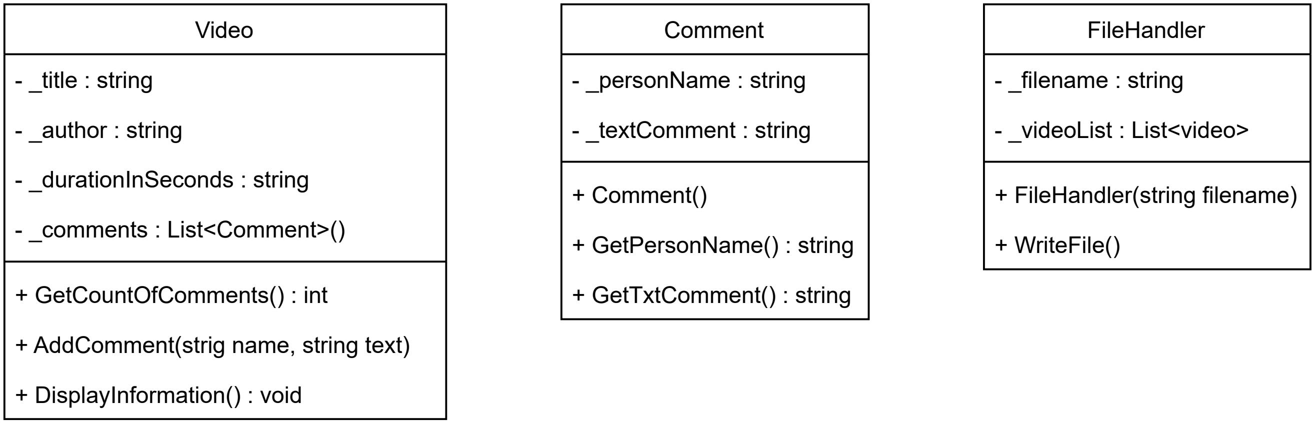
Program 1: Abstraction with YouTube Videos

