BS SOFTWARE ENGINEERING

4TH SEMESTER

SOFTWARE DESIGN & ARCHITECTURE

PROJECT REPORT

BANK MANAGEMENT SYSTEM

SUBMITTED BY:

RABIA LATIF	20-SE-024
M. AWAIS	20-SE-001
SYED HAIDER ZUBAIR	20-SE-002
USMAN ALI SHAH	20-SE-016
M USMAN	20-SE-018

SUBMITTED TO:

SIR JUNIAD ALI KHAN

DEPARTMENT OF SOFTWARE ENGINEERING

HITEC UNIVERSITY TAXILA CANTT

2022

BANK MANAGEMENT SYSTEM

SOFTWARE DESIGN AND ARCHITECTURE

STUDENTS OF HITEC UNIVERSITY TAXILA CANTT

BS SOFTWARE ENGINEERING | 4 TH SEMESTER

TABLE OF CONTENTS

ABSTRACT

The Bank Management System is an application for maintaining a person's account in a bank. This research work is based on **BANK MANAGEMENT SYSTEM** in which customers can access more accurate, quicker and rapid banking services from the bank management system. In this project we tried to show the working of a banking system and cover the basic functionality of the Bank Management System. Through bank management system, a client can acquire his record and manage numerous exchanges utilizing his cell phone or personal computer.

The main aim of this project is to develop software for Bank Management System. This project has been developed to carry out the processes easily and quickly, which is not possible with the manuals systems, which are overcome by this software. This system will help to solve problems by making transactions easier through mobile or any computerized system & many more. Thus, features of this project will save transaction time and therefore increase the efficiency of the system.

CHAPTER 1

INTRODUCTION

PURPOSE

Purpose of making this project is making our semester project "**Software Design and Architecture**". The main aim of this project is that we have to choose a system of our own choice and display all its architectural styles and also do its implementation in **java** programming language and also do its software modeling .

INTRODUCTION

Bank management system is use of computers and telecommunications to enable banking transactions to be done by telephone or computer rather than through human interaction. Its features include creating account, checking balance, deposit money , withdraw money and view transaction history . Electronic banking has vastly reduced the physical transfer of paper money from one place to another or even from one person to another.

In banking systems all work is done virtually rather than manually. It helps the user to work easily from home and it takes less time. Now a day's, managing a bank is tedious job up to certain limit. So software that reduces the work is essential. Also today's world is a genuine computer world and is getting faster and faster day-by-day. Thus, considering above necessities, the software for bank management system has become necessary which would be useful in managing the bank more efficiently.

This project is being designed on basis of pre-requisite courses: Object Oriented Programming, Software Engineering, Software

Requirement Engineering, Programming Fundamentals. In this project we have used the concepts of functions, classes, loops, conditional statements, inheritance, switch cases, abstraction, encapsulation etc.

IMPORTANCE OF E-BANKING

- Offer flexibility with online banking.
- Availability of banking services 24/7.
- Delivering bank services to the door-step of customer.
- Less reduction of cash during online transaction

GOALS & OBJECTIVES

In this project our motto is , we are going to make software which manages all the banking work virtually and help the user to take all services from home. Also check all the transactions details and other problems faced by customers.

CHAPTER 2

SYSTEM FUNCTIONALITIES

SYSTEM FUNCTIONALITIES

- Create account
- Balance enquiry
- Deposit money
- Withdraw money
- View transaction history

ID	Functionalities	Prioritization	User Group
1	Create Account	MUST	Customer, Bank
2	Balance Enquiry	MUST	Customer, Bank
3	Deposit_money	MUST	Customer, Bank
4	Withdraw money	MUST	Customer, Bank
5	View_transaction_history	COULD	Customer, Bank

