****

**Online Ordering Program Call Class Diagram**

* **Program CLASS (Main)**
  + - **Order constructor**
    - **Order.AddProduct()**
    - **Order.GetPackingLabel()**
    - **Order.GetShippingLabel()**
    - **Order.CalculateTotalCost()**
* **Address**
  + Attributes:
    - streetAddress: string
    - city: string
    - stateOrProvince: string
    - country: string
  + Methods:
    - IsInUSA(): bool
    - GetFullAddress(): string
* **Customer**
  + Attributes:
    - name: string
    - address: Address
  + Methods:
    - LivesInUSA(): bool
    - GetName(): string
    - GetAddress(): Address
* **Product**
  + Attributes:
    - name: string
    - productId: string
    - price: decimal
    - quantity: int
  + Methods:
    - TotalCost(): decimal
    - GetName(): string
    - GetProductId(): string
* **Order**
  + Attributes:
    - products: List<Product>
    - customer: Customer
  + Methods:
    - AddProduct(product: Product): void
    - CalculateTotalCost(): decimal
    - GetPackingLabel(): string
    - GetShippingLabel(): string

Each class maintains encapsulation by keeping its attributes private and only exposing necessary functionalities through methods. Relationships between classes, like Order containing Product and Customer, and Customer containing Address, are reflected in this structure.

1. Order is the central coordinator class

2. Customer acts as a bridge between Order and Address

3. Program only interacts directly with Order

4. All field accesses go through methods (proper encapsulation)

5. USA check follows the chain: Program → Order → Product→ Customer → Address

This shows both the static structure and the dynamic method call relationships between classes. Each arrow represents a method call from one class to another.



YouTube Video Call Class Diagram

**Class Diagram**

* **Video**
  + **Attributes**:
    - title: string
    - author: string
    - lengthInSeconds: int
    - comments: List<Comment>
  + **Methods**:
    - GetNumberOfComments(): int
    - AddComment(comment: Comment): void
    - GetTitle(): string
    - GetAuthor(): string
    - GetLength(): int
    - GetComments(): List<Comment>
* **Comment**
  + **Attributes**:
    - commenterName: string
    - commentText: string
  + **Methods**:
    - GetCommenterName(): string
    - GetCommentText(): string
* Aggregation:
  + A Video contains a list of Comment objects. Each Comment is related to a Video, but they can exist independently.
* Each Video manages its own comments and provides access to them.

This diagram demonstrates abstraction by focusing on the essential details and responsibilities for the Video and Comment classes, while hiding internal implementations.