VEHICLE RENTAL SHOP

Yella Sowmya Reddy

Information Technology and Management, The University of Texas at Dallas

MIS 6382.503: Object-Oriented Programming Python

Vatsal Kamleshbhai Maru

12/15/2022

I

RENTAL SHOP

The task that we have undertaken is the establishment of a rental shop. The main goal is to establish a vehicle rental shop that uses dedicated rents for the respective vehicles based on time period. A scooter can be rented for an hour, a day, or a week. The hourly rental fee for a scooter is $5, the daily rental fee is $20, and the weekly rental fee is $50. A person can rent multiple scooters. Promotional discounts are applied when renting more than two scooters, i.e. three to five. A single person renting multiple scooters will receive a 30% discount on the total price. When customers return a scooter, the rental shop can issue an invoice. Total due is calculated by multiplying respective rates and times by the appropriate discount. A customer may select additional scooters but not different time periods, and the request may not exceed the number of scooters available.

The Vehicles like scooters are being rented for the customers based on the time periods assigning a particular amount. Here an individual can rent more than one scooter where we can see that the promotional discounts are being applied to the total price when customers rent more than two scooters .The amount of money that a customer should pay are calculated by applying the appropriate discount to the respective rates and times and multiplying them. In , this report I have used the methods and functions that we used in creating the rental shop.One of the most important frameworks that is being used here is Unittest where the shop information and input of the customers are stored.

Outline

This paper proposes the novel methodologies for creation of rental shops for vehicles. The vehicles that are used here are scooters. A complete scooter rental system written in Python and based on object-oriented programming. The bike rental shop can issue a a bill when customer decides to return the bike.display available and inventory take requests on hourly, daily and weekly basis by cross-verifying stock. For the sake of simplicity, we will assume that any customer requests only one type of rental, namely hourly, monthly, or weekly. Is free to select the number of scooters he/she desires. The number of requested scooters should be less than the number of available scooters. Development Through Testing (TDD), Tests will help you save time. Tests not only detect but also prevent problems. Tests make your code more appealing. Teamwork is aided by tests.The unit test framework is used to test the cases.Diving into the unit test module we have to take note of the following Methods for testing based on the snake case. Test methods are descriptive, and their names indicate what functionality is being tested. Always test for extreme inputs such as null values, zero arrays, non-integer inputs, and invalid dates. Test files will frequently run longer than the main program, which is a good thing.

Description

The steps that we followed here are,Create a car rental module (.py file) and import the built-in module DateTime to handle the rental time and bill. Make a class for renting vehicles and include a constructor in it. Create a method for displaying available vehicles. Define procedures for renting vehicles on an hourly, daily, and weekly basis. Ensure that the number of requested vehicles is positive and less than the total number of available vehicles when using these methods. Save the

time spent renting a vehicle in a variable that will be used in the bill when the vehicle is returned. Define a method for returning the vehicle based on the rental time, mode of rental (hourly, daily, or weekly), and a number of vehicles rented. Within the return method Update the inventory stock, calculate the rental period and generate the final bill. Create a class for customers and include a constructor in it. Define methods for requesting and returning the vehicles. Next, create the main project (.ipynb) file and import the car rental module. Define the main method and create objects for both the vehicle rental and customer classes. Take the customer's input as a choice within the main method for displaying vehicle availability, rental modes, or returning the vehicles. Use the appropriate method for the customer's input and print relevant messages. To begin your project, execute the main method.

Elucidation of the code

First, we used the "import" module to import the unit test module. Import is equivalent to #include header file in C. Python modules can access code from other modules by using the import command to include the file or function. The import statement is the most commonly used method, but it is not the only one.

Import module__name

When an import is used, the __import__ () method is invoked to look for the module in the local scope first. The initial code's output then reflects the value returned by the function. Following that, we created a class called "Customer" to collect and store the necessary information.

The number of vehicles, hours, days, and weeks are referred to as global variables of the person class in this context.

```
class Customer:
    no_of_vehicles = 0
    hours = 0
    days = 0
    weeks = 0
```

We've defined the init method. This is the constructor that was used to store the value of the customer; we also assigned values to the various fields. Next, create a Rental services class in which the number of vehicles available is displayed as a global variable of the Rental service class. In this class, we did mathematical calculations and printed the results. When we define the init method, the constructor for the library class takes the available vehicles

```
def init(self,no_of_vehicles,hours,days,weeks):
        self.no_of_vehicles = no_of_vehicles
        self.hours = hours
        self.days = days
        self.weeks = weeks
```

The amount for object c is calculated here using the formula, which consists of the number of vehicles taken for the specified amount of time. We are attempting to highlight the bill for the rented books by further defining the displayBill.

```
def calculateBill(self,c):
```

```
        amount = c.no_of_vehicles*((c.hours*5)+(c.days*20)+(c.weeks)*50)
        return amount
    def applyDiscount(self,c):
        amount = (self.calculateBill(c))*0.7
        return amount.


      def displayBill(self,c):                    # we are printing the bill
        print("***** INVOICE *****")
        print("Number of vehicles ordered: ",c.no_of_vehicles)
        print("Time spent: ",c.weeks,"weeks,",c.days,"days,",c.hours,"hours.")
        amount = self.calculateBill(c)
        if c.no_of_vehicles <=2:
            print("Total Amount: ", amount)
        else:
            print("Total Amount: ", amount)
            print("Discount  applied: 30%")
            print("Total amount after applying discount: ", self.applyDiscount(c))
            print("**** THANK YOU *****")
```

