Activity 3: Class Diagrams

Table of Contents

---

1. Use Class, Responsibilities and Collaboration (CRC) Cards

a. UserInterface

Class: UserInterface	
UserInterface provides the basic login functionality and the main menu display and dispatch for the system.	
Responsibility:	**Collaborators:**
Prompts the user to enter a valid username and password	
Checks to see if the username and password are valid	
If valid, initializes empty TenantList, RentRecords and ExpenseRecords. Instantiates TenantInputScreen(TenantList), RentInputScreen(TenantList, RentRecords), ExpenseInputScreen(ExpenseRecords), and AnnualReport objects for reuse. This is all done once before displaying the main menu for the first time.	TenantList RentRecords ExpenseRecords TenantInputScreen RentInputScreen ExpenseInputScreen AnnualReport
Displays the main menu and prompts the user to choose one of the presented information input or output options.	
Dispatches a valid user selection to the appropriate collaborator.	TenantInputScreen RentInputScreen ExpenseInputScreen AnnualReport

b. TenantInputScreen

Class: TenantInputScreen	
System-provided user interface class for adding or replacing a **Tenant** in the **TenantList**.	
Responsibility:	**Collaborators:**
Prompts the landlord to enter the tenant's name, storing it in private memory.	
Prompts the landlord to enter the apartment number, storing it in private memory.	
If the apartment is occupied, prompts the landlord to confirm tenant replacement.	TenantList, Tenant
Adds new tenant and apartment number to TenantList.	Tenant, TenantList
Outputs new tenant's name and apartment number.	Tenant

c. RentInputScreen

Class: RentInputScreen	
System-provided user interface for adding or replacing a **rent_payment** in the **payment_list**.	
Responsibility:	**Collaborators:**
Prompts the LandLord to enter a specified tenants name	
Prompts the LandLord to enter a specified amount paid (including cents)	
Prompts the LandLord to enter the month the rent is for (1-12).	
The system takes the sum of the current amount and the amount paid. If there is a value already posted, we will add it to the paid amount's value.	TenantList RentRow RentRecords

d. ExpenseInputScreen

Class: ExpenseInputScreen	
System-provided user interface for adding or replacing an **Expense** in the **ExpenseRecords**.	
Responsibility:	**Collaborators:**
Prompts the LandLord to enter a specified month for record of expenses paid	
Prompts the LandLord to enter a specified day in the month for record of expenses paid	
Prompts the LandLord to enter(String value) a category regarding the expense paid out	
Prompts the LandLord to enter the payee (person paid) regarding who accepts the capital.	
Prompts the LandLord to enter the amount paid out.	
Input values are used as parameters to create an **Expense** object for storage by **ExpenseRecords** collaborator	Expense ExpenseRecords

e. Tenant

Class: Tenant	
Tenant provides a class object type that contains an apartment number and tenant name and public methods used with this object.	
Responsibility:	**Collaborator:**
Stores apartment number and tenant name in private memory.	
Deletes this Tenant object containing an apartment number and tenant name from memory.	
Returns apartment number.	
Returns tenant number.	
Returns apartment number and tenant name.	
Returns True if a tenant occupies the apartment.	
Returns True if this tenant name is equal to another given Tenant object tenant name, else False.	
Returns True if this tenant name is alphabetically before another Tenant object tenant name, else False.	
Returns a formatted string containing the apartment number and tenant name.	

f. TenantList

Class: TenantList	
The **TenantList** class maintains a list of **Tenant** objects in private memory and provides public methods for list manipulation and output.	
Responsibility:	**Collaborator:**
Maintains a list of Tenant objects in private memory.	Tenant
Deletes the list of Tenant objects from private memory.	Tenant
Returns the private index position of apartment number and/or tenant name in the list, else None.	Tenant
Inserts a Tenant object to the list or deletes one.	Tenant
Returns the number of apartments and tenants in the list.	Tenant
Returns the number of Tenant objects in the private list	
Returns the Tenant object given the apartment number and/or tenant name if it exists in the list.	Tenant
Returns a string with each row containing an apartment number and tenant name (if any).	Tenant
Outputs the tenant list sorted by apartment number.	Tenant

