**Foundation Programs Design Document**

**Foundation Program #1: Abstraction with YouTube Videos**

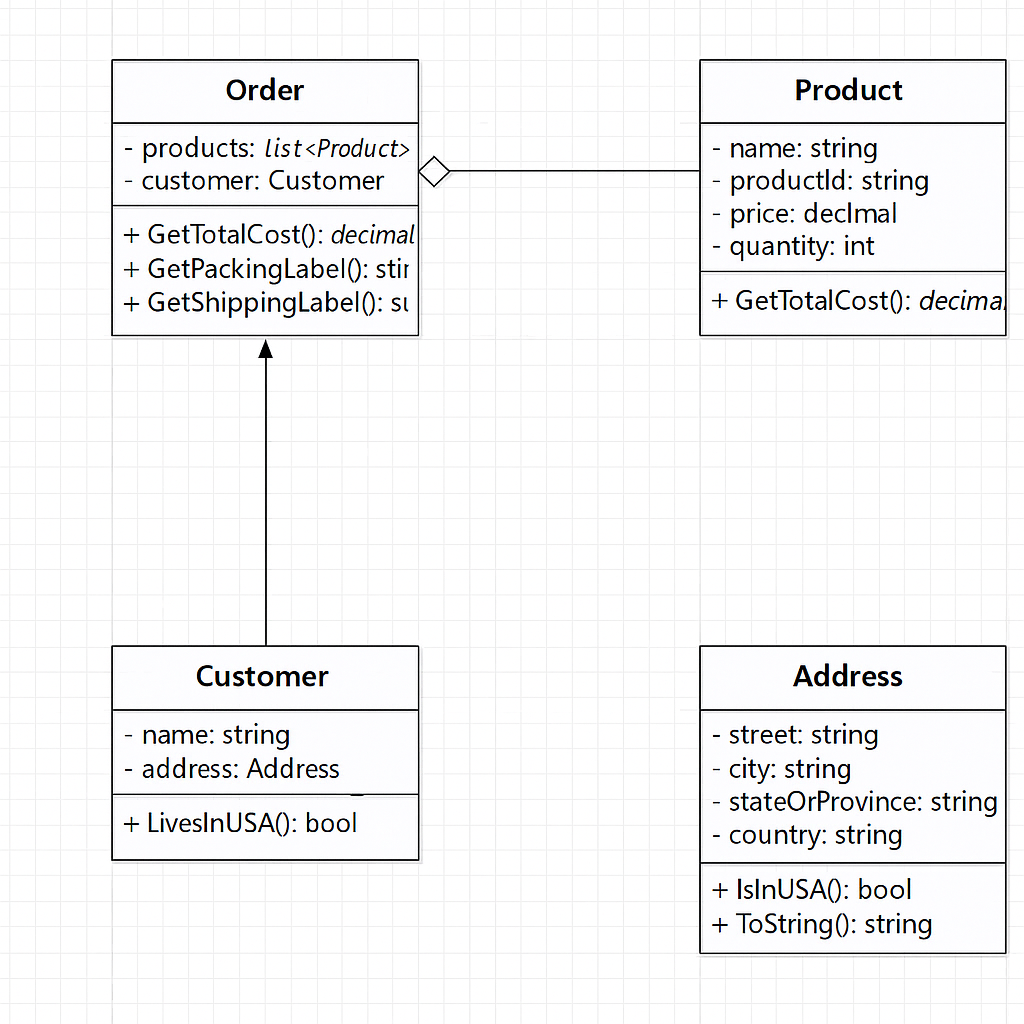
**What does the program do?**

This program simulates the management of YouTube videos, allowing creation, viewing, and management of video data including titles, authors, durations, views, likes, and comments.

**Candidates for Classes and Responsibilities**

|  |  |  |
| --- | --- | --- |
| Class Name | Responsibilities (Methods) | Attributes (Member Variables) |
| Video | Store video info; display video details | title, author, lengthInSeconds, views, likes, dislikes |
| Comment | Store comment details; display comment | commenterName, commentText |
| VideoManager | Manage a list of videos; add video; display all videos | videos: List |

**Class diagram**



**Behaviors and Methods**

* Video

displayDetails()

* Comment

displayComment()

* VideoManager

addVideo(video: Video)

showAllVideos()

**Class Diagram (Text Representation)**

**Video**

* title: string
* author: string
* lengthInSeconds: int
* views: int
* likes: int
* dislikes: int
* displayDetails(): void

**Comment**

* commenterName: string
* commentText: string
* displayComment(): void

**VideoManager**

* videos: List
* addVideo(video: Video): void
* showAllVideos(): void

**Program Flow Description**

1. Program initializes and creates several Video objects.
2. Each Video object can have multiple Comment objects.
3. VideoManager stores and manages all Video objects.
4. User can view video details and associated comments.
5. Methods handle data display and management, abstracting internal details.

