H-TechServices

Assignment<1> - Fall 2024

Course Title:	Java	Course Code:	Java-01	Credit Hours:	4(3,1)
Course Instructor:	Muhammad Hasnat Rasool	Program Name:	Java Mastery		
Due Date:	**03-12-2024**	**Maximum Marks:**		**10**	

Important Instructions / Guidelines:

The submission date is Dec 03, 2024. Submit your assignment in the form of a report. It should contain a problem statement, solution (code), and output. Your pdf/docx file name should be your name.

Upload your file on G**ithub .**

Ensure your program runs without errors and follows the structure .

Learning Objectives: Java programming.

Question 2: Building an Online Store System (Composition, Aggregation, Inheritance, Polymorphism, Interfaces)

Problem Statement:

Design an online store system where the `Product` can be either a `PhysicalProduct` or a `DigitalProduct`, each with its own pricing and delivery methods. The system should allow customers to browse products, add them to their cart, and checkout.

Requirements:

1. **Composition and Aggregation**:
    - The `ShoppingCart` class should aggregate multiple `Product` objects. It should have methods for adding/removing products and calculating the total cost of items in the cart.
    - The `Store` class should be able to list all available products and display the types of products (i.e., physical vs. digital). The products are stored in a collection.
2. **Polymorphism**:

- Define a `Product` interface with methods `getPrice()` and `getDescription()`. Both `PhysicalProduct` and `DigitalProduct` should implement this interface, with different behaviors.
- Each type of `Product` should override the `getPrice()` method, where `PhysicalProduct` adds shipping costs, and `DigitalProduct` adds a licensing fee.
- A `Cart` object should accept any type of `Product` and correctly calculate the total price, demonstrating **polymorphism**.

3. **Inheritance**:
    - Create a superclass `Product` with common attributes (like `name`, `price`, and `id`) and subclass it into `PhysicalProduct` and `DigitalProduct`.
    - The `PhysicalProduct` class should include shipping-related methods, while the `DigitalProduct` class should handle digital delivery (e.g., email delivery or download links).
4. **Interfaces**:
    - Use an interface `Purchasable` with methods `addToCart()` and `checkout()`. Both `PhysicalProduct` and `DigitalProduct` should implement `Purchasable`.
    - The `ShoppingCart` class should have a method `checkout()` that handles the checkout process. This process should vary depending on whether the items are physical or digital.

What is expected:

- **Composition and Aggregation**: The `ShoppingCart` aggregates `Product` objects. The `Store` holds a collection of products, and each product can be of different types (either `PhysicalProduct` or `DigitalProduct`).
- **Polymorphism**: The checkout system should handle different product types polymorphically, where each product's `getPrice()` and `getDescription()` methods behave differently depending on the product type.
- **Inheritance**: The `PhysicalProduct` and `DigitalProduct` classes inherit from `Product` and add their own specific attributes and methods.
- **Interfaces**: `Purchasable` is an interface implemented by products, allowing polymorphic behavior for adding products to the cart and completing the purchase process.
-

Output:

Store: My Online Store

Products Available:

1. Product: "Laptop", Price: 1000.00, Type: Physical

2. Product: "Ebook: Java Programming", Price: 30.00, Type: Digital

3. Product: "Headphones", Price: 50.00, Type: Physical

Shopping Cart:

Items in your cart:

1. Laptop (Physical)

2. Ebook: Java Programming (Digital)

Total: 1080.00 (Shipping: 50.00, Licensing Fee: 30.00)

Checkout:

Thank you for purchasing the following items:

- Laptop (Physical) with shipping fee: 50.00

- Ebook: Java Programming (Digital) with licensing fee: 30.00

Total: 1080.00

Marks Breakdown (Total: 10)

1. **Composition and Aggregation (2 marks)**:
    - **1 mark** for correctly implementing the **ShoppingCart** class that aggregates multiple **Product** objects and provides methods to add/remove products and calculate the total cost.
    - **1 mark** for creating a **Store** class that holds and manages a collection of products, allowing customers to browse available **PhysicalProduct** and **DigitalProduct** items.
2. **Polymorphism (2 marks)**:
    - **1 mark** for defining the **Product** interface with methods `getPrice()` and `getDescription()`, and implementing it in **PhysicalProduct** and **DigitalProduct** with different behaviors for each product type.
    - **1 mark** for ensuring the **ShoppingCart** correctly calculates the total price using polymorphism, handling both **PhysicalProduct** and **DigitalProduct** in the same way by invoking the `getPrice()` method.
3. **Inheritance (2 marks)**:
    - **1 mark** for creating a base **Product** class with common attributes like `name`, `price`, and `id`, and subclassing it into **PhysicalProduct** and **DigitalProduct**.
    - **1 mark** for ensuring that **PhysicalProduct** and **DigitalProduct** override specific methods, such as `getPrice()` and `getDescription()`, and include product-specific attributes like shipping costs for **PhysicalProduct** and licensing fees for **DigitalProduct**.
4. **Interfaces (2 marks)**:
    - **1 mark** for defining and implementing the **Purchasable** interface, which includes methods `addToCart()` and `checkout()`. Both **PhysicalProduct** and **DigitalProduct** should implement this interface.
    - **1 mark** for ensuring that the **ShoppingCart** class uses the `checkout()` method and handles different behaviors depending on whether the products in the cart are physical or digital.
5. **Output/Functionality (2 marks)**:
    - **1 mark** for ensuring the system can list products of different types (Physical and Digital), and display them correctly.

- **1 mark** for correctly simulating a shopping cart system, including adding products to the cart, calculating the total, and handling the checkout process, with correct outputs like total price calculation and delivery methods.

Sample Output (for reference):

Store: My Online Store

Products Available:

1. Product: "Laptop", Price: 1000.00, Type: Physical

2. Product: "Ebook: Java Programming", Price: 30.00, Type: Digital

3. Product: "Headphones", Price: 50.00, Type: Physical

Shopping Cart:

Items in your cart:

1. Laptop (Physical)

2. Ebook: Java Programming (Digital)

Total: 1080.00 (Shipping: 50.00, Licensing Fee: 30.00)

Checkout:

Thank you for purchasing the following items:

- Laptop (Physical) with shipping fee: 50.00

- Ebook: Java Programming (Digital) with licensing fee: 30.00

Total: 1080.00

Summary of Marks Allocation:

Task Aspect	Marks
Composition and Aggregation	2
Polymorphism	2
Inheritance	2
Interfaces	2
Output/Functionality	2
Total	**10**