g. RentRecord

Class: RentRecords	
RentRecords maintains a list of RentRow objects for each apartment. All records from **RentInputScreen** will be appended to the rent record of a tenant specified by apartment number**.**	
Responsibility:	**Collaborator:**
Stores rental inputs from specified valid apartment numbers in private memory.	TenantList
Sums the rental input if there is a current value greater than one stored in the specified index (month)	
Returns the list of all monthly payments (even if the value is 0) for a specified Tenant object.	TenantList, RentRow
Prints the RentRow list for the tenants with each month specified.	TenantList, RentRow

h. RentRow

Class: RentRow	
RentRow maintains a list of received monthly rents from a tenant in private memory and provides public methods for list manipulation and output.	
Responsibility:	**Collaborator:**
Stores rent amount for specific apartment number under a specified month in a private list	RentInputScreen, TenantList
Sums the rental prices of an entire row (1 year) if there is at least one rent amount inputted	

i. AnnualReport

Class: AnnualReport	
AnnualReport is a yearly report that gives the total rents paid, expenses for each budget category, and a resulting balance.	
Responsibility:	**Collaborators:**
Display RentRecords payments	RentRecords
Display ExpenseRecords for budget categories	ExpenseRecords
Calculate Resulting Balance	RentRecords ExpenseRecords
Display Resulting Balance	RentRecords ExpenseRecords