After calculating the bill on applying the discount we are printing the bill based on the vehicles ordered. Next unit testing is done to test the test cases, we have run a test to calculate the bill and apply the discount. Following the execution of a function "********** Welcome to the Library ********** "will be displayed, and the number of books available in the library will be taken into account. It displays the currently available books, and we must then enter the number of books that we require. If the number of books we entered is greater than it displays "Sorry! only", "books available", and "books are available" when there are a limited number of books available. Then it will ask if you want more books, and if you do, it will display "**********Thank you for using our services**********".Furthermore, we will enter the time constraints such as the number of hours, days, and weeks. If the time constraints are not met, it will display "**** Thank you for using our services***".

Methods

Unit test:

Unittest is a testing framework. It is used to put the test cases to the test. It makes test automation easier by allowing you to share setup and shutdown code, organize tests into collections, and isolate tests from the reporting system. Unit tests, to that end, provide object-oriented support for a number of critical concepts. A test case is a unit of testing code. It searches for a specific response to a specific set of inputs. A unit test includes a base class called TestCase that can be used to create new test cases. The unit test module comes with a complete set of tools for creating and running tests. This section demonstrates how a small subset of tools can meet the majority of people's needs. A test case is created by subclassing the unittest. Testcase. Methods beginning with the letters test define the three individual tests. The test runner is informed by this naming convention which methods represent tests.

__init__:

The __init__ method in Python is the object-oriented equivalent of the C++ constructor. When an object is created from a class, the __init__ function is called. The __init__ method's sole purpose is to allow the class to initialize the object's attributes. It is only used in classes. In the __init__ method, the self keyword is used to set the values passed as arguments to the object attributes.

When you call a class, Python creates a new instance of that class and then calls its __init__ method, passing in the newly constructed instance as the first argument (self).__init__ is not known as the "constructor method" in many programming languages. The initializer method refers to Python's __init__ method. The initializer method creates a new instance of our class. As a result, by the time the initializer method is called, the class instance has already been created. When you create a new Python class, the first method you'll probably create is the __init__ method. You can accept arguments to your class using the __init__ method. More importantly, the __init__ method allows you to initialize various attributes on your class instances.

In this problem, we have used an init constructor in order to store the values of the customer for the customer class and also to take input vehicles available from the rental service class. We can pass arguments to the __init__ method just like we can to any other method in a class. When creating an object in the Python program, we pass a string value to the __init__ method. Class properties that are unique to the class objects can be created and initialized within the __init__ method. Because of the two underscores on either side of its name, the __init__ method is also known as double underscores init or dunder init. The double underscores on both sides of init

indicate that the method is invoked and used internally in Python without the object having to explicitly call it.

Results

```
        Enter the no.of vehicles available in shop: 50
********** Welcome to Car Rental Services **********
50  vechicles available
Please enter no.of vehicles needed: 10
Please enter the time duration
no.of weeks: 3
no.of days: 4
no.of hours: 6
********** INVOICE **********
Number of vehicles ordered:  10
Time spent:  3 weeks, 4 days, 6 hours.
Total Amount:  2600
Discount  applied: 30%
Total amount after applying discount:  1819.9999999999998
********** THANK YOU **********
Do you want to rent more vehicles? Y/N : Y
********** Welcome to Car Rental Services **********
40  vechicles available
Please enter no.of vehicles needed: 7
Please enter the time duration
no.of weeks: 4
no.of days: 8
Error: Days must be less than 7. If more than 7, add to weeks.

********** Welcome to Car Rental Services **********
40  vechicles available
Do you want to rent more vehicles? Y/N : N
**********Thank you for using our services**********
```

We received the result and the total amount to be paid based on the time constraints and availabil ity of the vehicles.We receive an invoice from the app based on the number of vehicless we have rented. If we require more books it will ask for Yes or No .If we enter YES the process will be continued ,If No then it displays **Thank you for using our services** followed by System Exit.

REFERENCES

1. Waspodo, Bayu, Qurrotul Aini, and Syamsuri Nur. "Development of car rental management information system." In Proceeding International Conference onInformation Systems For Business Competitiveness (ICISBC), pp. 101-105. 2011
2. https://www.freeprojectz.com/python-django-project/car-rental-system
3. https://www.scribd.com/document/451743979/online-car-rental-system#
4. https://github.com/topics/car-rental-management-system
5. https://www.coursehero.com/file/49862130/Bike-Rental-Final-Project-Reportpdf/
6. Thakur, A., & Dhiman, K. (2021). Chat Room Using HTML, PHP, CSS, JS, AJAX. International Research Journal of Engineering and Technology (IRJET), 08(June), 1948–1951. https://doi.org/https://doi.org/10.6084/m9.figshare.14869167
7. SharnbasvaUniversity https://www.studocu.com/in/document/sharnbasva-university/bachelors-of-computer-application/documentation-bike-rental-final/30578146