NON-FUNCTIONAL REQUIREMENTS

AREA	REQUIREMENT
Security	1. The user shall be authenticated before granting access. 2. All sensitive data held in database shall be encrypted. 3. Data stored within the database shall follow Data Protection Act 1998.
Efficiency	4. Latency shall be under 5 seconds across all internet connections. 5. The system shall be able to run smoothly with 1,000 concurrent users.
Portability	6. The system shall be accessible from desktops and mobile phones.
Availability	7. The system shall be available 99.8% of the time during the year. 8. The system shall be restored within 2 hours after a crash.
Reliability	9. The system shall be able to process 10,000 requests within 12 hours. 10. The copy of the database shall be kept on a separate server. 11. The system shall have a ROCOF of 0.00001.
Usability	12. The system shall use consistent layout on all pages. 13. The system shall be accessible by people with disabilities.
Scalability	14. New system functionality shall be added within current architecture/code.

CHAPTER 3

ARCHITECTURAL STYLES

"Service oriented architecture" of Bank Management System

DEFINTION

A service is software and hardware independent, stand-alone, scalable, reusable, self-contained, distributed, loosely coupled and standardised solution delivering a particular functionality. A service is made-up ofservice implementation and service interface, and can consist of other services.

Service-Oriented Architecture or SOA is an architectural approach for designing and developing a web application. In this approach, an application uses services available over the network via communication calls or requests.

SOA includes a set of design principles that structure system development while providing means for integrating components into a unified and decentralized architecture.

It allows developers to merge a considerable amount of facilities from existing services to the application.

In SOA, resources are packages as "services", that are self-contained and well-defined modules that offer standard business functionality and are independent of the context or state of other services.

"Layered architecture" of Bank Management System

There are five layers in layered architecture:

- User interaction layer
- Functionality layer
- Business rules layer
- Application core layer
- Database layer

User Interaction layer

The menus and buttons used for create account, transactions & every other action performing through clicking held in this layer. It is the most visible layer of the application we are displaying.

Functionality layer

In this layer the functions of a system performed like we check the activity of login, transactions held by user, balance inquiry, and all the functions perform in this layer.

Business rule layer

The business rule of banking system is security & availability. In this we assure that our system is secure & safe and all the requirements are available 24/7.

Application layer

All the codes are there in this layer. The code that we used to implement this system is in application layer.

Database layer

In this layer all the transactions, balance inquiries and all the functions that are performing will update, delete, & retrieve.

