WEEK-3

STRUCTURAL MODELING CLASS DIAGRAMS

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

Common Modeling Techniques for Class Diagram:

1. Modeling Simple Collaborations
2. Modeling a logical database Schema
3. Forward and Reverse Engineering

1)Class Diagram for Intelligent Information Service System of Smart Library

Design and develop a class diagram for a Intelligent Information Service System of Smart Library, which is a software built to handle the primary functions of a library. An innovative system provides intelligent services in the library to both users and the terminal. Compared to the core digital reading room, they can make wise judgments on the retrieval and use of information assets. The implementation of succeeding value management based on the latest technological tools is required for learning to provide knowledge services and fulfil its role as convergence is capable of reacting to varied data needs. The major obstacles to digital libraries are lack of planning and software, import restrictions on equipment, inadequately skilled staff, lack of standards, and a refusal to cooperate. Libraries rely on library Automation systems to manage asset collections as well as relationships with their members. Library Automation systems help libraries keep track of the books and their checkouts, as well as members' subscriptions and profiles.

Library Automation systems also involve maintaining the database for entering new books and recording books that have been borrowed with their respective due dates.

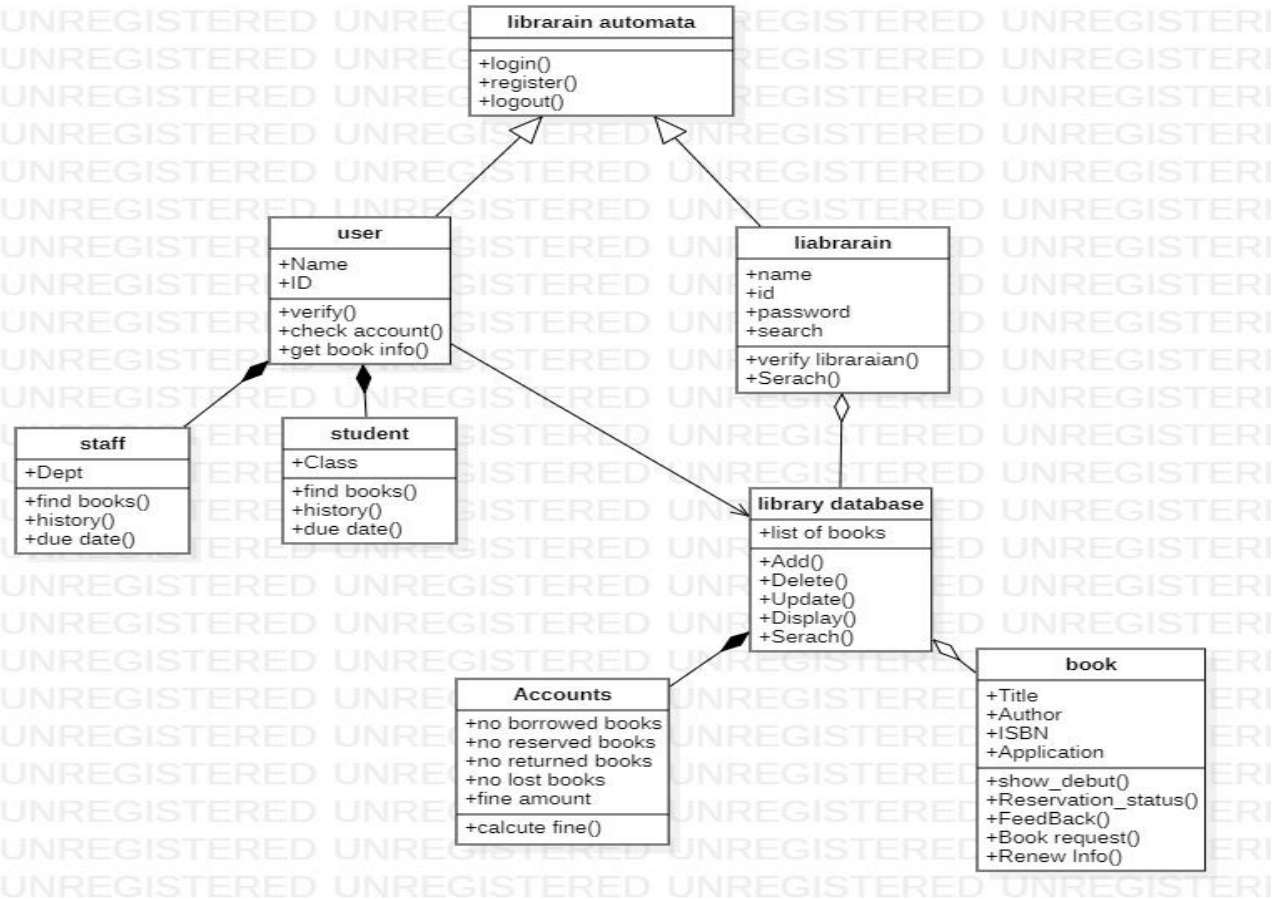
1. Identify the objects and classes
2. Clearly identify what each class is responsible for
3. Identify attributes and methods of each class

