**Improvement Report - Event Planning**

1. Class Structure:

- Base and derived classes were created to represent different types of events.

- Classes: `Event`, `Lecture`, `Reception`, `OutdoorGathering`.

- An additional class `Address` was created to encapsulate address details.

2. Encapsulation:

- All attributes were defined as private to ensure the principle of encapsulation.

- Public methods were created to access the necessary attributes.

3. Added Methods:

- Methods to return standard details, full details, and a short description of each event.

- Specific methods were overridden in the derived classes to provide additional information as needed.

4. Organization and Readability:

- Detailed comments were added to describe the functionality of each class and method.

- Code was divided into regions for better organization and clarity.

5. Demonstration:

- Examples of each type of event were created in the `Main` method.

- Displaying the event details using the created methods.

6. Additional Information:

- The code was structured to easily allow adding new types of events in the future.

Example Usage:

Address address1 = new Address("123 Park Ave", "New York", "NY", "USA"); Event lecture = new Lecture("Tech Talk", "An in-depth look at modern technology.", new DateTime(2024, 6, 15), "10:00 AM", address1, "Dr. Smith", 100);

Address address2 = new Address("456 Ocean Blvd", "Los Angeles", "CA", "USA"); Event reception = new Reception("Networking Event", "An opportunity to connect with professionals.", new DateTime(2024, 6, 20), "6:00 PM", address2, "rsvp@example.com");

Address address3 = new Address("789 Mountain Rd", "Denver", "CO", "USA"); Event outdoorGathering = new OutdoorGathering("Hiking Adventure", "Join us for a day of hiking and fun.", new DateTime(2024, 6, 25), "9:00 AM", address3, "Sunny and warm");

Event[] events = { lecture, reception, outdoorGathering };

foreach (var eventObj in events) { Console.WriteLine(eventObj.GetStandardDetails()); Console.WriteLine(); Console.WriteLine(eventObj.GetFullDetails()); Console.WriteLine(); Console.WriteLine(eventObj.GetShortDescription()); Console.WriteLine(new string('-', 50)); }

*(hope this is not cofused for you)*

**Improvement Report - Customer and Order Management**

1. Class Structure:

- Classes were created to represent addresses, customers, products, and orders.

- Classes: `Address`, `Customer`, `Product`, `Order`.

2. Encapsulation:

- All attributes were defined as private to ensure the principle of encapsulation.

- Public methods were created to access the necessary attributes.

3. Added Methods:

- Methods to calculate total cost, generate packing and shipping labels.

- Methods to check if the address is in the USA for appropriate shipping costs.

4. Organization and Readability:

- Detailed comments were added to describe the functionality of each class and method.

- Code was divided into regions for better organization and clarity.

5. Demonstration:

- Examples of customers and orders were created in the `Main` method.

- Displaying the packing and shipping labels and total cost for each order.

6. Additional Information:

- The code was structured to easily allow adding new products and managing different orders in the future.

Example Usage:

Address address1 = new Address("123 Main St", "New York", "NY", "USA"); Customer customer1 = new Customer("John Doe", address1);

Address address2 = new Address("456 Elm St", "Toronto", "ON", "Canada"); Customer customer2 = new Customer("Jane Smith", address2);

Product product1 = new Product("Laptop", "A123", 999.99m, 1); Product product2 = new Product("Mouse", "B456", 25.50m, 2); Product product3 = new Product("Keyboard", "C789", 45.75m, 1);

Order order1 = new Order(customer1); order1.AddProduct(product1); order1.AddProduct(product2);

Order order2 = new Order(customer2); order2.AddProduct(product2); order2.AddProduct(product3);

Console.WriteLine(order1.GetPackingLabel()); Console.WriteLine(order1.GetShippingLabel()); Console.WriteLine($"Total Cost: {order1.GetTotalCost()

}\n");

Console.WriteLine(order2.GetPackingLabel()); Console.WriteLine(order2.GetShippingLabel()); Console.WriteLine($"Total Cost: {order2.GetTotalCost()

}");

*(hope this is not cofused for you)*

**Improvement Report - Video and Comment Management**

1. Class Structure:

- Classes were created to represent videos and comments.

- Classes: `Video`, `Comment`.

2. Encapsulation:

- All attributes were defined as private to ensure the principle of encapsulation.

- Public methods were created to access the necessary attributes.

3. Added Methods:

- Methods to add comments to videos and retrieve comment counts and details.

- Methods to get video titles, authors, lengths, and comments.

4. Organization and Readability:

- Detailed comments were added to describe the functionality of each class and method.

- Code was divided into regions for better organization and clarity.

5. Demonstration:

- Examples of videos with comments were created in the `Main` method.

- Displaying the video details and associated comments.

6. Additional Information:

- The code was structured to easily allow adding new videos and managing comments in the future.

Example Usage:

List<Video> videos = new List<Video>();

Video video1 = new Video("How to Cook Pasta", "Chef John", 600); video1.AddComment(new Comment("Alice", "Great recipe!")); video1.AddComment(new Comment("Bob", "Tried it and loved it!")); video1.AddComment(new Comment("Charlie", "Will try this weekend."));

Video video2 = new Video("Yoga for Beginners", "YogaWithAdriene", 1800); video2.AddComment(new Comment("Dave", "Feeling relaxed already.")); video2.AddComment(new Comment("Eve", "Perfect for starting my day.")); video2.AddComment(new Comment("Frank", "Loved the session."));

Video video3 = new Video("Travel Vlog: Japan", "TravelWithMe", 1200); video3.AddComment(new Comment("Grace", "Amazing views!")); video3.AddComment(new Comment("Heidi", "Can't wait to visit Japan.")); video3.AddComment(new Comment("Ivan", "Great tips for travelers."));

videos.Add(video1); videos.Add(video2); videos.Add(video3);

foreach (Video video in videos) { Console.WriteLine($"Title: {video.GetTitle()}"); Console.WriteLine($"Author: {video.GetAuthor()}"); Console.WriteLine($"Length: {video.GetLength()} seconds"); Console.WriteLine($"Number of Comments: {video.GetCommentCount()}");

*(hope this is not cofused for you)*

**Improvement Report - Activity Tracking**

1. Class Structure:

- Base and derived classes were created to represent different types of activities.

- Classes: `Activity`, `Running`, `Cycling`, `Swimming`.

2. Encapsulation:

- All attributes were defined as protected to ensure the principle of encapsulation while allowing access in derived classes.

- Public methods were created to access the necessary attributes and calculations.

3. Added Methods:

- Methods to calculate distance, speed, and pace for each activity type.

- Methods to provide a summary of each activity.

4. Organization and Readability:

- Detailed comments were added to describe the functionality of each class and method.

- Code was divided into regions for better organization and clarity.

5. Demonstration:

- Examples of each type of activity were created in the `Main` method.

- Displaying the activity summaries using the created methods.

6. Additional Information:

- The code was structured to easily allow adding new types of activities in the future.

Example Usage:

List<Activity> activities = new List<Activity> { new Running(new DateTime(2024, 5, 1), 30, 3.0), new Cycling(new DateTime(2024, 5, 2), 45, 15.0), new Swimming(new DateTime(2024, 5, 3), 60, 20) };

foreach (var activity in activities) { Console.WriteLine(activity.GetSummary()); }

*(hope this is not cofused for you)*