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

**Program Description**

This program allows users to manage a list of YouTube videos. Users can add, remove, and view videos. The program demonstrates abstraction by hiding the implementation details of video management while exposing a simple interface for users to interact with.

**Candidate Classes**

1. **Video** – Represents a single YouTube video.
2. **VideoLibrary** – Manages a collection of videos.
3. **Program** – Handles user input/output and interacts with VideoLibrary.

**Responsibilities of Each Class**

* **Video**
  + Store video title, URL, and duration.
  + Provide methods to display video info.
* **VideoLibrary**
  + Add and remove videos.
  + List all videos.
  + Search videos by title.
* **Program**
  + Display menu to the user.
  + Accept user input.
  + Call methods on VideoLibrary.

**Class Attributes and Methods**

| **Class** | **Attributes** | **Methods** |
| --- | --- | --- |
| Video | \_title, \_url, \_duration | GetDisplayText() |
| VideoLibrary | \_videos (list of Video) | AddVideo(), RemoveVideo(), ListVideos(), SearchVideo() |
| Program | none | Run() |

**Class Diagram**

Program

|

v

VideoLibrary

|

v

Video

**Program Flow**

1. Program starts and displays menu.
2. User selects action (add, remove, view).
3. Program calls VideoLibrary methods accordingly.
4. VideoLibrary manipulates Video objects.
5. Output displayed to user.

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

**Program Description**

This program simulates an online ordering system. It demonstrates encapsulation by restricting direct access to order details and providing controlled access through methods.

**Candidate Classes**

1. **Product** – Represents a product available for purchase.
2. **OrderItem** – Represents a product added to an order with a quantity.
3. **Order** – Contains a list of OrderItem objects.
4. **Customer** – Stores customer information.
5. **Program** – Handles user interaction.

**Responsibilities of Each Class**

* **Product**
  + Store product name, price, and stock quantity.
  + Provide methods to get product info.
* **OrderItem**
  + Store a product and quantity.
  + Calculate total price for this item.
* **Order**
  + Add/remove OrderItems.
  + Calculate total order cost.
  + Display order details.
* **Customer**
  + Store name, address, and contact info.
  + Retrieve customer info.
* **Program**
  + Display menu.
  + Take user input.
  + Call methods on Order and Customer.

**Class Attributes and Methods**

| **Class** | **Attributes** | **Methods** |
| --- | --- | --- |
| Product | \_name, \_price, \_stock | GetInfo(), UpdateStock() |
| OrderItem | \_product, \_quantity | GetTotalPrice() |
| Order | \_items (list of OrderItem) | AddItem(), RemoveItem(), GetTotalCost(), DisplayOrder() |
| Customer | \_name, \_address, \_contact | GetInfo() |
| Program | none | Run() |

**Class Diagram**

Program

|

v

Order ----> OrderItem ----> Product

|

v

Customer

**Program Flow**

1. Program starts and displays menu.
2. Customer selects products and quantities.
3. Order adds items and calculates totals.
4. Customer information is collected and stored.
5. Order summary displayed to the user.