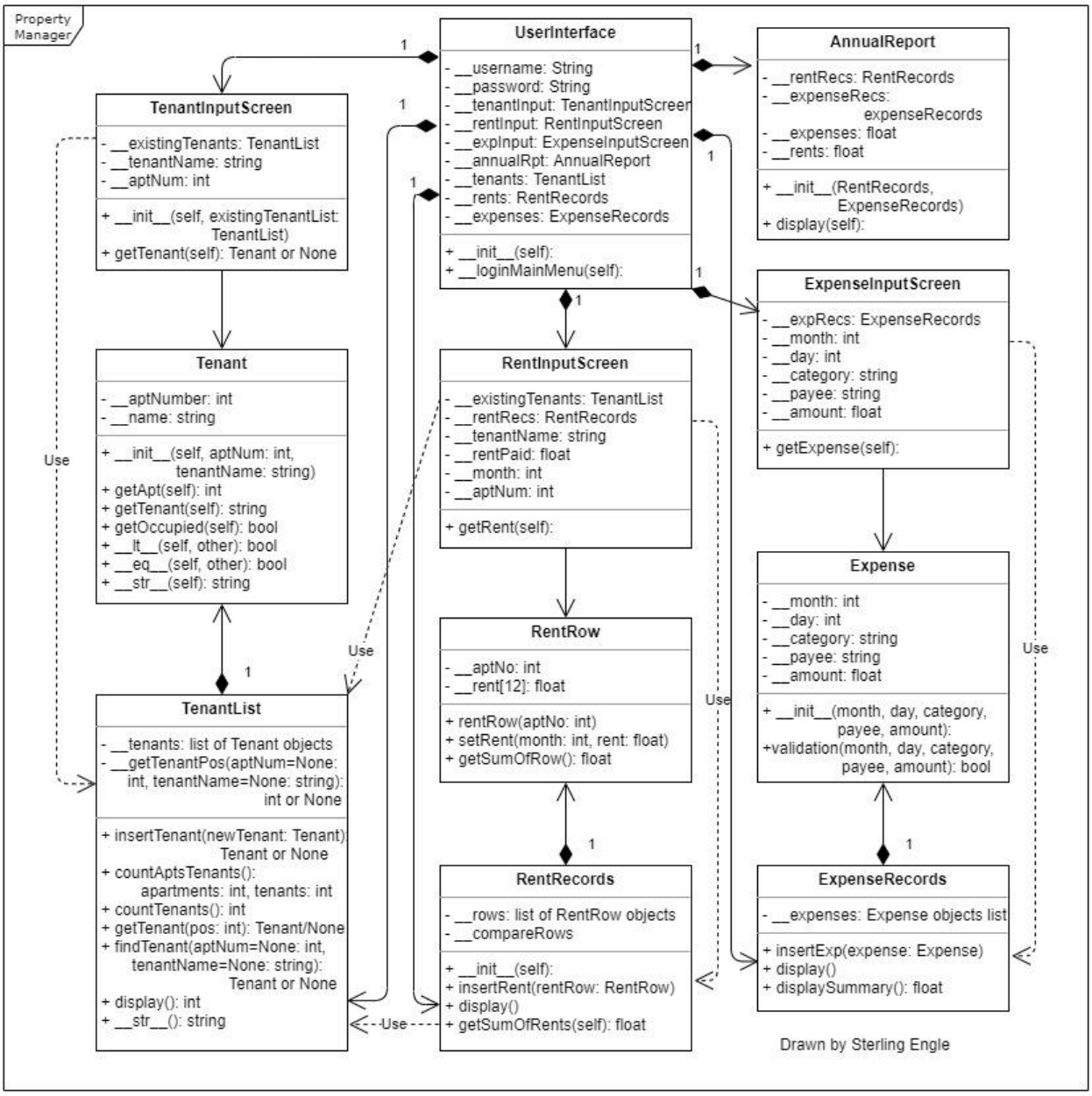
j. Expense

Class: Expense	
Expense class is an object class that holds the attributes of date, category, amount, and payee (person paid out to).	
Responsibility:	**Collaborator:**
Construction of our object that will take the user-inputted information	
Assign our expense with input parameters	
Check if inputs have been used or created already.	

k. ExpenseRecord

Class: ExpenseRecords	
ExpenseRecords maintains a list of Expense objects for each apartment. All records from ExpenseInputScreen will be appended to the expense record of a tenant specified by the apartment number. All records from **ExpenseInputScreen** will be created in the **Expense** class then added by **ExpenseRecords**.	
Responsibility:	**Collaborator:**
Stores expense inputs from specified valid apartment numbers in private memory.	TenantList
Sums the expense input if there is a current value greater than one stored in the specified index (month)	
Returns the list of all monthly expenses (even if the value is 0) for a specified Tenant object.	TenantList, AnnualReport
Prints the AnnualReport list for the tenants with each month specified.	TenantList, AnnualReport

2. Overall Class Diagram



The overall class diagram shows the two-level hierarchical class structure of the property management system. At the top of this hierarchy is the class UserInterface. After the landlord successfully logs in, UserInterface instantiates one private object for each input screen class and AnnualReport for reuse. It also creates private objects for TenantList, RentRecords, and ExpenseRecords for data storage. These maintain lists of multiple Tenant, RentRow, and Expense objects, respectively.

3. Designed Classes with Attributes and Methods

a. UserInterface

UserInterface
- __username: string - __password: string - __tenants: TenantList - __rents: RentRecords - __expenses: ExpenseRecords - __tenantInput: TenantInputScreen - __rentInput: RentInputScreen - __expenseInput: ExpenseInputScreen - __annualRpt: AnnualReport
+ __init_(self): + loginMainMenu(self):

UserInterface is our login and main menu user interface class. It defines nine private members. The program's main() entry point instantiates one object of UserInterface, then calls its loginMainMenu() public method to proceed to the login function, followed by the main user interface menu if login is successful.

Since this program is only meant for one user (John Nguyen), the program will not be required to store multiple users and password objects. Instead, UserInterface initially serves as a barrier or a lock to protect personal information about the landlord's apartment. The login part of this class will prompt and input a username and password and check to see if the user has entered the correct username and password to log into the system.

