**Christopher T. Bahinting**

**W04 Team Activity: Foundation Programs Design**

Abstraction is the process of turning complex ideas into simple ones. It is removing characteristics from something so that only the essential ones remain. As programmers, we create and use abstractions all the time.

The following below are the examples of how the program can works.

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

**Class: Comment**

Purpose: Represents a comment left on a YouTube video.

Attributes:

* + commenterName: stores who made the comment.
  + commentText: stores the content of the comment.

Constructor: Initializes a comment with a name and text.

Why important: Helps us keep comment data organized.

**Class: Video**

Purpose: Represents a Youtube video and keeps track of its comments.

Attributes:

* + title: title of the video.
  + author: creator of the video.
  + lengthInSeconds: duration of the video.
  + \_comments: a private list that stores all comments related to this video.

Methods:

* + AddComment(Comment): adds a new comment to the video.
  + GetCommentCount(): returns how many comments the video has.

Why important: This class abstracts the video and its comments, hiding the internal comment list but providing ways to interact with it.

Encapsulation is the act of enclosing something, as if it were in a capsule. It means thinking carefully about the behaviors your classes need and then hiding the details of how they perform those behaviors, even making it so other code cannot see or manipulate these details.

The following below are the examples of how the program can works.

1. **Foundation Program : Encapsulation with Online Ordering**

**Class: Product**

Purpose: Represents a product being sold.

Attributes:

* name, productId, price, quantity.

Methods:

* + GetTotalCost(): calculates total price based on quantity.

Encapsulation note: All variables are private; accessed through methods/getters to protect data.

**Class: Address**

Purpose: Represents a customer’s address.

Attributes:

* + street, city, state, country.

Methods:

* + IsInUSA(): returns true if country is USA.
  + GetFullAddress(): returns full address string.

Why: Helps in deciding shipping cost based on location.

**Class: Customer**

Purpose: Stores customer information.

Attributes:

* + name
  + address (which is an Address object).

Methods:

* + IsInUSA(): calls address.IsInUSA() to determine customer location.

**Class: Order**

Purpose: Represents an order made by a customer.

Attributes:

* + List of Product objects.
  + Customer object.

Methods:

* GetTotalCost(): sums all product costs + shipping.
* GetPackingLabel(): lists product names and IDs.
* GetShippingLabel(): shows customer name and address.

Below are the examples of **CLASS DIAGRAM**

