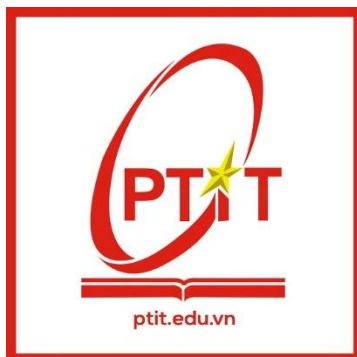


**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ TẠI TP. HCM**  
**KHOA CÔNG NGHỆ THÔNG TIN 2**



**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

<b>BẢNG PHÂN CÔNG CÔNG VIỆC .....</b>	<b>1</b>
<b>MỤC LỤC .....</b>	<b>2</b>
<b>DANH MỤC HÌNH ẢNH .....</b>	<b>3</b>
<b>CHƯƠNG 5.....</b>	<b>4</b>
BÀI 1:.....	4
BÀI 2:.....	6
BÀI 3:.....	7
BÀI 4:.....	8
<b>CHƯƠNG 6.....</b>	<b>10</b>

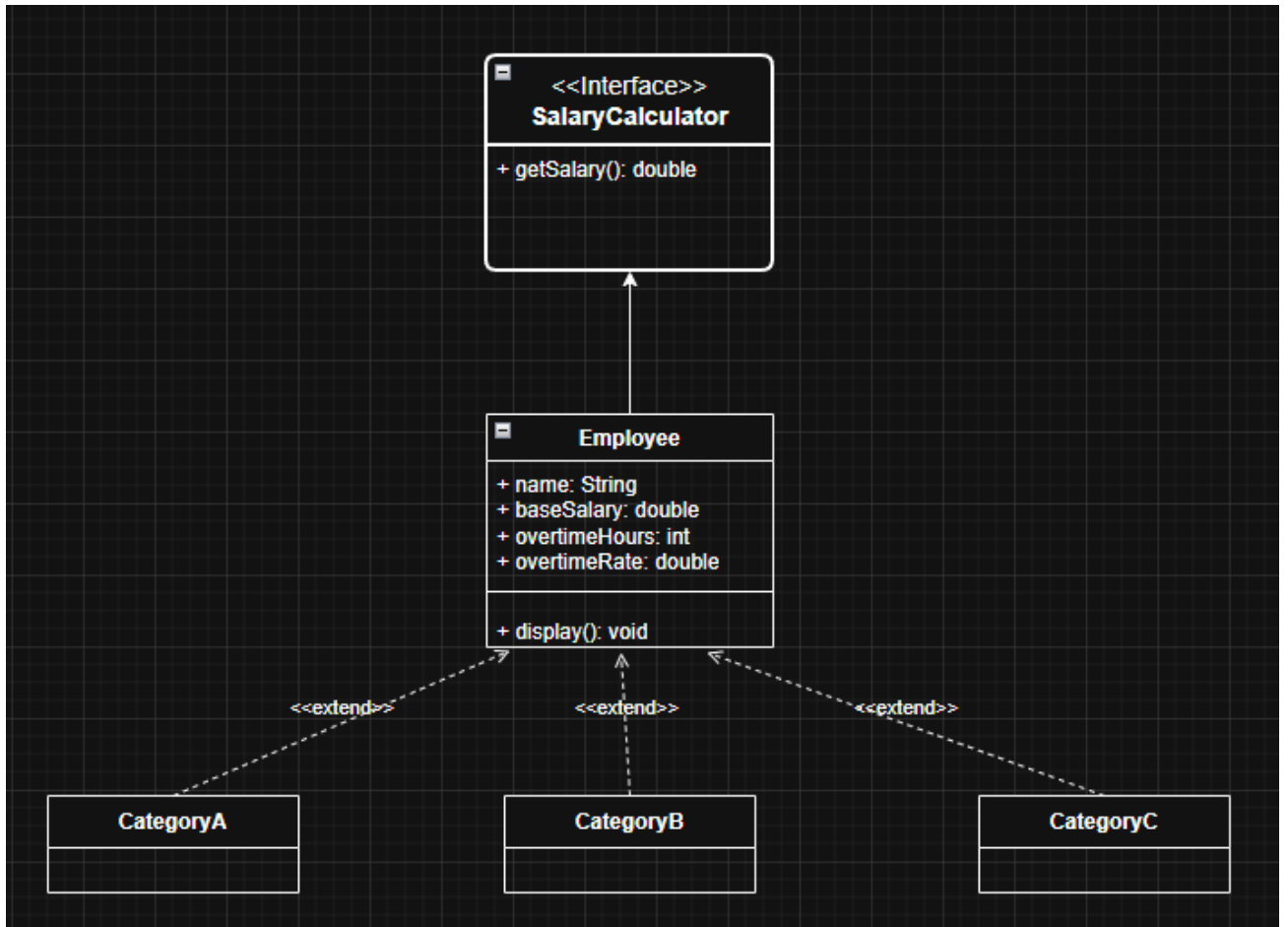
## DANH MỤC HÌNH ẢNH

Hình 1: Biểu đồ UML.....	4
Hình 2: Code Employee .....	5
Hình 3: Interface Search .....	6
Hình 4: Address Validator.....	7
Hình 5: Abstract Employee .....	8
Hình 6: Interface Employee.....	9
Hình 7: Abstract Book.....	10
Hình 8: Book Factory .....	11
Hình 9: Order.....	12
Hình 10: Storage.....	13
Hình 11: Main.....	14

## CHƯƠNG 5

### BÀI 1:

#### 1.1: Biểu đồ UML:



Hình 1: Biểu đồ UML

#### 1.2: Code:

```

1 interface SalaryCalculator {
2     double getSalary();
3 }
4
5 class Employee implements SalaryCalculator {
6     protected String name;
7     protected double baseSalary;
8     protected int overtimeHours;
9     protected double overtimeRate;
10
11     public Employee(String name, double baseSalary, int overtimeHours, double overtimeRate) {
12         this.name = name;
13         this.baseSalary = baseSalary;
14         this.overtimeHours = overtimeHours;
15         this.overtimeRate = overtimeRate;
16     }
17
18     @Override
19     public double getSalary() {
20         return baseSalary + (overtimeHours * overtimeRate);
21     }
22
23     public void display() {
24         System.out.println("Name: " + name);
25         System.out.println("Salary: " + getSalary());
26     }
27 }
28
29 class CategoryA extends Employee {
30     public CategoryA(String name, int overtimeHours) {
31         super(name, 1500, overtimeHours, 10);
32     }
33 }
34
35 class CategoryB extends Employee {
36     public CategoryB(String name, int overtimeHours) {
37         super(name, 1700, overtimeHours, 12);
38     }
39 }
40
41 class CategoryC extends Employee {
42     public CategoryC(String name, int overtimeHours) {
43         super(name, 600, overtimeHours, 5);
44     }
45 }
46
47 public class MainApp {
48     public static void main(String[] args) {
49         Employee employee1 = new CategoryA("Phuc", 10);
50         Employee employee2 = new CategoryB("Ngan", 20);
51         Employee employee3 = new CategoryC("Ruffa", 15);
52
53         System.out.println("Giam doc: ");
54         employee1.display();