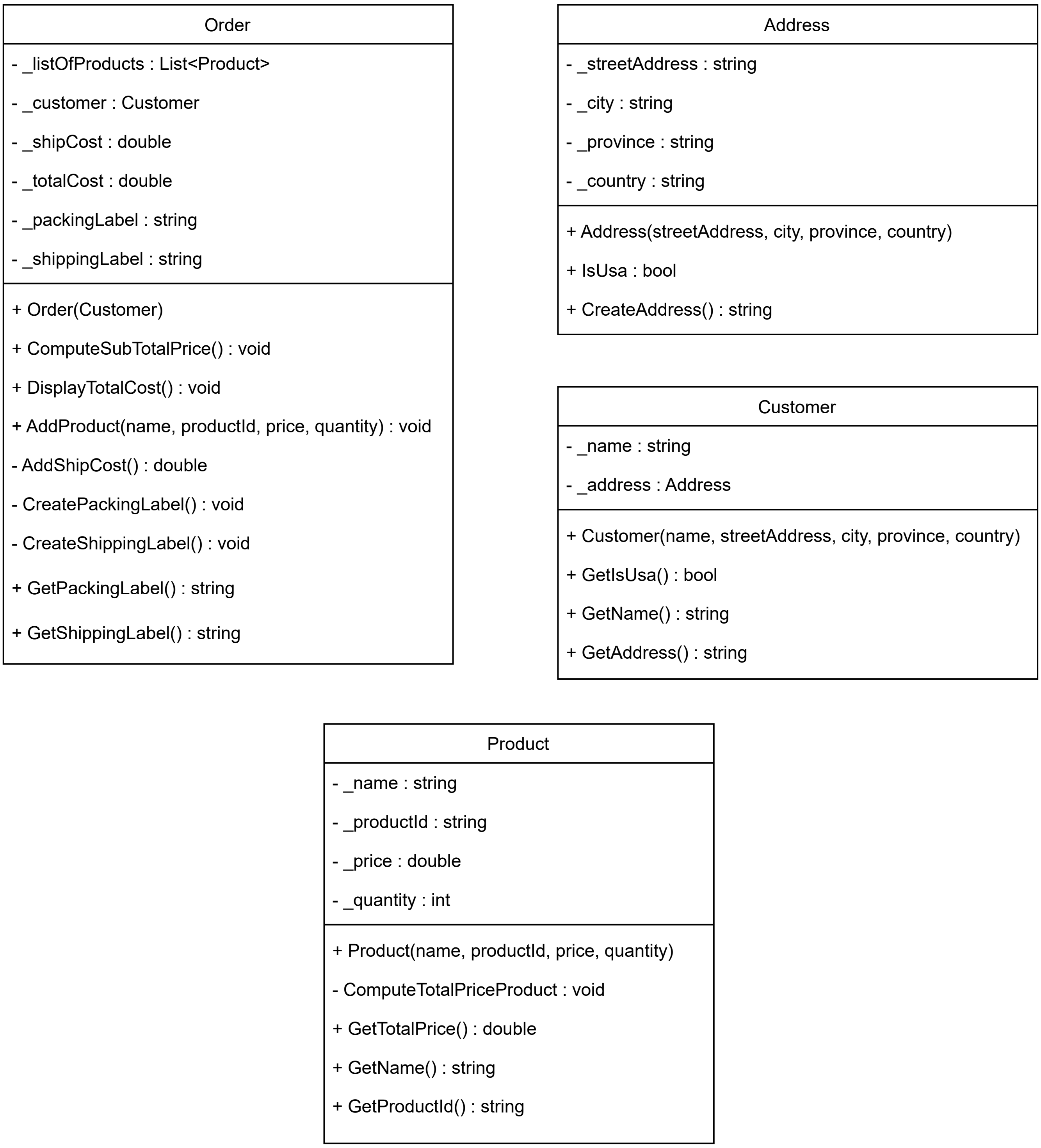


Video
- _title : string
- _author : string
- _durationInSeconds : string
- _comments : List<Comment>()
+ GetCountOfComments() : int
+ AddComment(strig name, string text)
+ DisplayInformation() : void
Comment
- _personName : string
- _textComment : string
+ Comment()
+ GetPersonName() : string
+ GetTxtComment() : string
FileHandler
- _filename : string
- _videoList : List<video>
+ FileHandler(string filename)
+ WriteFile()

Description of Class Diagram

This program allows the creation of videos from the Video class, whose attributes are title, author, duration and comments. It also has methods such as GetCountComments() that returns the number of comments for each video, AddComment() inserts a comment each time this method is invoked in the list of comments, and the DisplayInformation() method that shows all the information of the video in this order: Title, author, duration, number of comments and lists the comments made for each video. Each video can have an indefinite number of comments therefore an attribute called comments was created which is of the Comment class type. The Comment class has two attributes, the name of the person and the comment made on the video. The constructor of the Comment class, establishes the name and comment text of each comment. We will also have a class called FileHandler that will allow to save in a file all the objects created from the Program class through the WriteFile() method.

Program 2: Encapsulation with Online Ordering



Order

- _listOfProducts : List<Product>
- _customer : Customer
- _shipCost : double
- _totalCost : double
- _packingLabel : string
- _shippingLabel : string

+ Order(Customer)
+ ComputeSubTotalPrice() : void
+ DisplayTotalCost() : void
+ AddProduct(name, productId, price, quantity) : void
- AddShipCost() : double
- CreatePackingLabel() : void
- CreateShippingLabel() : void
+ GetPackingLabel() : string
+ GetShippingLabel() : string

Address

- _streetAddress : string
- _city : string
- _province : string
- _country : string

+ Address(streetAddress, city, province, country)
+ IsUsa : bool
+ CreateAddress() : string

Customer

- _name : string
- _address : Address

+ Customer(name, streetAddress, city, province, country)
+ GetIsUsa() : bool
+ GetName() : string
+ GetAddress() : string

Product

- _name : string
- _productId : string
- _price : double
- _quantity : int

+ Product(name, productId, price, quantity)
- ComputeTotalPriceProduct : void
+ GetTotalPrice() : double
+ GetName() : string
+ GetProductId() : string

Description of Class Diagram

This program is designed for the creation of a store sales system. It will create 4 classes, among them are: Order, Address, Customer, Product. The Order class will have the control of all the orders that are generated in the program, this class will have attributes like listOfProducts, customer of type Customer, ShipCost, totalCost, packingLabel and shippingLabel. It will also have a class called Customer that will have the attributes name and address of type Address. Each object created from the Customer class will have an address that will be created from the Address class. The Address class will allow us to store all the information about the street, city, province/state and country, so we make sure that the objects of the Customer class have a defined address. And finally each Order must have a list of products, so we created a class called Product that contains the information of a Product according to the quantity entered in the Order, so we can add products to our order as the customer needs.

