Practical Exam: SWD392-FAL25

Duration: 90 Minutes

Instructions

- **Internet access is NOT permitted during the exam.**
- You are allowed to use UML modeling tools such as **Visual Paradigm, Draw.io, or PlantUML in offline mode**.

Submission Requirements

- Submit your answer as a single word file or a .rar,zip archive. Name your file exactly: SWD392_PE_FAL25_[StudentID][FullName].docx or WD392_PE_FAL25[StudentID]_[FullName].rar
- (Replace [StudentID] with your student ID and [FullName] with your full name, no accents, no spaces, e.g., NguyenVanA). Example:
- SWD392_PE_FAL25_2012345_NguyenVanA.docx

Submissions with incorrect file names may not be graded.

Scenario: Your project is to design a new **Online Bookstore Management System**.

Question 1: Structural Modeling (3.0 points)

Business Scenario:

The Online Bookstore System is a web application used by **Customers** to browse and purchase books. It also integrates with an external **Shipping Service** to handle deliveries. Internally, the system models an **Inventory**, which is composed of many **Warehouses**. Each Warehouse is an integral physical part of the inventory system. Each Warehouse stores many **Books**. A Book can be either a **Paperback** or a **Hardcover** edition.

Task: Based on the scenario, you will create two separate diagrams.

1. **System Context Diagram (1.0 point):**
    - **Hint:** Focus on the "outside view" of the system.
    - Draw the **Software System Context Class Diagram**. Identify the main software system and all external entities (<<external user>>, <<external system>>) that interact with it. Show the relationships and their multiplicities.
2. **Conceptual Domain Model (2.0 points):**
    - **Hint:** Focus on the "inside view" of the system's data structure.

- Draw a **Class Diagram** for the internal entities (Inventory, Warehouse, Book, etc.). Your diagram must correctly use and distinguish between **association**, **composition**, and **generalization** relationships, including correct multiplicities.

Question 2: Dynamic Interaction Modeling (2.0 points)

Business Scenario: A customer wants to place an order. The ShoppingCartUI initiates the process. It sends a checkout request to an OrderController. The controller first validates the items in the cart with an InventoryService. If all items are in stock, the controller creates an Order object and saves it via the OrderRepository. Finally, the controller calls a PaymentService to process the customer's payment.

Task: Draw a **Sequence Diagram** for a "successful checkout" scenario. Each object in the diagram must be correctly classified with its logical stereotype based on its described role.

Question 3: State Machine Modeling (2.0 points)

Business Scenario: An **Order** in the system has a lifecycle.

- **Business rules:**
    1. An order starts in the **PendingPayment** state.
    2. Upon successful payment, it transitions to the **Processing** state.
    3. From Processing, once the warehouse has packed the items, the order moves to **ReadyToShip**.
    4. An order can only be moved to the **Shipped** state from ReadyToShip if it has a valid tracking number.
    5. After being shipped, it moves to the **Delivered** state upon confirmation from the shipping service.
    6. A customer can cancel the order only when it is in the PendingPayment or Processing state, which moves it to the **Cancelled** state.

Task: Draw a **Statechart Diagram** for the Order object that correctly models this lifecycle, including states, events, and any necessary guard conditions based on the business rules.

Question 4: Architectural Design and Trade-offs (1.0 point)

Business Scenario: You are designing the high-level structure of the server-side application. The system needs to be well-organized and maintainable. You decide to use a **Layered Architecture**.

Task:

1. **Model:** Draw a simple block diagram illustrating a standard three-layer architecture (Presentation, Business Logic, Data Access).
2. **Interpret:** Explain the responsibility of each layer. Crucially, **map the components from your previous answers into this layered structure**. For example, in which layer would the OrderController or OrderRepository reside? What is the key advantage of this separation?

Question 5: Design Pattern Identification and Analysis (2.0 points)

Business Scenario: The system needs a flexible way to add features to a book order, such as "Gift Wrapping" or "Express Shipping Insurance". These features add cost and behavior to the original order object without changing its fundamental structure. A customer can choose to add gift wrapping, then add insurance, or just have a plain order.

Examine the following pseudo-code:

```
code Java
expand_less

interface IOrder {
    double calculateCost();
}

class BasicOrder implements IOrder {
    public double calculateCost() { return 100.0; /* base price */ }
}

abstract class OrderDecorator implements IOrder {
    protected IOrder decoratedOrder;
    public OrderDecorator(IOrder order) { this.decoratedOrder = order; }
    public double calculateCost() { return decoratedOrder.calculateCost(); }
}

class GiftWrapDecorator extends OrderDecorator {
    public GiftWrapDecorator(IOrder order) { super(order); }
    public double calculateCost() {
        return super.calculateCost() + 5.0; // add gift wrap cost
    }
}
class InsuranceDecorator extends OrderDecorator {
    public InsuranceDecorator(IOrder order) { super(order); }
    public double calculateCost() {
        return super.calculateCost() + 10.0; // add insurance cost
```

```
    }
}

// How client uses it
IOrder myOrder = new BasicOrder();
myOrder = new GiftWrapDecorator(myOrder);
myOrder = new InsuranceDecorator(myOrder);
// myOrder.calculateCost() will now be 115.0
```

Tasks:

1. **Identify (0.5 points):** What is the name of the design pattern used? What is its family?
2. **Draw (1.0 point):** Draw the **Class Diagram** for this pattern based on the provided code.
3. **Explain (0.5 points):** Briefly explain the primary "intent" of this pattern and why it is useful for adding optional features dynamically.

END OF EXAMINATION