55
56         System.out.println("\nQuan ly ban hang: ");
57         employee2.display();
58
59         System.out.println("\nNhan vien ban hang: ");
60         employee3.display();
61     }
62 }
63

```

Hình 2: Code Employee

## BÀI 2:

```
1 interface Search {
2     int search(Book[] books, String title);
3 }
4
5 class Book {
6     private String title;
7
8     public Book(String title) {
9         this.title = title;
10    }
11
12    public String getTitle() {
13        return title;
14    }
15 }
16
17 class LinearSearch implements Search {
18     @Override
19     public int search(Book[] books, String title) {
20         for (int i = 0; i < books.length; i++) {
21             if (books[i].getTitle().equals(title)) {
22                 return i;
23             }
24         }
25         return -1;
26     }
27 }
28
29 class BinarySearch implements Search {
30     @Override
31     public int search(Book[] books, String title) {
32         int left = 0, right = books.length - 1;
33         while (left <= right) {
34             int mid = left + (right - left) / 2;
35             int comparison = books[mid].getTitle().compareTo(title);
36             if (comparison == 0) {
37                 return mid;
38             } else if (comparison < 0) {
39                 left = mid + 1;
40             } else {
41                 right = mid - 1;
42             }
43         }
44         return -1;
45     }
46 }
47
48 public class MainApp {
49     public static void main(String[] args) {
50         Book[] books = {
51             new Book("Data Structures and Algorithms"),
52             new Book("Object Oriented Programming"),
53             new Book("Developing Java Applications"),
54             new Book("Design Patterns"),
55             new Book("Storytelling with Data"),
56         };
57
58         java.util.Arrays.sort(books, (a, b) -> a.getTitle().compareTo(b.getTitle()));
59
60         Search linearSearch = new LinearSearch();
61         int linearResult = linearSearch.search(books, "Data Structures and Algorithms");
62         System.out.println(
63             "Linear Search: "
64             + (linearResult != -1 ? "Position: " + linearResult : "Cannot find"));
65
66         Search binarySearch = new BinarySearch();
67         int binaryResult = binarySearch.search(books, "Design Patterns");
68         System.out.println(
69             "Binary Search: "
70             + (binaryResult != -1 ? "Position: " + binaryResult : "Cannot find"));
71     }
72 }
73
```