**Foundation Program #2: Encapsulation with Online Ordering**

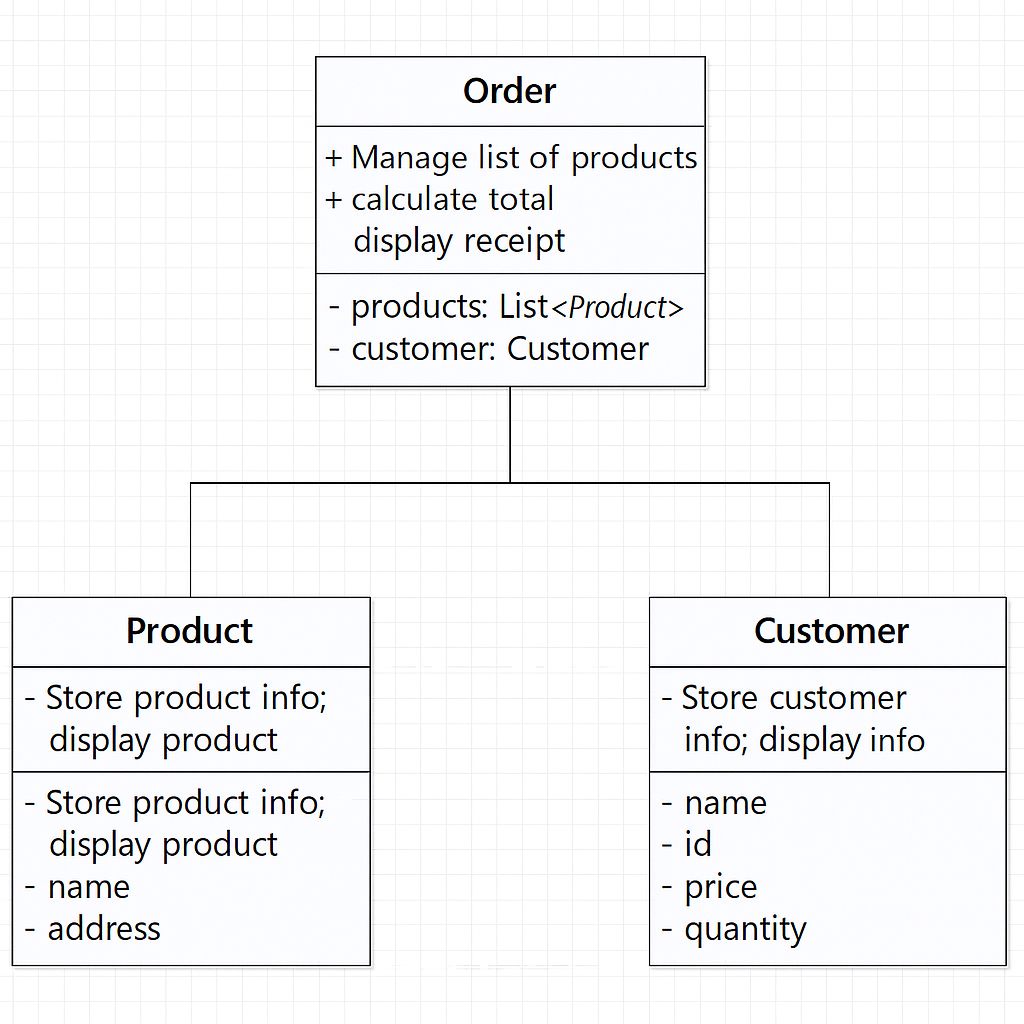
**What does the program do?**

This program simulates an online ordering system, where users can add products to an order, calculate the total cost, and view receipts. The program uses encapsulation to hide internal data.

**Candidates for Classes and Responsibilities**

|  |  |  |
| --- | --- | --- |
| Class Name | Responsibilities (Methods) | Attributes (Member Variables) |
| Product | Store product info; display product | name, id, price, quantity |
| Customer | Store customer info; display info | name, address |
| Order | Manage list of products; calculate total; display receipt | products: List, customer: Customer |

**Class diagram**



**Behaviors and Methods**

* Product

getPrice()

displayProduct()

* Customer

displayCustomerInfo()

* Order

addProduct(product: Product)

calculateTotal()

displayReceipt()

**Class Diagram (Text Representation)**

**Product**

* name: string
* id: string
* price: float
* quantity: int
* getPrice(): float
* displayProduct(): void

**Customer**

* name: string
* address: string
* displayCustomerInfo(): void

**Order**

* products: List
* customer: Customer
* addProduct(product: Product): void
* calculateTotal(): float
* displayReceipt(): void

**Program Flow Description**

1. Program begins by collecting customer information.
2. User selects products to add to their order.
3. Each product is encapsulated within the Order class.
4. Order class calculates total cost internally.
5. Program prints receipt with customer and order details.

**Summery**

The designs presented provide a foundation for the development of both programs. Additional classes and methods can be added as needed during implementation.