***Monitors Youtube Videos:***

*Class: Video*

Responsibilities:

track the title, author, and length (in seconds) of the video and a List of comments (class)

# Additionally, it also tracks a list of Videos to work as expected.

Behaviors:

Create a constructor to track the title author, length and the list of Comments.

Display the number of comments that the video had.

Display video info like (title, author and length (in seconds) to user

Display the comments for the video

*Class: Comment*

Responsibilities:

Track the name of the person who made a comment and the text of the comment.

Behaviors:

Create a Constructor to set the values of person name and the comment.

display the name of the person and the text of the comment.

**Aditionally:**

*Class: User Generator*

Responsibilities:

Track a list of comments and a list of Users.

Behaviors:

Get a random comment from the list.

Get a random user from the list.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

With the information above we can create the following diagram:

**UserGenerator**

\_randomComments : List<string>

\_randomUsers : List<string>

SetRandomUser() : string

SetRandomComment() : string

**Comment**

\_personName : string

\_comment : string

Comment(string *personName*, string *comment*)

DisplayComment() : void

SetRandomUser() : string

SetRandomComment() : string

**Video**

\_title : string

\_author : string

\_length : int

\_comments : List <Comment>

\_videos : List<Video>

Video (string *title*, string *author*, int *length*)

AddComments (string *user*, string *comment*) : void

DisplayCommentQuantity () : void

DisplayVideoInfo() : void

DisplayAll() : void

***Product Ordering System***

*Class: Product*

Responsibilities:

Track the name, product id, price and quantity of a product

Behaviors:

Create a constructor for each one of the properties (name, id, quantity, product)

Compute the total cost of the product (quantity \* price)

Set the product name

Set the product id

*Class: Customer*

Responsibilities:

Track the name of a customer and an address (class)

Behaviors:

Create a constructor to store name of the customer and the address Class.

Get the confirmation if a customer lives in United States.

Get the message for a Shipping Label

*Class: Address*

Responsibilities:

Tracks the street address, city, state and country of a customer

Behaviors:

Create a constructor for each one of the properties

Get the confirmation if a customer lives in USA.

Get the complete address (address – city – state/province – country)

Get the actual address of the customer

*Class: Order*

Responsibilities:

Tracks the information of a Customer (class) as well as a shipping cost ($5 if the customer lives in US and $35 if lives outside) contain a list of Products (class)

Behaviors:

Compute the total cost of the order (Each product and quantity + the shipping cost)

Get a message for the packing label (name and product id)

Get a message for the shipping label (name and address of the customer)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

With the information above we can create the following diagram:

**Customer**

\_name : string

\_Address : Address

Customer (string *name*, Address *address*)

LivesInUSA () : bool

DisplayShippingLabel () : void

**Address**

\_address : string

\_city : string

\_state : string

\_country : string

Address (string *address*, string *city*, string *state*, string *country*)

IsInUSA () : void

GetCompleteAddress : string

GetStreetAddress () : string

**Products**

\_name : string

\_productId : string

\_price : decimal

\_quantity : int

Product (string *name*, string *productid*, int *price*, int *quantity*)

ComputeProductCost () : decimal

DisplayPackingLabel () : void

**Order**

\_customer : Customer

\_products : List<Product>

\_shippingCost : decimal

Order (Customer *customer*)

AddProduct (Product *product*) : void

ComputeTotalCost () : decimal

DisplayPackingLabel() : void