Hình 3: Interface Search

### BÀI 3:

```
1 interface AddressValidator {
2     boolean validateStreet(String street);
3
4     boolean validateCity(String city);
5
6     boolean validatePostalCode(String postalCode);
7
8     boolean validateCountry(String country);
9 }
10
11 class USAAAddress implements AddressValidator {
12     @Override
13     public boolean validateStreet(String street) {
14         return street != null && !street.isEmpty();
15     }
16
17     @Override
18     public boolean validateCity(String city) {
19         return city != null && !city.isEmpty();
20     }
21
22     @Override
23     public boolean validatePostalCode(String postalCode) {
24         return postalCode != null && postalCode.matches("\\d{5}(-\\d{4})?");
25     }
26
27     @Override
28     public boolean validateCountry(String country) {
29         return "USA".equalsIgnoreCase(country);
30     }
31 }
32
33 class VNAddress implements AddressValidator {
34     @Override
35     public boolean validateStreet(String street) {
36         return street != null && !street.isEmpty();
37     }
38
39     @Override
40     public boolean validateCity(String city) {
41         return city != null && !city.isEmpty();
42     }
43
44     @Override
45     public boolean validatePostalCode(String postalCode) {
46         return postalCode != null && postalCode.matches("\\d{6}");
47     }
48
49     @Override
50     public boolean validateCountry(String country) {
51         return "Vietnam".equalsIgnoreCase(country) || "VN".equalsIgnoreCase(country);
52     }
53 }
54
55 public class MainApp {
56     public static void main(String[] args) {
57         AddressValidator usaAddress = new USAAAddress();
58         System.out.println("USA Address Validation:");
59         System.out.println("Street valid: " + usaAddress.validateStreet("911 Main Street"));
60         System.out.println("City valid: " + usaAddress.validateCity("New York"));
61         System.out.println("Postal Code valid: " + usaAddress.validatePostalCode("12345-6789"));
62         System.out.println("Country valid: " + usaAddress.validateCountry("USA"));
63
64         AddressValidator vnAddress = new VNAddress();
65         System.out.println("\nVietnam Address Validation:");
66         System.out.println("Street valid: " + vnAddress.validateStreet("97 Man Thien"));
67         System.out.println("City valid: " + vnAddress.validateCity("Hồ Chí Minh"));
68         System.out.println("Postal Code valid: " + vnAddress.validatePostalCode("700000"));
69         System.out.println("Country valid: " + vnAddress.validateCountry("Vietnam"));
70     }
71 }
72
```