Conclusions

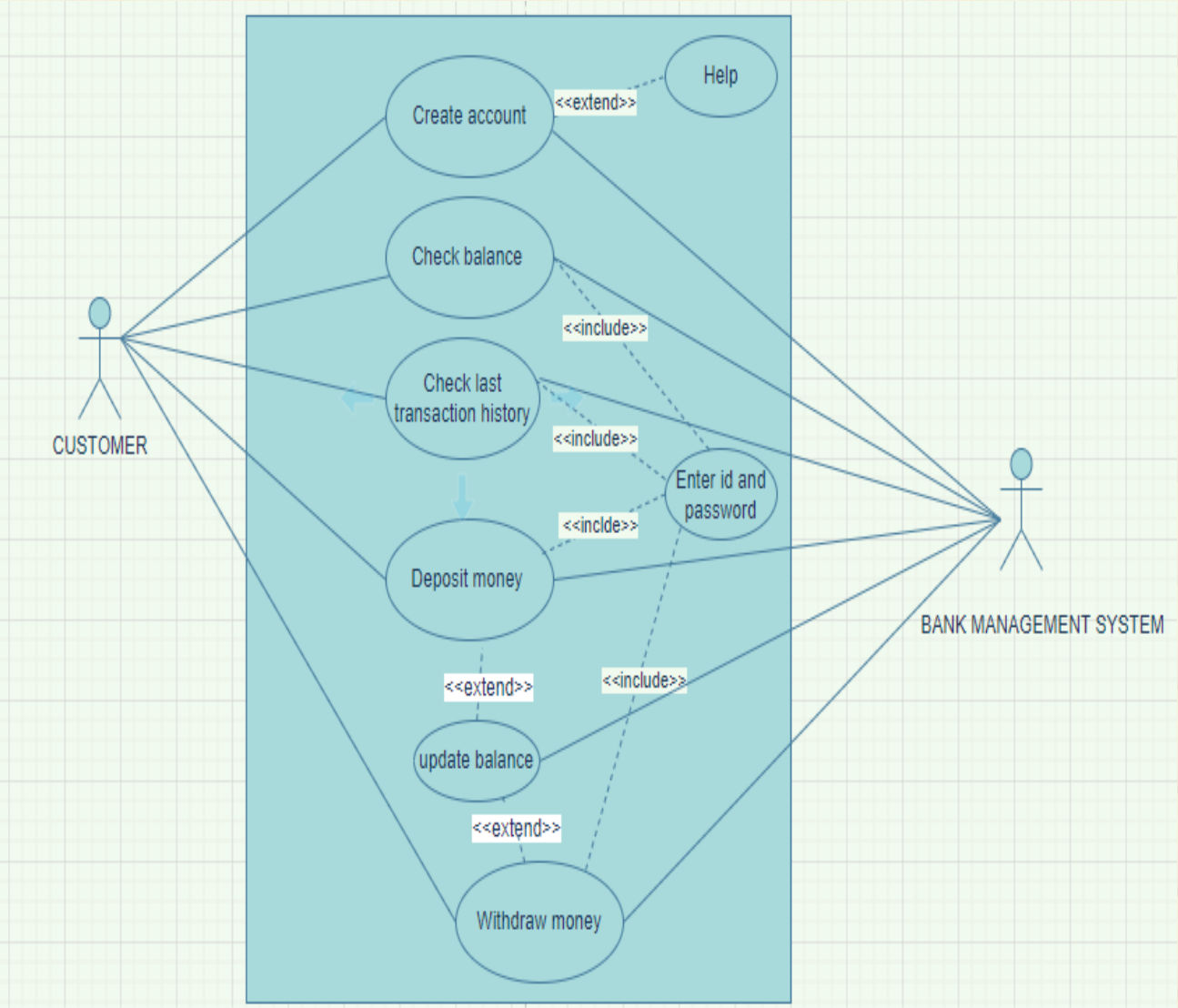
Having reviewed each architectural style it became apparent that the two most suitable styles for this project are Service-Oriented and Object-Oriented. These two styles match majority of the client requirements as well as being widely-used approaches today, indicating that the deployed system will be futureproof. Being so popular presently, the maintenance of the system will be carried-out easily by the most software engineers.

The layered architectural style was abandoned due to weak performance. As mentioned earlier, the layered approach requires more time to process data when compared to SOA and OO. The component-based architectural style will be taken forward and implemented as a SOA. The OO architectural style was chosen because it is easily scalable due to polymorphism, reusable due to inheritance and fully testable using most available techniques. Similarly, SOA is scalable, reusable, loosely coupled and independent making it an ideal competitor to OO. Both architectural styles bring more advantages to the projects than disadvantages, making them ideal solutions to the problem.

CHAPTER 4

UML MODELING

USE CASE DIAGRAM



DESCRIPTION

The relationship between actors & use cases are:

Customer

It includes:

- Create account
- Check balance
- Deposit money
- Withdraw money
- View Transaction History
- Help