4. Identify the suitable relationships among the classes

```
/* Class diagram for Library Automation System */

 1. Classes of Library Automation System:
•Library Automation System class
It manages all operations of Library Automation System. It is central part
of organization for which software is being designed.
•User Class
It manages all operations of user.
•Librarian Class
It manages all operations of Librarian.
•Book Class
It manages all operations of books. It is basic building block of system.
•Account Class
It manages all operations of account.
•Library database Class
It manages all operations of library database.
•Staff Class
It manages all operations of staff.
•Student Class
It manages all operations of student.
2.  Attributes of Library Automation System:
•Library Automation System Attributes
```

```
User Type, Username, Password
•User Attributes
Name, Id
•Librarian Attributes
Name, Id, Password, Search_String
•Book Attributes
Title, Author, ISBN, Publication
•Account Attributes
no_borrowed_books,  no_reserved_books,  no_returned_books,  no_lost_books
fine amount
•Library database Attributes
List_of_books
•Staff Class Attributes
Dept
•Student Class Attributes
Class
Methods of Library Automation System:
•Library Automation System Methods
Login (), Register (), Logout ()
•User Methods –
Verify (), Check Account (), get_book_info ()
•Librarian Methods
Verify librarian (), Search ()
•Book Methods
Show_duedt  (),  Reservation_status  (),  Feedback  (),  Book_request  (),
Renew_info ()
•Account Methods
Calculate fine ()
•Library database Methods
Add (), Delete (), Update (), Display (), Search ()
```



2)Class Diagram for Smart Automatic Teller Machine

Design and develop a class diagram for Smart ATM system, this diagram should provide an effective visual representation of how an automated teller machine operates. An ATM class diagram contains common classes, relationships, associations and components of an ATM, including the customer, card, terminal and bank. This diagram should show how the different components interact within the system to enable customers to withdraw and deposit money. The ATM class diagram should also highlight the role of ATMs in financial transactions, such as transferring funds from one account to another, bill payments and banking operations. This diagram is essential for understanding the functionality of ATMs and how they are used in financial services.

1. Identify the objects and classes
2. Clearly identify what each class is responsible for
3. Identify attributes and methods of each class