Hình 4: Address Validator



## BÀI 4:

### 4.1: Abstract:

```
1  abstract class Employee {
2      private String name;
3      private int id;
4
5      public Employee(String name, int id) {
6          this.name = name;
7          this.id = id;
8      }
9
10     public void displayData() {
11         System.out.println(
12             "Staff id: " + id + ", Name: " + name + ", Monthly salary: " + calculateMonthlyIncome() + "vnd");
13     }
14
15     public String getName() {
16         return name;
17     }
18
19     public int getId() {
20         return id;
21     }
22
23     public void setName(String name) {
24         this.name = name;
25     }
26
27     public void setId(int id) {
28         this.id = id;
29     }
30
31     public abstract double calculateMonthlyIncome();
32 }
33
34 class SalesRep extends Employee {
35     private double baseSalary;
36     private double commission;
37
38     public SalesRep(String name, int id, double baseSalary, double commission) {
39         super(name, id);
40         this.baseSalary = baseSalary;
41         this.commission = commission;
42     }
43
44     @Override
45     public double calculateMonthlyIncome() {
46         return baseSalary + commission;
47     }
48 }
49
50 class Consultant extends Employee {
51     private double hourlyRate;
52     private int hoursWorked;
53
54     public Consultant(String name, int id, double hourlyRate, int hoursWorked) {
55         super(name, id);
56         this.hourlyRate = hourlyRate;
57         this.hoursWorked = hoursWorked;
58     }
59
60     @Override
61     public double calculateMonthlyIncome() {
62         return hourlyRate * hoursWorked;
63     }
64 }
65
66 public class MainApp {
67     public static void main(String[] args) {
68         Employee salesRep = new SalesRep("Thien Phuc", 168, 3200, 600);
69         salesRep.displayData();
70
71         Employee consultant = new Consultant("Thuy Ngan", 412, 60, 180);
72         consultant.displayData();
73     }
74 }
75
```

Hình 5: Abstract Employee

## 4.2: Interface:

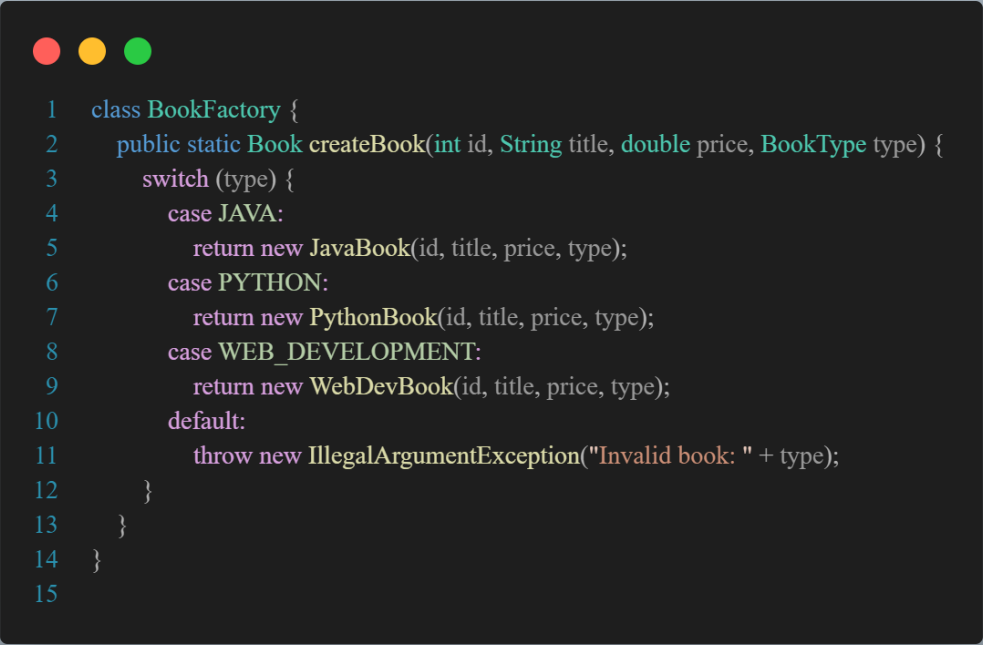
```
1 interface IEmployee {
2     void displayData();
3
4     double calculateMonthlyIncome();
5 }
6
7 class SalesRep implements IEmployee {
8     private String name;
9     private int id;
10    private double baseSalary;
11    private double commission;
12
13    public SalesRep(String name, int id, double baseSalary, double commission) {
14        this.name = name;
15        this.id = id;
16        this.baseSalary = baseSalary;
17        this.commission = commission;
18    }
19
20    @Override
21    public void displayData() {
22        System.out.println(
23            "Staff id: " + id + ", Name: " + name + ", Monthly salary: " + calculateMonthlyIncome() + "d");
24    }
25
26    public String getName() {
27        return name;
28    }
29
30    public int getId() {
31        return id;
32    }
33
34    @Override
35    public double calculateMonthlyIncome() {
36        return baseSalary + commission;
37    }
38
39    public void setName(String name) {
40        this.name = name;
41    }
42
43    public void setId(int id) {
44        this.id = id;
45    }
46 }
47
48 class Consultant implements IEmployee {
49     private String name;
50     private int id;
51     private double hourlyRate;
52     private int hoursWorked;
53
54    public Consultant(String name, int id, double hourlyRate, int hoursWorked) {
55        this.name = name;
56        this.id = id;
57        this.hourlyRate = hourlyRate;
58        this.hoursWorked = hoursWorked;
59    }
60
61    @Override
62    public void displayData() {
63        System.out.println(
64            "Staff id: " + id + ", Name: " + name + ", Monthly salary: " + calculateMonthlyIncome() + "d");
65    }
66
67    public String getName() {
68        return name;
69    }
70
71    public int getId() {
72        return id;
73    }
74
75    public void setName(String name) {
76        this.name = name;
77    }
78
79    public void setId(int id) {
80        this.id = id;
81    }
82
83    @Override
84    public double calculateMonthlyIncome() {
85        return hourlyRate * hoursWorked;
86    }
87 }
88
89 public class MainApp {
90     public static void main(String[] args) {
91         IEmployee salesRep = new SalesRep("Phuc", 168, 3000, 300);
92         salesRep.displayData();
93
94         IEmployee consultant = new Consultant("Ngan", 412, 50, 100);
95         consultant.displayData();
96     }
97 }
```

