PYTHON OOP: Design of the Reservation Management System (RMS) for a Resort - Training

Reservation Management System (RMS) for Blue Lagoon Resort , Miami, Florida

To make this a fun learning experience, let us design the classes for the RMS of a boutique resort that rents villas in Miami, Florida.

Different Types of Entities a Resort business will have

1#villa

2#guest

3#reservation

4#resort

Types of Villas

The resort rents four standard villas and two VIP villas. VIP villas are larger and come with a personal yacht. All villas come with a personal assistant.
THE VILLA TYPES ARE :

#villa (Standard Villas)

#vipvilla (VIP Villas)

So now We will model the RMS system with the following classes:

1#stdVilla

2#vipvilla

3#guest

4#reservation

5#resort

Designing Classes and Data Enscapsulation

Classes must protect the most critical asset, our data. On the other hand, they must promote transparency and synergy by exposing certain data to the outside world using get/set access functions. This OOP concept will permeate the design of all of the classes in our example, and each one of them will be designed such that it contains the data and only the data that is relevant to the class characteristics in the real world.

Designing the Villa Class

Standard Villa: One of the least privileged class is villa, in the sense that it encapsulates the least data (the name of the villa and the name of the villa's personal assistant).

It also offers informative functions about the hours that the personal assistant will be on call

and the dates that the villa will be cleaned and keys will be changed.

Also, it has a function to print the label of the gift that is left in the room of each new guest:

VipVIlla:

offers all the features of a Standar Villa plust it offers additional facilites like:

1)longer hours of a Personal assistant

2)personal yatch

In [1]:

```
1 import datetime
2 from datetime import date
```

1.Class Villa

Class villa encapsulates the following rented villa characteristics:

Property/Attributes:
name of villa,
name of personal assistant.

Functions/Behaviours:
It also offers functions that inform about the hours that the personal assistant will be on call and the dates that the villa will be cleaned and keys will be changed.
In addition, it has a function to print the label of the gift that is left in the room of each new guest.

In [2]:

```
class stdVilla(object):

#class villa():
    def __init__(self,n,id):
        self.villaName=n
        self.resid=id

    def setPersonalAssistant(self,pa):
        self.personalAssistant=pa
        #f is for formatting
        print(f"Your Resort PA {pa} will be on call from 8.00am to 8.00pm for Standard
    def cleanAndChangeKey(self,d1,d2):
        print(f"Your Villa {self.villaName} will be cleaned and keys will be changed or
    def printGiftLabel(self,s):
        print(f"**********************")
        print(f"Welcome at the {self.villaName} villa, {s} party!")
        print(f"**********************")
```

Inheritance and Polymorphism

Class vipVilla
offers examples of implementation of both
class inheritance and polymorphism.

As we can see below, the class inherits from class villa

and it redefines the base class method setPersonalAssistant().

Which method gets invoked, gets resolved by Python using method resolution order.

2.VipVilla

Class vipVilla inherits from class villa.
It encapsulates the name of the VIP personal assistant
and offers an access function to it.

In [21]:

```
#Example Inheritance
class vipVilla(stdVilla):
    def __init__(self,nn,id):
        stdVilla.__init__(self,nn,id)
    #Example of overriding-setPersonalAssistant, implements polymorphism. Personal Assi
    def setPersonalAssistant(self,pa):
        self.vipPersAssist=pa
        print(f"Your Resort PA {pa} will be on call (7.00am-9.00pm) for  VIP villa {sel
```

3 Class Guest

Class guest encapsulates the following :
Attributes/Property of a guest:
first and last name, number of adults, and number of children in the room.

Behaviour/Functions
It offers an access function to last name
and a printing function for the guest object. This printing function is used for printing labels with the Guest Name. The Labels will be sticked on the GIFTS that will be given to the Guests.

In [22]:

```
class guest(object):
    def __init__(self,l1,f1,b,c):
       self.first=l1
       self.last=f1
       self.noofAdults=b
       self.noofChildren=c
    def getLastName(self):
        return self.last

    #__repr__
    #The __repr__ method simply tells Python how to print objects of a class

    def __repr__(self):
       return 'Guest: (%s, %s)' % (self.first, self.last)
```

4 Class Reservation

Class reservation encapsulates the following attributes of a reservation: Property/Attributes
the name of the reserved villa, checkin date, checkout date, reservation ID,

Behaviour/Functions:
a printing function for the reservation class.

In [23]:

```
class reservation(object):
    def __init__(self,n,de,le):
        self.checkinDate=de
        self.lengthofStay=le
        self.resVillaName=n
        self.checkoutDate=de+datetime.timedelta(days=le)
    def getResVillaName(self):
         return self.resVillaName
    def getResCheckinDate(self):
        return self.checkinDate
    def getResCheckoutDate(self):
        return self.checkoutDate
    def setReservID(self,id):
        self.reservID=id

    def __repr__(self):
        return 'Reservation: (%s, %s, %s, %s)' % (self.checkinDate, self.lengthofStay,
```

Object Associations

An association is a “using” relationship between two or more objects in which the objects have their own lifetime and there is no owner. ... The objects that are part of the association relationship can be created and destroyed independently.