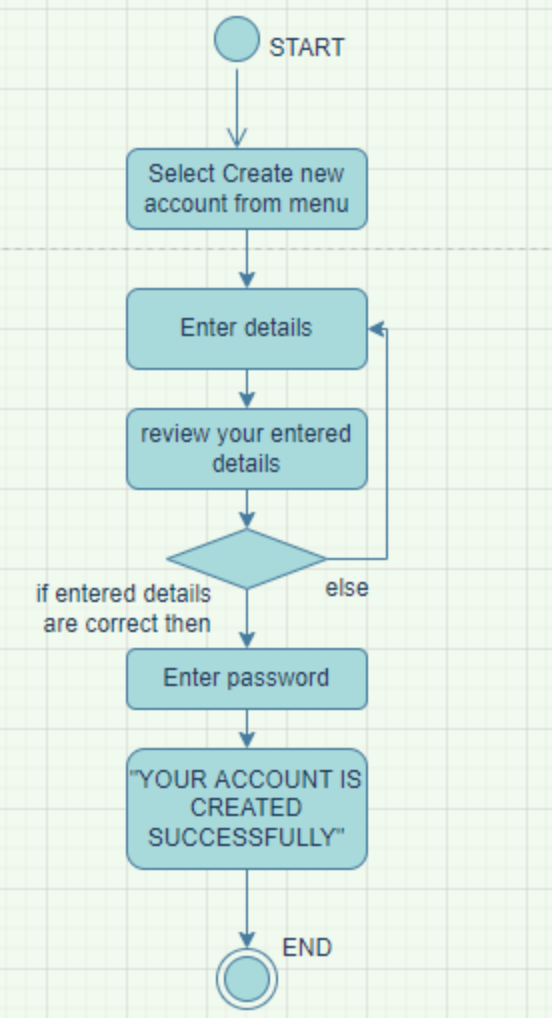
Bank

It includes:

- Create account
- Check balance
- Enter id and password
- Deposit money
- Update balance
- Withdraw money
- View Transaction History

ACTIVITY DIAGRAM

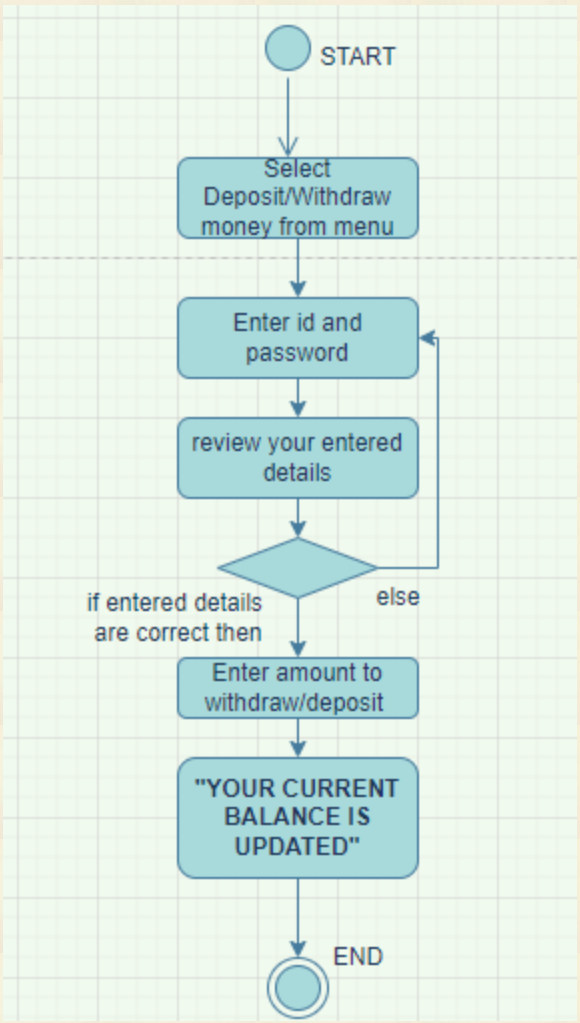
ACTIVITY DIAGRAM FOR CREATE ACCOUNT



DESCRIPTION OF CREATE ACCOUNT

In this customer need to enter all his credentials in order to create account and after that he/she rechecks his credentials that either they are correct if they are correct then he must submit it and hence his account will be created. And the process will terminate.

