonBook.getPrice());
        Storage.getInstance().getCarts().add(order1);
        Storage.getInstance().getCarts().add(order2);

        System.out.println("\nCart:");
        for (Order order : Storage.getInstance().getCarts()) {
            System.out.println("Order id: " + order.getOrderId() + ", Customer name: " +
order.getCustomerName() +
                    ", Book: " + order.getBook().getTitle() + ", Quantity: " + order.getQuantity() +
```

```
                ", Total price: " + order.getTotalPrice());
        }

        Storage.getInstance().setOrders(Storage.getInstance().getCarts());
        Storage.getInstance().setCarts(new ArrayList<>());

        System.out.println("\nOrder list:");
        for (Order order : Storage.getInstance().getOrders()) {
            System.out.println("Order id: " + order.getOrderId() + ", Customer name: " +
order.getCustomerName() +
                ", Book: " + order.getBook().getTitle() + ", Quantity: " + order.getQuantity() +
                ", Total price: " + order.getTotalPrice());
        }

        System.out.println("\nCart list after created order:");
        for (Order order : Storage.getInstance().getCarts()) {
            System.out.println("Order id: " + order.getOrderId() + ", Customer name: " +
order.getCustomerName() +
                ", Book: " + order.getBook().getTitle() + ", Quantity: " + order.getQuantity() +
                ", Total price: " + order.getTotalPrice());
        }
    }
}
```