5 Class Resort

Resort will be using villa, vipVilla, Guest, Reservation objects

Class resort encapsulates the following attributes:
a list with the names of the (standard) villas,
a list with the names of the VIP villas,
a guest list,
a reservation list
and a reservation ID list.

It offers access functions to a Guest object, Reservation object using the following set methods: The methods setGuest() accept guest object

setReservation() accepts reservation object.

def getresID(self):#Returns the last index or last Reservation Id

def updateResIDList:#reservation ID

def printLists(self):

#function that prints all lists. like Guest List, Reservation List, Reservation Ids

In [35]:

```
class resort(object):
      #[list]
      #Names for the Std and Vip villa
      stdVilList=['ElektraStd','PersephoneStd','ArtemisStd','KourosStd']
      vipVilList=['ZeusVip','AlexandrianVip']
      #Empty Guest and Reservation lists
      guestList=[]
      reservationList=[]
      #Names for the Std and Vip villa
      #Reservation Id already initiated at 0 because we want the Reservation to start a
      #the reservation list is intiated at 0.
      resIDList=[0]


      def __init__(self):
      #the Moment the resort object is created ,
      #it will print a message Welcome to Blue Lagoon Resort
          print("Welcome to Blue Lagoon Resort!")
      #setGuest=When a new Guest comes to the Resort,
      #the Guest details are added to the Resort Object
      def setGuest(self,g):
          self.guestList.append(g)
      #setReservation=here the Reservation details are added to the Reservation List.
      def setReservation(self,r):
          self.reservationList.append(r)
      #getresID= gets you last Reservation id.
      #To update the Reservation List with a new Reservation id,
      # we need the last reservation id
      def getresID(self):
          #Returns the last index
          return self.resIDList[-1]
      #getresID= The last Reservation id we get is incremented by +1
      # The new Reservation Id is added to the Reservation List.
      # we need the last reservation id
      def updateResIDList(self):
         i=self.getresID()+1
         self.resIDList.append(i)
         return(self.resIDList[-1])

      def printLists(self):
        print(f"******print Guest List*****************")
        print(f" The guest list is: {self.guestList}")
        print(f"******print Reservation List*****************")
        print(f" The reservation list is: {self.reservationList}")
        #to print the last 2 digits
        print(f" The resID list is: {self.resIDList[1:]}")
        print(f"**********************")
```

In [ ]:

```
1
```

Implementation

Now making the Class, Objects and Business Rules work

The next cell is the test module.
We create a resort object and two guests.

1)The first guest is- Mitchell
The Guest object will have "self.noofAdults=2", "self.noofChildren=1"
The Guest object is passed into the Resort Object.
Reservation object: Guest Mitchell is booking a VIP villa by name- ZeusVip, with Start Date and Duration of stay.
The Reservation Object is passed into the Resort Object.
objResort.updateResIDList()

2)The Second guest is- Marchese
The Guest object will have "self.noofAdults=2", "self.noofChildren=1"
The Guest object is passed into the Resort Object.
Reservation object: Guest Mitchell is booking a Std villa by name- ArtemisStd, with Start Date and Duration of stay.

Finall all the data about the RESORT (All Lists) printed at the end!!

```
objResort=resort()
#Guest 1
saguest=guest('Albert','Mitchell',2,1)
objResort.setGuest(saguest)
objReservation=reservation('ZeusVip',date(2019,6,3),5)


objResort.setReservation(objReservation)
newid=objResort.updateResIDList()
objReservation.setReservID(newid)

#At the time of creating Villa object we check if it is a vipvilla or std villa
#because we have to create objects either Std or VIP.

if objReservation.getResVillaName() == 'ZeusVip' or
objReservation.getResVillaName()=='AlexandrianVip':
    villarese=vipVilla(objReservation.getResVillaName(),newid)
else:
    villarese=villa(objReservation.getResVillaName(),newid)

villarese.printGiftLabel(saguest.getLastName())
villarese.setPersonalAssistant('Eleni')
villarese.cleanAndChangeKey(objReservation.getResCheckinDate(),objReservation.getRes
CheckoutDate())
#print villa reservation id
print("Reservation Id for Guest:" ,villarese.resid)

#Guest 2

saguest2=guest('Simon','Marchese',2,1)
objResort.setGuest(saguest2)
objReservation2=reservation('ArtemisStd',date(2019,7,3),8)
```

```
objResort.setReservation(objReservation2)
newid2=objResort.updateResIDList()
objReservation2.setReservID(newid2)
#print villa reservation id


if objReservation2.getResVillaName() == 'ZeusVip' or
objReservation2.getResVillaName()=='AlexandrianVip':
    villarese2=vipVilla(objReservation2.getResVillaName(),newid2)
else:
    villarese2=stdVilla(objReservation2.getResVillaName(),newid2)
villarese2.printGiftLabel(saguest2.getLastName())
villarese2.setPersonalAssistant('Dorian')
villarese2.cleanAndChangeKey(objReservation2.getResCheckinDate(),objReservation2.get
ResCheckoutDate())
print("Reservation Id for Guest:" ,villarese2.resid)


objResort.printLists()

```