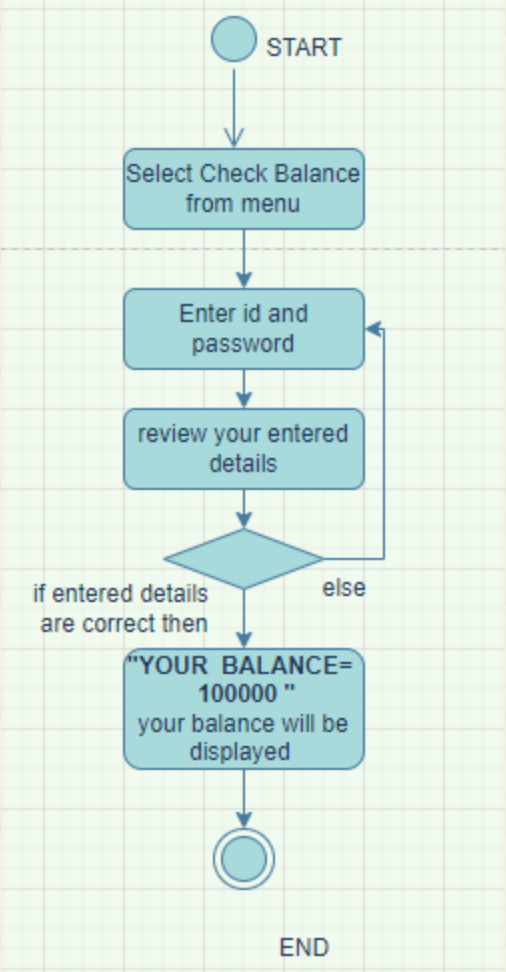
ACTIVITY DIAGRAM FOR DEPOSIT /WITHDRAW MONEY



DESCRIPTION OF DEPOSIT/ WITHDRAW MONEY

For this customer need to enter all his credentials in order to fist login to his account and after that he/she can deposit /withdraw money. If he/she has entered the correct password then he can deposit/withdraw money and then his updated balance will be displayed on screen . And the process will terminate.

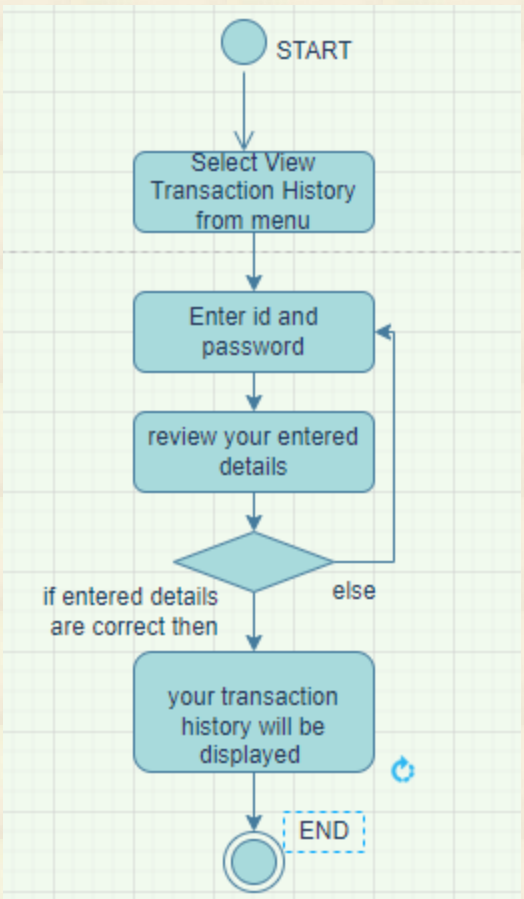
ACTIVITY DIAGRAM FOR CHECK BALANCE



DESCRIPTION OF CHECK BALANCE

For this customer need to enter all his credentials in order to fist login to his account and after that he/she can check balance. If he/she has entered the correct password then his account statement would be displayed. And the process will terminate.