Before the menu is displayed for the first time, the constructors for TenantList, RentRecords, and ExpenseRecords are each called once to create and save empty lists in their corresponding private class variables. Next, UserInterface instantiates reusable objects: RentInputScreen(TenantList, RentRecords), AnnualReport(), TenantInputScreen(TenantList), and ExpenseInputScreen(ExpenseRecords). After instantiating them, UserInterface displays the main menu and prompts the landlord to enter an option. UserInterface dispatches the action method for the reusable object class corresponding to the selected option if the option is valid. When the action returns, UserInterface re-prompts and calls the next action.

b. TenantInputScreen

TenantInputScreen
- __existingTenants: TenantList - __tenantName: string - __aptNum: int
+ __init__(self, existingTenantList: TenantList) + getTenant(self): Tenant or None

The TenantInputScreen class is a user interface object with three private members: an existing tenants list; and landlord input tenant name and apartment number. The TenantInputScreen public constructor __init__(self, existingTenantList: TenantList) returns a reference to a TenantInputScreen object.

The TenantInputScreen class has one public method, getTenant(self), that prompts the landlord for a new tenant name and apartment number. If there is an existing tenant in that apartment, the method prompts the user to replace them optionally. Collaborator TenantList adds the new Tenant object to the sorted list by apartment number. TenantInputScreen prints the string returned by the collaborator Tenant class containing the new tenant's apartment number and name. getTenant(self) returns the new Tenant object or None if not entered or the landlord did not overwrite an existing tenant.

c. RentInputScreen

RentInputScreen
- __existingTenants: TenantList - __rentRecs: RentRecords - __tenantName: string - __rentPaid: float - __month: int - __aptNum: int
+ getRent(self):

RentInputScreen is an input class where information given will represent instances of rent from tenants. The RentRecords object will store the rent value as an object instance of RentRow. This class has six private members, four of which will become definite values once the user inputs data. Invalid data types will not be accepted, and an exception will be flagged.

The getRent public method prompts the user to enter necessary data for storage in the RentRecords list. If data already exists in memory, an exception will be flagged to notify the user that data has already been input.

d. ExpenseInputScreen

ExpenseInputScreen
- __expRecs: ExpenseRecords - __month: int - __day: int - __category: string - __payee: string - __amount: float
+ getExpense(self):

The ExpenseInputScreen is an input class where information given will represent an instance of our expenses. ExpenseInputScreen will eventually pass this value to ExpenseRecords, which will store the Expense object in its list. The ExpenseInputScreen has six public members, five of which become definite values once we have our user input values. Exceptions will be flagged and will not accept other data types.

The getExpense function will prompt the user to enter values for all instance variables, which ExpenseInputScreen will use to create our Expense object (the item intended to be stored). If the object already exists in ExpenseRecords memory, a flag will raise notifying the user that the information entered is already stored in the list of objects.

e. Tenant

Tenant
- __aptNumber: int - __name: string
+ __init__(self, aptNum: int, tenantName: string) + getApt(self): int + getTenant(self): string + getAptTenant(self): int, string + aptOccupied(self): bool + __lt__(self, other): bool + __eq__(self, other): bool + __str__(self): string

The Tenant class is an object type that has two private members: an __aptNumber integer and a (tenant) __name string. The tenant constructor takes parameters integer aptNum, string tenantName, then returns a reference to a tenant object.

The Tenant class provides three public accessor methods: getApt(self) returns the aptNumber integer, getTenant(self) returns the name, and getAptTenant(self) that returns the aptNumber integer and the tenantName string. One public status method returns True if a tenant occupies the apartment, else False.

There are two overloaded comparison operators: __lt__(self, other) for '<', and __eq__(self, other) for '=='. They compare the tenant names from two Tenant objects ignoring their apartment numbers, returning True or False, for sorting by tenant name. Finally, __str__(self) is a public method that overloads print() for Tenant object types. It returns the apartment number and tenant name as a string.

f. TenantList

TenantList
- __tenants: list of Tenant objects - __getTenantPos(self, aptNum=None: int, tenantName=None: string): int or None
+ __init__(self) + insertTenant(self, newTenant: Tenant): Tenant or None + countAptsTenants(self): apartments: int, tenants: int + countTenants(self): int + getTenant(self, pos: int): Tenant or None + findTenant(self, aptNum=None: int, tenantName=None: string): Tenant or None + display(self): int + __str__(self): string

The TenantList class creates a list object type with one private member: a tenants list of Tenant objects. It has a private method: __getTenantPos(self, aptNum=None: int, tenantName=None: string) used by insertTenant() to see if aptNum already has a tenant to replace. The TenantList public constructor __init__(self) returns a reference to a TenantList object containing an empty private tenants list. There is also a public __del__(self) object destructor method.

