Object-Oriented Programming Lab#8, Fall 23

Today's Topics

- Inheritance
- Encapsulation
- Polymorphism
- Abstraction

ArrayLIst:

Action	Code
Creating an ArrayList	ArrayList<T> list = new ArrayList<T>();
Adding element to arraylist	list.add(T t);
Adding multiple elements to arraylist	list.addAll(ArrayList<T> t);
Remove an element	list.remove(int index) list.remove(T t)
Remove multiple elements from an arraylist	list.removeAll(ArrayList<T> t);
Accessing an element	List.get(int index)
Size of arraylist	list.size();

Problems/Assignments – Property Management System

Create an Online property management system to help the owner of the property to manage and rent their properties. There will be 2 types of users of this system; admin and customer who wants to rent the property. Bothalso has to create account if he/she wants to rent a property.

Here is the list of the classes to implement the Application

A. **Create a project name "PropertyManagementLib" and add the following class to this project**

1. **Property** Class (an **abstract** class and under **uap** package):
    a. Attributes (all private): id, ~~description~~, location, ~~category~~, ~~ArrayList<String> facilites,~~ floorspace, rent, isAvailable, rentedBy
    b. Constructor: pass parameters for all except **isAvailable** and **rentedBy**. Inside the constructor, initialize the attributes with respective parameters and set the **isAvailable** to true.
    c. Methods:

Method Header	What the method should do?
add getter methods for all attributes	
add setter method for rent, isAvailable, and rentedBy	
public void rentProperty(double rent, String rentedBy)	set the **isAvailable** to false, **rent** and **rentedBy** using the setter methods.
public double giveMonthlyPayment()	Return the value of **rent** variable.

public void leaseOver()	set the **isAvailable** to true and **rentedBy** to **null** using the setter methods.
public abstract double getSecurityDeposit()	An abstract method.
public String toString()	Return the attribute values as String.

2. **Apartment** class – a subclass of **Property** class and under **uap** package
    a. **Attributes** (all private): noOfBedRoom, noOfWashRoom, hasGenerator
    b. **Constructor** – pass parameter for all except **isAvailable** and **rentedBy**. Inside the constructor, call parent's constructor and initialize the rest of the attributes with respective parameters.
    c. **Override** getSecurityDeposit() method – return the rent of 3 months as securityDeposit.

3. **Store** class – a subclass of **Property** class and under **uap** package
    a. **Attributes** (all private): hasFireExit
    b. **Constructor** – pass parameter for all except **isAvailable** and **rentedBy**. Inside the constructor, call parent's constructor and initialize the rest of the attributes with respective parameters.
    c. **Override** getSecurityDeposit() method – return the rent of 6 months as securityDeposit.

4. **OfficeSpace** class – a subclass of **Property** class and under **uap** package
    a. **Attributes** (all private): **minLeaseTime (in month)**
    b. **Constructor** – pass parameter for all except **isAvailable,** and **rentedBy**. Inside the constructor, call parent's constructor and initialize the rest of the attributes with respective parameters.
    c. **Override** getSecurityDeposit() method – return the rent of 6 months as securityDeposit.

5. **PropertyManager** class (under **uap** package):
    a. Attributes (all private): name, ArrayList<Property> properties
    b. Constructor- pass parameter for name. Inside the constructor, initialize the name attribute with the parameter and instantiate the ***properties*** arraylist.
    c. Methods:

Method Header	What the method should do?
public void addProperty(String id, String location, double flooSpace, double rent, int noOfBedRooms, int noOfWashRooms, boolean hasGenerator)	Create an object of **Apartment** class using the parameters and add the object to ***properties*** attribute/list
public void addProperty(String id, String location, double flooSpace, double rent, boolean hasFireExit)	Create an object of **Store** class using the parameters and add the object to ***properties*** attribute/list
public void addProperty(String id, String location, double flooSpace, double rent, in minLeaseTime)	Create an object of **OfficeSpace** class using the parameters and add the object to ***properties*** attribute/list
public ArrayList<Apartment> findApartments(String location, int noOfBedRooms, int noOfWashRooms)	Loop through the ***properties*** attribute and find the apartments that has matching attributes and return all those apartments as an arraylist. If no apartment found, an empty arraylist will return.

public ArrayList<Apartment> findApartments (String location, int noOfBedRooms, int noOfWashRooms, int minRate, int maxRate)	Loop through the ***properties*** attribute and find the apartments that has matching attributes and return all those apartments as an arraylist. If no apartment found, an empty arraylist will return.
public ArrayList<Store> findStores(String location, double minFloorSpace, double maxFloorSpace, int minRate, int maxRate)	Loop through the ***properties*** attribute and find the vehicles that has matching attributes and return all those vehicles as an arraylist. If no vehicle found, an empty arraylist will return.
public ArrayList<OfficeSpace> findOfficeSpaces(String location double minFloorSpace, double maxFloorSpace)	Loop through the ***properties*** attribute and find the restaurants that has matching attributes and return all those restaurants as an arraylist. If no restaurants found, an empty arraylist will return.
public Property findProperty(String id)	Loop through the ***properties*** attribute and find the Property that has matching id. If no property found, return null.
public void rentProperty(String id, double rent, String rentedBy)	Call **findProperty** using the id. If the property is found, call getSecurityDeposit() and rent(..) method using the found property.
public void leaseOver(String id)	Call ***findProperty*** method. If the property is available, call ***leaseOver*** method of the **Property** class.
public ArrayList<Property> getProperties()	Getter method for ***properties*** attribute.
public void viewAll()	Loop through the ***properties*** attribute and print each property.
public void viewDetails(String id)	Call ***findProperty*** method and print the property if the property is found.

B. **<u>Create another project name "PropertyManagementSystemApp" and do the following.</u>**

1. Add the project reference of "PropertyManagementLib" project.
2. Add **App** class (under **uap.app** package):
    a. Add main method, create an object of **PropertyManager** class and provide menu for each method.
        a. Add Property (Apartment, Store, OfficeSpace) info to the system.
        b. Search property
        c. View by category
        d. Rent a property
        e. Lease Over
        f. Exit