1. User

This is the abstract class from which every human related object in the system will inherit. This abstract class contains the below data members;

Id - This is representative of the object instance's ID number.

Firstname - This represents the first name of the inheriting object

Lastname - This is the last name of the inheriting object

DOB - this represents the date of birth of the inheriting object

Nationality - This stands for the nationality of the objec

IDNumber - This could be nullable if the passport number is provided. It stands for the national ID number of the individual object

PassportNum - This field contains the Passport Number of the object class. This field could be nullable if the ID number has been provided.

This class contains properties for each of it's attributes apart from message which has helper opertations for manipulating it, in addition it contains the below operations:

(1) Constructor - takes the parameters firstname, lastname, dob (date of birth), nationlaity, idNumber. If this sontructor is used then the passport number is made null.

(2) Constructor - This is similar to the first constructor but differs only by the last parameter which instead of the idNumber as in the first case, this second one takes a passport number and nullifies the ID number.

(1) SendMessage - This polymorphed function takes a single message which calls user.Inbox() which appends messages to the user's list of messages.

(2) SendMessage - This polymorphed function takes a list of messages and sends it (by appending to the inbox)

CheckInbox - This function calls message.Print() which returns all information about the message namely, date, content, source adn reciepient. The message.Print() method changes the read status to true.

ClearInbox - This function clears the list of messages in the object.

Inbox - this function appends messages to the users' list of messages. it is used in sending messages around.

Print() - This is an abastract function to be overriden by derived classes.

2. Owner

This class is meant to represent any entity who owns property(ies). Thi object therefore owns the property(ies) which in turn own the Customer(s) and Employee(s) - to be explained in the property section. Appropriately therefore, this class inherits from the User class all the identification particulars, which are protected fields in the User class, and implementes some additional private fields namely;

publicKey - This will be the identifying key when this object interacts with other object where the instructions being sent would need verification. Like adding a new employee, this is only done internally.

privateKey - This field, along with the publicKey, is used in providing verification that instructions came from the specific owner.

property - This field contains the list of property the owner owns.

The Owner class contains properties for each field in the class in addition to the below operations:

(1) Cosntructor - This first contrustor extends from the first constructor in the User absrtact class. Therefore, it takes the parameters firstname, lastname, dob (date of birth), nationlaity, idNumber. If this sontructor is used then the passport number is made null.

(2) Cosntructor - This constructor extends from the second constructor in the base class User. This is similar to the first constructor but differs only by the last parameter which instead of the idNumber as in the first case, this second one takes a passport number and nullifies the ID number.

(1) AddProperty - This is a polymorphed function that takes a single prperty, and appends it to the list of properties in the object.

(2) AddProperty - This polymorphed operation takes a list of properties as a parameter and appends each one of them to the list of properties in the owner object.

UpdateProperty - This function provides a utility for updating property information. It takes a property parameter and updates the preexisting on by searching for the passed property's id in the lists of properties then updating the preexisting values with the item passed.

RemoveProperty - This is used for removing a property from the list of properties of the object.

EmployeeAnnouncement - The owner can use this utility to cast a message to all employees in a certain property, byt passing the parameter propertyID. This operation takes a parameter of property id and after finding the id in the list, it calls user.Inbox for every employee in that property in order to send them the message.

ContactEmployee - This is similaar so sending a message however in this case the owner will be contacting on employee at a time. This function takes the parameters propertyID and employeeID

RemoveEmployee - This is the analogous of firing an employee, it finds first then removes an employee from the list of employees in a property.This function takes the parameters propertyID and employeeID.

AddEmployee - This is the analogous of hiring and it adds the employee to the list This function takes the parameters propertyID and employee instance.

AssignTask - This function takes the parameters propertyID, employeeID and a task instance. This opertation is used in assigning tasks to employees and makes calls to the employee.Tasks property to append tasks to the list of tasks in the employee tasks field.