4. Identify the suitable relationships among the classes

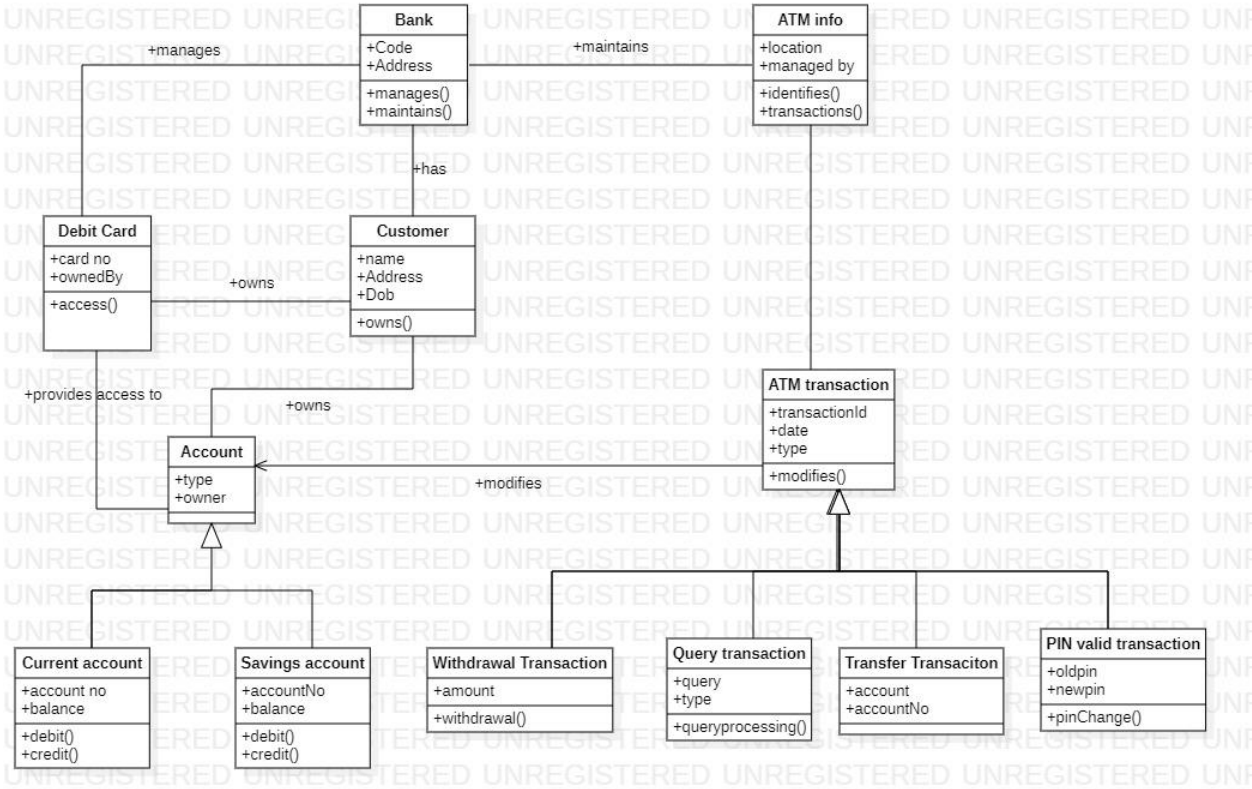
```
/* Class diagram for Smart Automatic Teller Machine */

To design a Class Diagram:
First select an element where a new Class Diagram to be contained as a
child.
Select Model | Add Diagram | Class Diagram in the Menu Bar or select Add
Diagram | Class Diagram in Context Menu.
Name Expression: Edit name expression.
Syntax of Name Expression
expression: = [ '<<' stereotype `>>`] [ visibility] name
stereotype :: = (identifier)
visibility:: = '+' | '#' | '-' | '~'
name:: = (identifier)
Visibility: Change visibility property.
Add Note: Add a linked note.
Add Constraint: Add a constraint.
Add Attribute (Ctrl+Enter): Add an attribute.
Add Operation (Ctrl+Shift+Enter): Add an operation.
Add Reception: Add a reception.

To design a Class:
Select Class in Toolbox.
Drag on the diagram as the size of Class.
```

```
To design a Class (model element only) by Menu:
Select an Element where a new Class to be contained.
Select Model | Add | Class in Menu Bar or Add | Class in Context Menu.
Name Expression: Edit name expression.
Syntax of Name Expression
expression:: = [ '<<' stereotype `>>` ] [ visibility] name
stereotype:: = (identifier)
visibility:: = '+' | '#' | '-' | '~'
name:: = (identifier)
Visibility: Change visibility property.
Add Note: Add a linked note.
Add Constraint: Add a constraint.
Add Attribute (Ctrl+Enter): Add an attribute.
Add Operation (Ctrl+Shift+Enter): Add an operation.
Add Template Parameter: Add a template parameter.
Add Reception: Add a reception.
Add Sub-Class: Add a sub-class.
Add Super-Class: Add a super class.
Add Provided Interface: Add a provided interface.
Add Required Interface: Add a required interface.
Add Associated Class: Add an associated class.
Add Aggregated Class: Add an aggregated class.
Add Composited Class: Add a composited class.
Add Port: Add a port.

Attribute
To add an Attribute:
Select a Classifier.
```