Hình 6: Interface Employee

## CHƯƠNG 6

```
1  enum BookType {
2      JAVA,
3      PYTHON,
4      WEB_DEVELOPMENT
5  }
6
7  abstract class Book {
8      protected int id;
9      protected String title;
10     protected double price;
11     protected BookType type;
12
13     public Book(int id, String title, double price, BookType type) {
14         this.id = id;
15         this.title = title;
16         this.price = price;
17         this.type = type;
18     }
19
20     public abstract void displayInfo();
21
22     public int getId() {
23         return id;
24     }
25
26     public String getTitle() {
27         return title;
28     }
29
30     public double getPrice() {
31         return price;
32     }
33
34     public BookType getType() {
35         return type;
36     }
37
38     public void setId(int id) {
39         this.id = id;
40     }
41
42     public void setTitle(String title) {
43         this.title = title;
44     }
45
46     public void setPrice(double price) {
47         this.price = price;
48     }
49
50     public void setType(BookType type) {
51         this.type = type;
52     }
53 }
54
55 class JavaBook extends Book {
56     public JavaBook(int id, String title, double price, BookType type) {
57         super(id, title, price, type);
58     }
59
60     @Override
61     public void displayInfo() {
62         System.out.println("Java book: " + getTitle() + " | Price: $" + getPrice() + " | ID: " + getId());
63     }
64 }
65
66 class PythonBook extends Book {
67     public PythonBook(int id, String title, double price, BookType type) {
68         super(id, title, price, type);
69     }
70
71     @Override
72     public void displayInfo() {
73         System.out.println("Python book: " + getTitle() + " | Price: $" + getPrice() + " | ID: " + getId());
74     }
75 }
76
77 class WebDevBook extends Book {
78     public WebDevBook(int id, String title, double price, BookType type) {
79         super(id, title, price, type);
80     }
81
82     @Override
83     public void displayInfo() {
84         System.out.println("Web Development book: " + getTitle() + " | Price: $" + getPrice() + " | ID: " + getId());
85     }
86 }
87
```

Hình 7: Abstract Book



```
1  class BookFactory {
2      public static Book createBook(int id, String title, double price, BookType type) {
3          switch (type) {
4              case JAVA:
5                  return new JavaBook(id, title, price, type);
6              case PYTHON:
7                  return new PythonBook(id, title, price, type);
8              case WEB_DEVELOPMENT:
9                  return new WebDevBook(id, title, price, type);
10             default:
11                 throw new IllegalArgumentException("Invalid book: " + type);
12             }
13         }
14     }
15 }
```


*Hình 8: Book Factory*

```

1
2 public class Order {
3     private int orderId;
4     private String customerName;
5     private int quantity;
6     private double totalPrice;
7     private Book book;
8
9     public Order(int orderId, String customerName, Book book, int quantity, double totalPrice) {
10         this.orderId = orderId;
11         this.customerName = customerName;
12         this.quantity = quantity;
13         this.totalPrice = totalPrice;
14         this.book = book;
15     }
16
17     public int getOrderId() {
18         return orderId;
19     }
20
21     public void setOrderId(int orderId) {
22         this.orderId = orderId;
23     }
24
25     public String getCustomerName() {
26         return customerName;
27     }
28
29     public void setCustomerName(String customerName) {
30         this.customerName = customerName;
31     }
32
33     public int getQuantity() {
34         return quantity;
35     }
36
37     public void setQuantity(int quantity) {
38         this.quantity = quantity;
39     }
40
41     public double getTotalPrice() {
42         return totalPrice;
43     }
44
45     public void setTotalPrice(double totalPrice) {
46         this.totalPrice = totalPrice;
47     }
48
49     public Book getBook() {
50         return book;
51     }
52
53     public void setBook(Book book) {
54         this.book = book;
55     }
56 }
57