RelieveTask - This function takes the parameters propertyID, employeeID and a task instance. This operation is used to relieve an employee of a duty it is also called when removing an employee from a property. After finding the property and the employee using their ids it removes the tasks from the List<Task> in the employee tasks field.

AddStream - The owner may choose to add a new revenue stream or source of income, in which case this implementation is used. The opertaion takes parameters, a revenueStream instance and a propertyID. After finding the property in the list of properties it appends the revenue stream in the List<RevenueStream> of the revenuestreams field.

RemoveStream - If the owner decides he's(she's) not getting enough income from a specific revenue stream they may choose to remove it. This Operation takes parameters streamID and propertyID and after finding the property and the stream it calls property.RemoveStream to remove the stream.

PrintBilling - This methosd takes a parameter propertyID allowing the owner to print the Billing from a certian property. This method, after finding the property calls the method property.Billing().Print() which returns the detailed particulars of the Billing to the owner.

ShowTrend - This method takes the parameter propertyID and finds the property after which it calls property.Billing().Trend().Print() which returns the trend of the billing. The owner can also provide start adn end dates to get a specific trend. Failure to providing the dates the program prints a two month trend.

Print - This prints out information about the owner, it also shows the count of properties they own and the number of employees and customers the properties have and which revenue strams they offer. This is possible because the method calls property.Print() which verbosely returns information about the property(ies)

3. Customer

This object inerits from the User class taking all of its protected and public fields and methods. The customer is limited in what they can do here since this is a remote management software made for admins who are owners, however the customer is allowed some interactivity in responding to what is sent to them. It additionally implements;

cardNumber - this is going to contain the customer's method of payment. THis could be their physical bank card or an ewallet code.

cart - this contains a List<RevenueStream> that the user has purchased from a property.

subscription - this contains a List<RevenueStream> that the user has subscribed to from a property.

This object contains properties for all it attributes, it also inherits all of the user's public opertaions as well a overrides the print method, additionally it implements the below.

Additional Operations;

(1) Cosntructor - This first contrustor extends from the first constructor in the User absrtact class. Therefore, it takes the parameters firstname, lastname, dob (date of birth), nationlaity, idNumber. If this sontructor is used then the passport number is made null.

(2) Cosntructor - This constructor extends from the second constructor in the base class User. This is similar to the first constructor but differs only by the last parameter which instead of the idNumber as in the first case, this second one takes a passport number and nullifies the ID number.

ConfirmSubscription - this function adds items to the customer subscription. It therefore takes a parameter of a stream and adds it in the List<RevenueStream> in the customer subscription field. If an existing on is found the stream is added with a differentID.

RemoveSubscription - This function removes a revenuestream from the customer's subscritpion. It therefore takes a parameter of a stream and removes it after finding in in the List<RevenueStream> in the customer subscription field.

ClearCart - This clears the cutomer cart

Checkout - This Send a message to the owner that the cutomer bought something(s).

Print - This method is overriden and returns the details of the Customer along with their cardnumber, cart, and subscriptions

4. Employee

This object inerits from the User class taking all of its protected and public fields and methods. It additionally implements;

role - This is the job role of the employee

status - this shows whether the worker is on leave, on site or off duty

tasks - This contains all the tasks of the employee.

The employee class implements properties for all of its attributes,along with those in herited from the base user class as well as overrides the Print method from the base class. In addition is contains the below operations;

(1) Constructor - This first contrustor extends from the first constructor in the User absrtact class. Therefore, it takes the parameters firstname, lastname, dob (date of birth), nationlaity, idNumber. If this sontructor is used then the passport number is made null.

(2) Constructor - This constructor extends from the second constructor in the base class User. This is similar to the first constructor but differs only by the last parameter which instead of the idNumber as in the first case, this second one takes a passport number and nullifies the ID number.

ApplyToChangeRole - this allows the employee to send a message to the owner that they'd like to change positions.

