*Program Design Document – Week 4*

C++ Language

Written by: Sergio Luilly Cabrera Dorado

This week's project focuses on applying the principles of abstraction and encapsulation in object-oriented programming using C++. These principles help to simplify complex systems, hide unnecessary details, and protect the internal state of objects.

**Program 1:** "YouTube Video Comments Simulation"

* + **Purpose**

Simulate the basic operation of YouTube by creating classes that represent videos and comments, applying **abstraction** to model real-world concepts without including unnecessary technical details (such as internet connection or databases).

* + **Identified classes**
    - **Class *Comment***

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Guy** | **Description** |
| Author | String | Name of the user who wrote the comment |
| Text | String | Content of the written comment, which must not violate the Community Guidelines |

**Methods**

* Comment(string author, string text) → Constructor.
* string GetAuthor() → Returns the author's name.
* string GetText() → Returns the comment text.
  + **Video Class**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Guy** | **Description** |
| Title | String | Video Title |
| Author | String | Name of the creator or person who uploaded the video to the platform |
| Duration | Int | Video Length in Seconds |
| Comments | Vector<comment> | List of associated comments |

**Methods**

* + Video(string title, string author, int duration) → Constructor.
  + void AddComment(Comment c) → Add a comment.
  + int GetCommentCount() → Returns how many comments there are.
  + void DisplayVideoInfo() → Displays title, author, duration, and comments.
  + Class Diagram (Summarized)

|  |  |  |
| --- | --- | --- |
| **Video** |  | **Comment** |
| * title: string * author: string * Duration: Int * comments: vector<Comment> | can contain several | * author: string * text: string |
| * AddComment(Comment) * GetCommentCount():int * DisplayVideoInfo():void |  | * GetAuthor():string * GetText():string |

* + Program Workflow
    1. The main program creates several *Video objects*.
    2. Multiple Comment objects are added to each video *.*
    3. Videos are saved in a vector<Video>.
    4. The vector is traversed showing the information of each video and its comments.

Final Notes

* + Video, can have many comments (composition) from different authors.
  + If a video is deleted, its comments are deleted as well.
  + Each class represents a concept, in this case the Video and *Comment classes*
  + The user of the code does not need to know how the comments made are stored, he only makes use of the public methods to publish his comments on the video he has seen, since he must be in a specific video to be able to make a comment.

**Program 2:** "Online Ordering System"

* + **Purpose**

Simulate the process of an online order by applying **encapsulation**, keeping data private within each class, and controlling access using public methods.

* + **Identified classes**
    - **Address Class**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Guy** | **Description** |
| Street | String | Address, because some international addresses are not just one Street, but a combination of two or more Streets |
| City | String | City |
| State | String | State, Province, Department, etc... |
| Country | String | Country |

**Methods**

* Address(string street, string city, string state, string country)
* bool IsInUSA()
* string GetFullAddress()
  + - **Costumer Class**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Guy** | **Description** |
| Costumer | String | Client Name |
| Address | Address | Address |

**Methods**

* Customer(string name, Address address)
* bool LivesInUSA()
* string GetName()
* Address GetAddress()
  + - **Product *Class***

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Guy** | **Description** |
| Name | String | Product Name |
| Product ID | String | Product Identification |
| Price | Double | Price per unit |
| Quantify | Int | Desired quantity |

**Methods**

* + Product(string name, string id, double price, int qty)
  + double GetTotalCost()
    - **Order Class**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Guy** | **Description** |
| Costumer | Costumer | Client Partner / Client Name |
| Products | Vector <Product> | List of available products |

**Methods**

* Order(Customer customer)
* void AddProduct(Product p)
* double GetTotalPrice()
* string GetPackingLabel()
* string GetShippingLabel()
  + Class Diagram (Summarized)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Order |  | Product | Costumer |  | Address |
| customer: Customer  products: vector<Product> |  | name: string  productId: string  price: double  Quantity: Int | name: string  address: Address |  | Street: Strin  City: String  state: string  country: string |
| AddProduct(Product)  GetTotalPrice():d ouble  GetPackingLabel():string  GetShippingLabel():string |  | GetTotalCost():d ouble | LivesInUSA():bool  GetName():string  GetAddress():Address |  | IsInUSA():bool  GetFullAddress() |

* + Program Workflow

1. Address and Customer objects are created.
2. Product objects are created with their prices and quantities.
3. Order orders are created that contain those products and the customer.
4. The program shows:
   1. Packaging label (products + ID).
   2. Shipping label (name and address).
   3. Total cost (adding products + shipping).

Final Notes

* Order contains several Product.
* Order has a Customer.
* Customer has an Address.
* All attributes are private.
* Methods control access to data.
* Each class is responsible for its own data and behaviors.