```

Hình 9: Order



```

1
2 import java.util.ArrayList;
3 import java.util.List;
4
5 class Storage {
6     private static Storage instance;
7     private List<Book> books;
8     private List<Order> carts;
9     private List<Order> orders;
10
11     private Storage() {
12         books = new ArrayList<>();
13         orders = new ArrayList<>();
14         carts = new ArrayList<>();
15     }
16
17     public static Storage getInstance() {
18         if (instance == null) {
19             instance = new Storage();
20         }
21         return instance;
22     }
23
24     public List<Book> getBooks() {
25         return books;
26     }
27
28     public void setBooks(List<Book> books) {
29         this.books = books;
30     }
31
32     public List<Order> getOrders() {
33         return orders;
34     }
35
36     public void setOrders(List<Order> orders) {
37         this.orders = orders;
38     }
39
40     public List<Order> getCarts() {
41         return carts;
42     }
43
44     public void setCarts(List<Order> carts) {
45         this.carts = carts;
46     }
47 }

```

Hình 10: Storage

```

1
2 import java.util.ArrayList;
3
4 public class Main {
5     public static void main(String[] args) {
6         Book javaBook = BookFactory.createBook(1, "Mastering Java", 15.99, BookType.JAVA);
7         Book pythonBook = BookFactory.createBook(2, "Python for Beginners", 25.99, BookType.PYTHON);
8         Book webBook1 = BookFactory.createBook(3, "HTML & CSS Crash Course", 18.50, BookType.WEB_DEVELOPMENT);
9         Book webBook2 = BookFactory.createBook(4, "Full-Stack Web Dev", 20.50, BookType.WEB_DEVELOPMENT);
10
11         Storage.getInstance().getBooks().add(javaBook);
12         Storage.getInstance().getBooks().add(pythonBook);
13         Storage.getInstance().getBooks().add(webBook1);
14         Storage.getInstance().getBooks().add(webBook2);
15
16         System.out.println("Book list:");
17         for (Book book : Storage.getInstance().getBooks()) {
18             book.displayInfo();
19         }
20
21         Order order1 = new Order(1, "Phuc", javaBook, 2, javaBook.getPrice() * 2);
22         Order order2 = new Order(2, "Phuc", pythonBook, 1, pythonBook.getPrice());
23         Storage.getInstance().getCarts().add(order1);
24         Storage.getInstance().getCarts().add(order2);
25
26         System.out.println("\nCart:");
27         for (Order order : Storage.getInstance().getCarts()) {
28             System.out.println("Order id: " + order.getId() + ", Customer name: " + order.getCustomerName() +
29                 ", Book: " + order.getBook().getTitle() + ", Quantity: " + order.getQuantity() +
30                 ", Total price: " + order.getTotalPrice());
31         }
32
33         Storage.getInstance().setOrders(Storage.getInstance().getCarts());
34         Storage.getInstance().setCarts(new ArrayList<>());
35
36         System.out.println("\nOrder list:");
37         for (Order order : Storage.getInstance().getOrders()) {
38             System.out.println("Order id: " + order.getId() + ", Customer name: " + order.getCustomerName() +
39                 ", Book: " + order.getBook().getTitle() + ", Quantity: " + order.getQuantity() +
40                 ", Total price: " + order.getTotalPrice());
41         }
42
43         System.out.println("\nCart list after created order:");
44         for (Order order : Storage.getInstance().getCarts()) {
45             System.out.println("Order id: " + order.getId() + ", Customer name: " + order.getCustomerName() +
46                 ", Book: " + order.getBook().getTitle() + ", Quantity: " + order.getQuantity() +
47                 ", Total price: " + order.getTotalPrice());
48         }
49     }
50 }
51

```

Hình 11: Main