The TenantList class provides seven public methods. insertTenant(self, newTenant) inserts a Tenant object in the list of tenant objects sorted by apartment, returning the Tenant object inserted or None if error. It overwrites any existing tenant in the same apartment. It removes an apartment if a negative apartment number equals one without a minus sign. countAptsTenants(self) returns the number of apartments and tenants, while countTenants(self) returns the number of Tenant objects. getTenant(self, pos) returns the Tenant object at sorted index position pos where the first is zero, or None. The TenantList class has one public accessor method. findTenant(self, aptNum=None, tenantName=None) returns the Tenant object for the given apartment number and/or tenant name, or None if not found. display(self) is a public method that outputs a list of apartment numbers and their tenant names, one per row, sorted by apartment number. Finally, __str__(self) is a public method that overloads print() for a TenantList object type. It returns the string that display(self) outputs.

g. RentRecords

RentRecords
- __rows: list of RentRow objects - __compareRows
+ __init__(self): + insertRent(row: RentRow): bool + display(self): + getSumOfRents(self): float

Class RentRecords holds rent received data for the tenants. It stores rental payment inputs in private memory, containing a list of RentRow objects from the RentRow class. We are also able to compare rows. There is a constructor for the record. Public method insertRent() inputs a RentRow object into the __rows list. Public method display() outputs the RentRecords. getSumOfRents() returns the sum of paid rents in the RentRecords.

h. RentRow

RentRow
- __aptNo: int - __rent[12]: float
+ __init__(aptNo: int) + setRent(month: int, rent: float) + getSumOfRow(): float

The RentRow class holds a row of the rent record containing the apartment number and 12 monthly received rent amounts. The private variables here are the apartment number, __aptNo, and array of rents received, __rent[12]. The __init__(aptNo: int) constructor creates an instance of a row. The setRent function will simply record the rent for a specific month. getSumOfRow function returns the sum of rents that are the row.

i. AnnualReport

AnnualReport
- __rentRecs: RentRecords - __expenseRecs: ExpenseRecords - __expenses: float - __rents: float
+ __init__(rentRecs: RentRecords, expenseRecs: expenseRecords): + display(self):

AnnualReport has four private class members (__rentRecs, __expenseRecs, __expenses, __rents) used to summarize expenses and rent information input by the landlord. The display() method outputs summary information of total rents paid, expenses for each budget category, and a resulting balance for the year to date for the user.

j. Expense

Expense
- __month: int - __day: int - __category: string - __payee: string - __amount: float
+ __init__(self, month, day, category, payee, amount): + validation(self, month, day, category, payee, amount): bool

The Expense class has five private members (__month, __day, __category, __payee, __amount) dedicated to serving the object. The constructor consists of all of our private members because they define what makes each object unique. **ExpenseRecords** stores Expense objects in a list to track the landlord's expenses.

Expense only has two methods which are the overloaded constructor and the validation method. The overloaded constructor sets the object's values. The validation() method ensures that the user has entered valid information. **ExpenseRecords** checks this object to ensure its data is unique.

k. ExpenseRecords

ExpenseRecords
- __expenses: list of Expense objects
+ insertExp(expense: Expense) + display() + displaySummary(): float

Class ExpenseRecords maintains a list of expense records for the landlord. insertExp(expense) will take inputs to be created by passing Expense objects ready to be recorded. Public method display() outputs the expense history. AnnualReport calls the displaySummary() method.