3)Class Diagram for e-caravansary System

Design and develop a class diagram for an e- caravansary System. An e- caravansary System is a software built to handle all online caravansary activities easily and safely. This System will give the hotel caravansary automation power and flexibility to manage the entire system from a single online portal. The system allows the manager to keep track of all the available rooms in the system as well as to book rooms and generate bills.

1. Identify the objects and classes
2. Clearly identify what each class is responsible for
3. Identify attributes and methods of each class
4. Identify the suitable relationships among the classes

```
/* Class diagram for e-caravansary System */

The classes used in this system are,
•Hotel Automation: This class depicts the entire hotel and says whether
the hotel is opened or closed.
```

•Employees: It contains the details of the Employee. There are two kinds of employees, Server and the chef. This employee class is the parent class of two subclass – Server and Chef
•Server: It contains the details of the server, the table to which they are assigned, the order which is currently serving, etc.
•Chef: It contains the details of the chef working on a particular order.
•Customer: It contains the details of the customer.
•Table: It contains the table details like table number and the server who are assigned to that table.
•Menu: Menu contains all the food items available in the restaurant, their availability, prize, etc.
•Order: Order depicts the order associated with a particular table and the customer.
•Bill: Bill is calculated using the order and menu.
•Payment: This class is for doing payment. The payment can be done in two ways either cash or card. So, payment is the parent class and cash and card are subclasses.
•Cash: Payment can be done by cash
•Card: Payment can be done by card or online
Attributes:
Hotel Automation – HotelName, NumberOfEmployees
•Employees – EmployeeId, EmployeeName, EmployeeSalary
•Server – ServerId, OrderId
•Chef – Chef_Id, OrderId
•Customer – CustomerId, CustomerName, Bill_Id, OrderId, PaymentId
•Table – TableNumber, OccupiedStatus, ServerId, CustomerId
•Menu – ItemId, ItemName, Amount
•Order – OrderId, ItemId, ItemName, Quantity, CustomerId, ServerId
•Bill – Bill_Id, OrderId, TotalBill
•Payment – PaymentId, Bill_Id