Program 3: Inheritance with Event Planning

```
Class Event (Parent)
    Attributes:
        private _eventType: string
        private _eventTitle: string
        private _description: string
        private _date: DateTime
        private _time: DateTime
        private _address: string
    Constructor:
        public Event(eventTitle:string, description:string, address:string, date:DateTime, time:DateTime)
    Behaviors:
        public GetEventType: string
        public GetTitle: string
        public GetDescription: string
        public GetAddress: Address
        public GetDate: string
        public GetTime: string
        public GetStandardDesc: string
        public virtual GetFullDesc: string
        public GetShortDesc: string

Class Lecture (Child)
    Attributes:
        private _speaker: string
        private _capacity: int
    Constructor:
        public Lecture(speaker:string, capacity:int eventType:string, eventTitle:string, address:Address, date:string, time:string) base(eventType, eventTitle, description, address,
date, time)
    Behaviors:
        public override GetFullDesc()

Class Reception (Child)
    Attributes:
        private _rsvp: string
    Constructor:
        public Reception(rsvp:string, eventType:string, eventTitle:string, address:Address, date:string, time:string) base(eventType, eventTitle, description, address, date, time)
    Behaviors:
        public override GetFullDesc()

Class Outdoor (Child)
    Attributes:
        private _weatherForecast string
    Constructor:
        public Outdoor(weatherForecast:string, eventType:string, eventTitle:string, address:Address, date:string, time:string) base(eventType, eventTitle, description, address, date,
time)
    Behaviors:
        public override GetFullDesc()

Class Address
    Attributes:
        private _eventAddress: string
    Constructor:
        public Address(address: string)
    Behaviors:
        public GetAddress: string

Class FileHandler
    Attributes:
        private _filename: string
        private _outputFile: string
    Constructor:
        public FileHandler(filename:string)
    Behaviors:
        public ReadFile(): List<Event>
        public WriteFile(): void
```

Description of Class Diagram

The program consists of multiple classes that interact with each other to manage events and manipulate files. Here's a description of how the elements in the program interact:

The "Event" class is the main class and acts as the parent class. It contains attributes to store information about an event such as event type, title, description, date, time, and address. It also has methods to retrieve the different attributes of the event.

The "Lecture," "Reception," and "Outdoor" classes are child classes of "Event" and inherit its attributes and behaviors. These specific classes represent different types of events with additional attributes, such as speaker for lectures, capacity for receptions, and weather forecast for outdoor events. These classes also have a "GetFullDesc()" method that is overridden to provide a complete description of the event.

The "Address" class represents the event's address and has an attribute to store it. It also has a "GetAddress()" method to retrieve the address.

The "FileHandler" class is responsible for reading and writing files. It has an attribute to store the file name and another for the output file. The "ReadFile()" and "WriteFile()" methods allow reading events from a file and writing events to a file, respectively.

Program 4: Polymorphism with Exercise Tracking

Class Activity: Abstract

Attributes:

private _date: string

private _activityLength: string

Constructor:

public Activity(date:string, activityLength:string)

Behaviors:

public virtual GetDistance: float

public virtual GetSpeed: float

public virtual GetPace: float

public virtual GetActivityLength: float

public GetSummary(): string

Class Running (child)

Attributes:

private _distance: float

Constructor:

public Running(distance: float, date:string, activityLength:string) :base(date, activityLength)

Behaviors:

public override GetDistance: float

Class StationaryBicicle (child)

Attributes:

private _speed: float

Constructor:

public StationaryBicicle(speed:float, date:string, activityLength:string) :base(date, activityLength)

Behaviors:

public override GetSpeed: float

Class Swimming (child)

Attributes:

private _laps: int

Constructor:

public Swimming(laps:int, date:string, activityLength:string, distance:float) :base(date, activityLength)

Behaviors:

public override GetDistance: float

Description of Class Diagram

The program consists of three classes: Activity, Running, StationaryBicicle, and Swimming, with inheritance and specific attributes.

Activity es an abstract class with private attributes (_date and _activityLength) and virtual methods.

Running, StationaryBicicle, and Swimming are child classes that inherit from Activity and add their own attributes and methods.

Each class has a constructor to initialize attributes and behaviors.

The child classes override the virtual methods of Activity to provide specific implementations.

Running overrides GetDistance to return the distance covered.

StationaryBicicle overrides GetSpeed to return the speed of a stationary bicycle.

Swimming overrides GetDistance to return the swimming distance.

The classes can call inherited methods and have additional methods, such as GetSummary in Activity.

Together, these components interact to define different activities, calculate distances, speeds, and summaries.