ChangeWorkStatus - this allows the employee to clock in and out and post that they're on a sick leave or maternity leave

Print - The employee can print all the information about themselves in this case.

5. WorkStatus

This is an enumeration used by the employee class to provide the employee work status. The below listed a the enumeartion literals in this enumerations

MaternalLeave - this is set when the employee is off due to a meternal leave

SickLeave - This is set when the employee cannot report to work because they are sick

OnDuty - This means the employee is expected to be onsite at that specific moment

OffDuty - This means that the employee is not on their shift at that specific moment.

6. Task

This class represents assignments given to employees. This class implements the below fields.

Name - The name of the task

Description - this contains information about the task

StartDate - This is the date the task is expected to start.

EndDate - This is the task's dealine.

Status - This refers to whether the tasks is complete or not.

Assignee - List of employees assigned the task

This class implements properties for all of its fields apart from the assignee which uses some other utility operations mentioned below.

Constructor - This constructor takes teh paramters name, description, start and end Dates. The assignee list is defaulted to an empty list.

AssignTask - This polymorphed operation takes and employee parameter and adds teh employee to the assignees. This is called by the property.

AssignTask - This polymorphed operation takes an employee id parameter and add the employee to the assignees. This is called by the property.

RelieveTask - This function removes the employee parameter passed from the assignees.

ChangeStatus - this change status to either true (maps to complete) or false (maps to incomplete)

MoveDeadline - this changes the end date of the task

Print - This prints out all the details of a tasks including name, description, start and end dates, status, and the assignees

7. Message

This class implements the below fields,

Date - This is the date of sending/creating the message

Content - this is the content of the message

Read - This shows whether the message has been read or not

Source - this contains the sender of the message

Receipient - This contains a list of receipients

This class implements properties for all of its fields. in addition it implements the below operations;

Constructor - takes the parameters content and a list of reciepients. The list of receipients are appended to the reciepients field, the date of the message creation is assigned to the date field the read status defaults to false, and the user sending it is assigned to User. Since this method is only accessible through a user, the user instantiating it gets to be the source. This happens in the user.SendMessage polymorphed operations.

Print - This operations prints out every detail of the message inclusive of its date, content source, and receipients. It also changes the status read to true if it was false.

8. Trend

This class contains the fields listed here;

startDate - The start date of the trend

endDate - the end date of the trend

parameters - this parameter is of type T making able to take any type where T is a class. This allows us to check revenueStream trends by passing a revenurStream as the T, employee turn over trend by passing the employees dropped count, the customer trend by passing the customer count as the parameter.

This class implements the below operations;

(1) Constructor - this constructor takes the parameters start and end dates as well as the revenue stream of type RevenueStream

(2) Constructor - this constructor takes the parameters start and end dates as well as the employees dropped of type int

(3) Constructor - this constructor takes the parameters start and end dates as well as the customer count of type int

Print - This method prints out the trend of the parameter passed

9. RevenueStream

This is an abstract class from which product and service derive. This class implements the below fields;

id - this is the id of the particular revenue stream

name - this is the name of the particular revenue stream

description - this contains the description of the revenue stream

price - this contains the price of the revenue stream

The class implemeents properties for all of its fields and also contains the below operations;

Constructor - this operation takes the parameters name, descrption adn price whereas the id defaults to a randoms number. The function random.Next() is used to prevent getting the same number twice.

Print - This operation is an abstract function whic the derived classes override.

10. Product

This class inherits from the base class revenue Streams from which it takes the protected fields mentioned above. it aditionally implements;

maufactureDate - This is the date of the product manufacturing

expiryDate - this is the expity date of the prouduct

The class implements properties for all of its fields, in addition it contains the below operations;

Constructor - thisoperation takes the parameter

Print - this operation overrides the base class' print operation and return every detail of teh product, this includes the inherited fields

11. Service

12. Billing

13. ElectricityMeter

14. WaterMater

15. GasMeter

16. IOT\_Device

17. Property