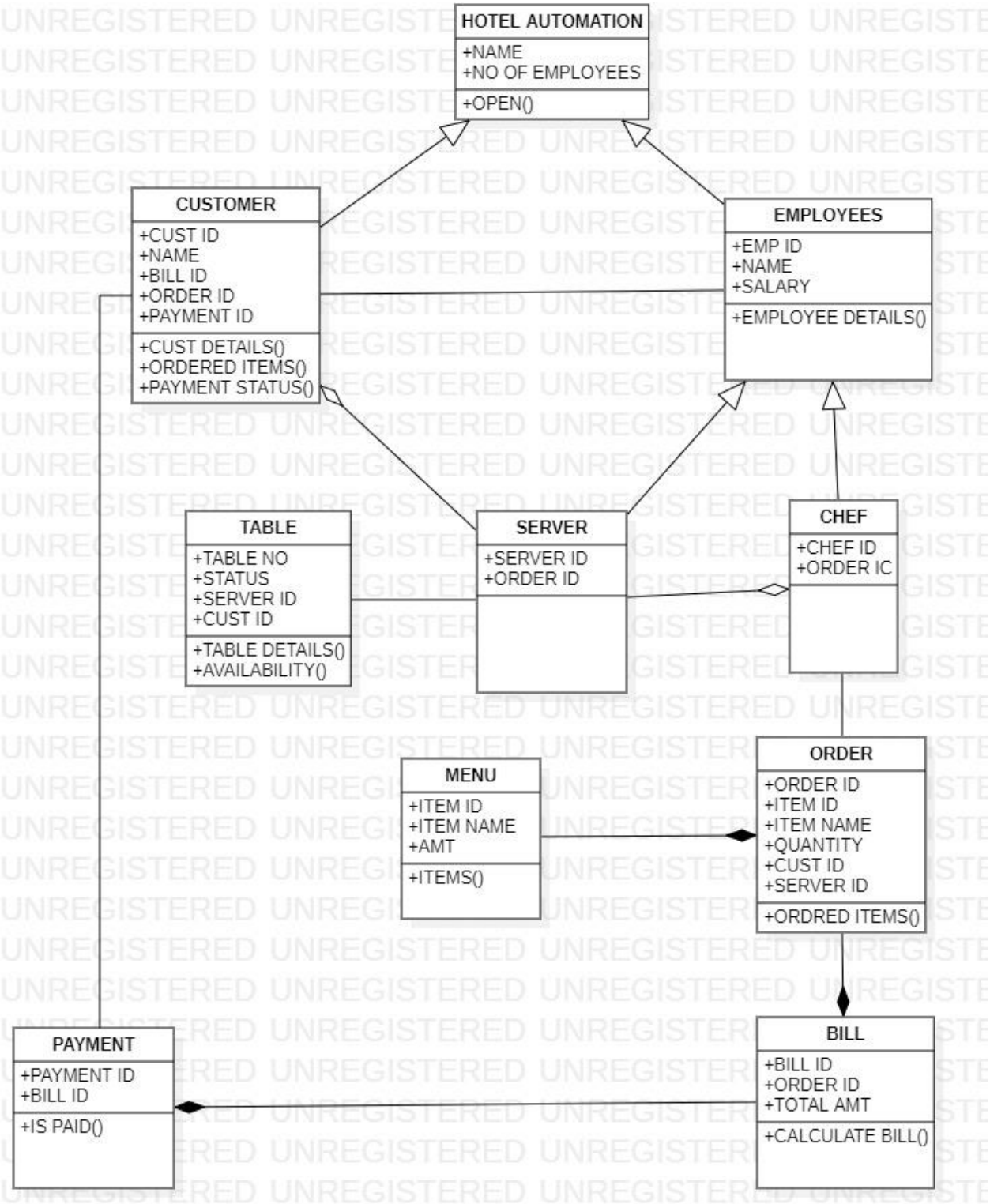
Methods:
1. Hotel Automation:
•open () -This is used to indicate if the hotel is functioning or not.
2. Employees:
•employee details () – This method contains the details of the employee.
3. Customer:
•customer_details () – This depicts the details of the customer.
•ordered_items () – This method contains the items which are ordered by the customer.
•payment_status () -This says whether the customer paid or not.
4. Table:
•table_details () – This method contains the details of the table along with the customer and no of seats.
•availability status () – This method says whether the table is occupied or not.
5. Menu:
•items () – This method displays the menu items, their availability and their price.
6. Order:
•order_items () – This method orders the items selected by the user from the menu.
7. Bill:
•calculate bill () – This method calculates the bill for a particular table.

8. Payment:
•ispaid () – It shows whether payment is successful or not.
Relationships:
Inheritance:
Here, Employee is parent class Server and Chef are child classes because server is a employee and chef is a employee.

Association:
Here,
•Employee and customer
•Server and table
•Customer and payment
•Chef and order
follows association relationships.

Composition:
Here,
•Menu and Order
•Order and Bill
•Bill and Payment
follows composition relation
Order cannot exist without Menu; Bill cannot exist without order and payment cannot exist without bill. So here order is contained inside the menu, bill is contained inside the order and payment is contained inside the bill.

Aggregation:
Here,
•Customer and Server
•Chef and Server
follows Aggregation relation
Server is associated with the customer but can exist without the customer as well, Likewise Chef is associated with the server but can exist without the server as well.


HOTEL AUTOMATION
+NAME
+NO OF EMPLOYEES
+OPEN()
CUSTOMER
+CUST ID
+NAME
+BILL ID
+ORDER ID
+PAYMENT ID
+CUST DETAILS()
+ORDERED ITEMS()
+PAYMENT STATUS()
EMPLOYEES
+EMP ID
+NAME
+SALARY
+EMPLOYEE DETAILS()
TABLE
+TABLE NO
+STATUS
+SERVER ID
+CUST ID
+TABLE DETAILS()
+AVAILABILITY()
SERVER
+SERVER ID
+ORDER ID
CHEF
+CHEF ID
+ORDER IC
MENU
+ITEM ID
+ITEM NAME
+AMT
+ITEMS()
ORDER
+ORDER ID
+ITEM ID
+ITEM NAME
+QUANTITY
+CUST ID
+SERVER ID
+ORDRED ITEMS()
PAYMENT
+PAYMENT ID
+BILL ID
+IS PAID()
BILL
+BILL ID
+ORDER ID
+TOTAL AMT
+CALCULATE BILL()

4)Class Diagram for Automated Financial Services

Design and develop a class diagram for an Automated Financial Services. Automated Financial Services is a web-based tool that allows the Bureau of the Fiscal Service to pay financial institutions for services rendered. BMS also has analytical tools that may be used to examine and approve pay, budgets, and outflows. Along with this, the main goal of bank Automation is to make sure that all of the different parts of a bank work together in a way that makes the most money possible. Besides, the system allows customers to design accounts, deposit and withdraw money from their accounts, and examine reports for all of their accounts.

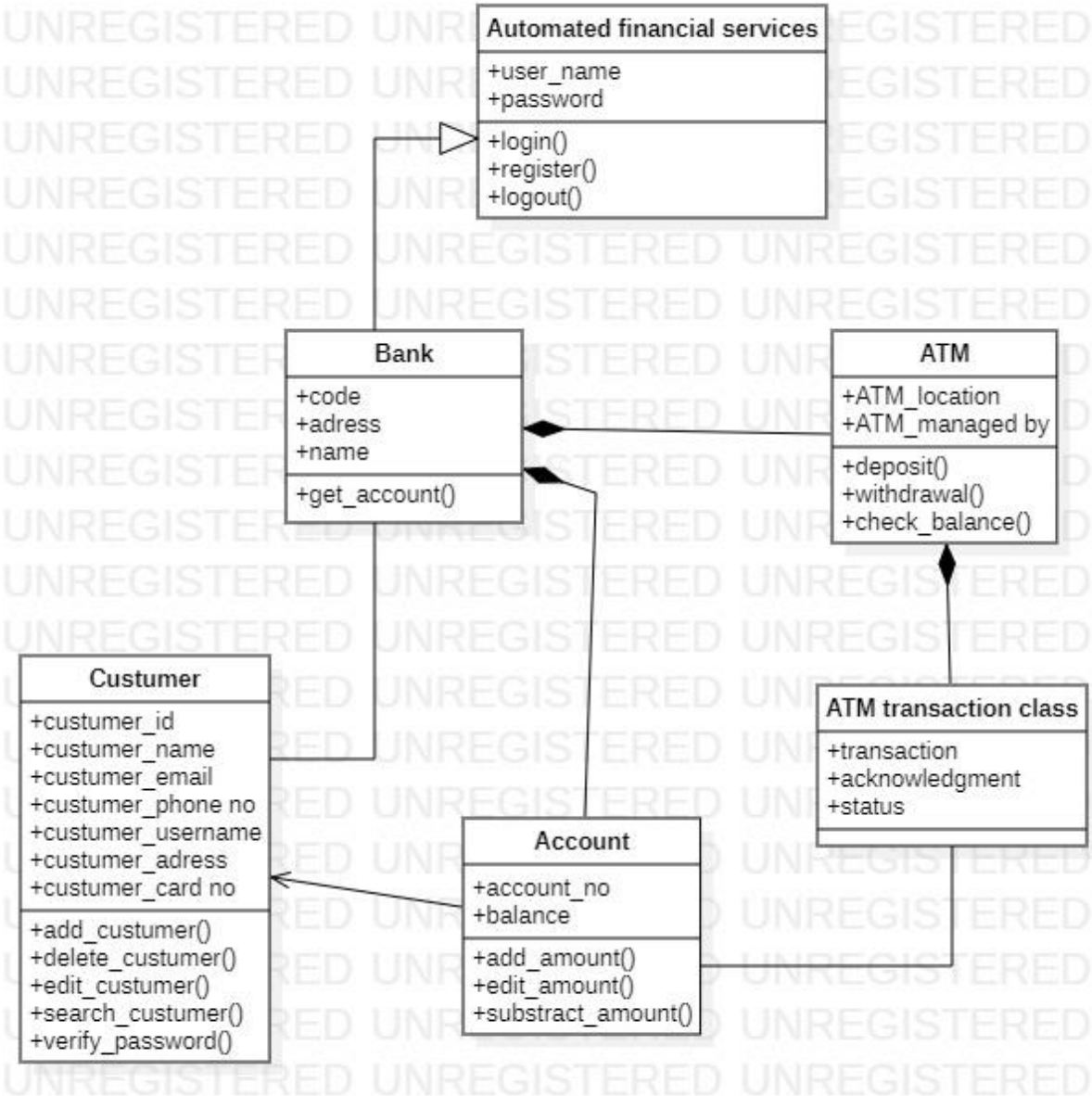
1. Identify the objects and classes
2. Clearly identify what each class is responsible for
3. Identify attributes and methods of each class
4. Identify the suitable relationships among the classes

```
/* Class diagram for Automated Financial Services */
List of Classes
1.Customer
2.Bank
3.ATM
4.Account
5.ATM transaction
class (customer)

Attributes/Variables of the class (customer)
1.customer_ password: varchar
Public Attributes/Variables
There are following public attributes in the mentioned class diagram.
1.+customer_ id: int
2.+customer_ name: string
3.+customer_ email: string
4.+customer_ phone No: string
1.+customer_ username: string
2.+customer_ address: string
3.+customer_ card no: int
Functions of the class (customer)
```

functions
1.+add_ customer ()
2.+delete_ customer ()
3.+edit_ customer ()
4.+search_ customer ()
5.+verify_ password ()
class (Bank)
Attributes/Variables of the class (Bank)
Private Attributes/Variables:
We can assign private attributes/ values to this class, but suppose that currently we are not willing to make the attributes private.

Public Attributes/Variables:
public attributes.
1. +code: int
2.+address: int
3.+name: string
Functions of the class (bank)
There are following functions in the mentioned class diagram.
+get_ Account ()
class (ATM)
Attributes/Variables of the class (ATM)
Private Attributes/Variables:
•+ATM_ location: int
•+ATM_ managed by int
Functions of the class (ATM)
There are the following functions in the mentioned class diagram.
•+deposit ()
•+withdraw ()
•+check_ balance ()
class (Account)


Automated financial services
+user_name
+password
+login()
+register()
+logout()
Bank
+code
+adress
+name
+get_account()
ATM
+ATM_location
+ATM_managed by
+deposit()
+withdrawal()
+check_balance()
Custumer
+custumer_id
+custumer_name
+custumer_email
+custumer_phone no
+custumer_username
+custumer_adress
+custumer_card no
+add_custumer()
+delete_custumer()
+edit_custumer()
+search_custumer()
+verify_password()
Account
+account_no
+balance
+add_amount()
+edit_amount()
+substract_amount()
ATM transaction class
+transaction
+acknowledgment
+status