ACTIVITY DIAGRAM FOR TRANSACTION HISTORY



DESCRIPTION OF VIEW TRANSACTION HISTORY

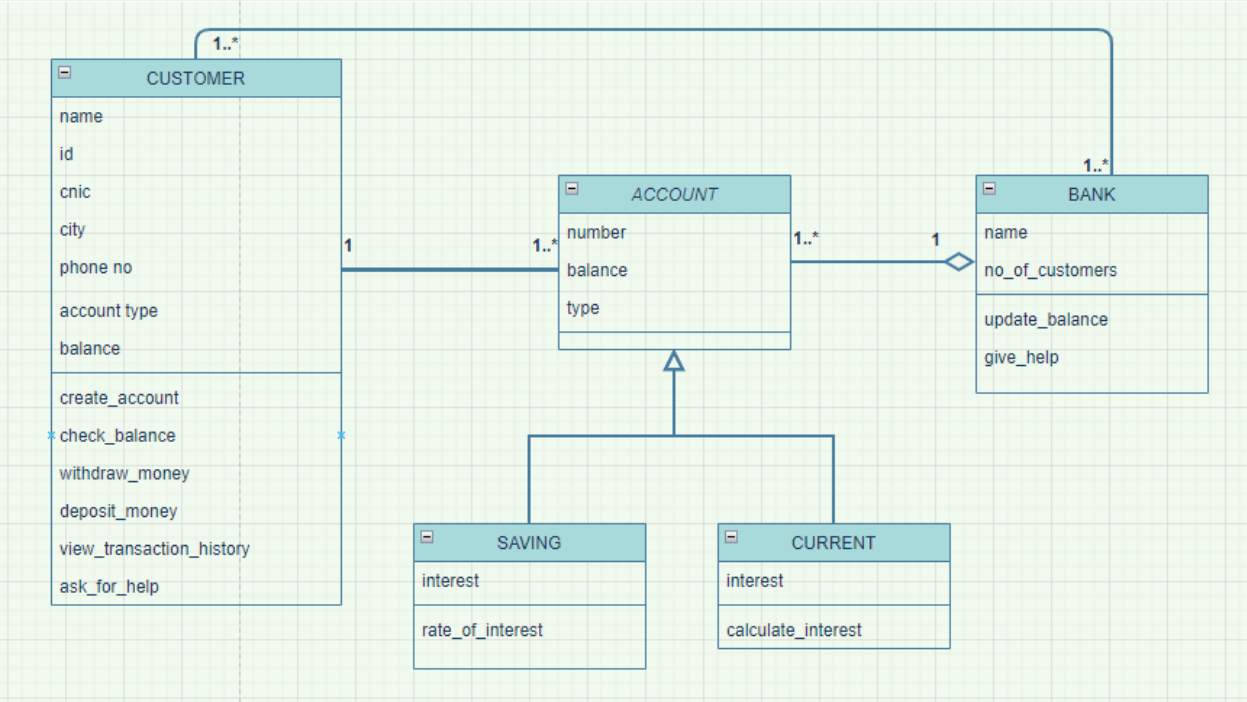
For this customer need to enter all his credentials in order to fist login to his account and after that he/she can view transaction history. If he/she has entered the correct password then his transaction history would be displayed. And the process will terminate.

CLASS DIAGRAM

CLASS DIAGRAM

DESCRIPTION
Classes
Bank , customer , account
Attributes
Customer - name, id, cnic, phone no, account_type, balance, city

Bank- name,no_of_customers **Account -**number, balance, type
Methods
Customer- check_balance, create account, deposit money, withdraw money, view_transaction_history and ask_for_help **Bank-** update_balance and give_help **Account-** saving(rate_of_interest), current(calculate_interest)
Inheritance/Generalization
Account- saving,current
Aggregation
Account- Bank
Association
Customer- Account



DESCRIPTION

Classes

- BANK

Bank class has attributes name,no_of_customers and member functions update_balance and give_help.

- CUSTOMER

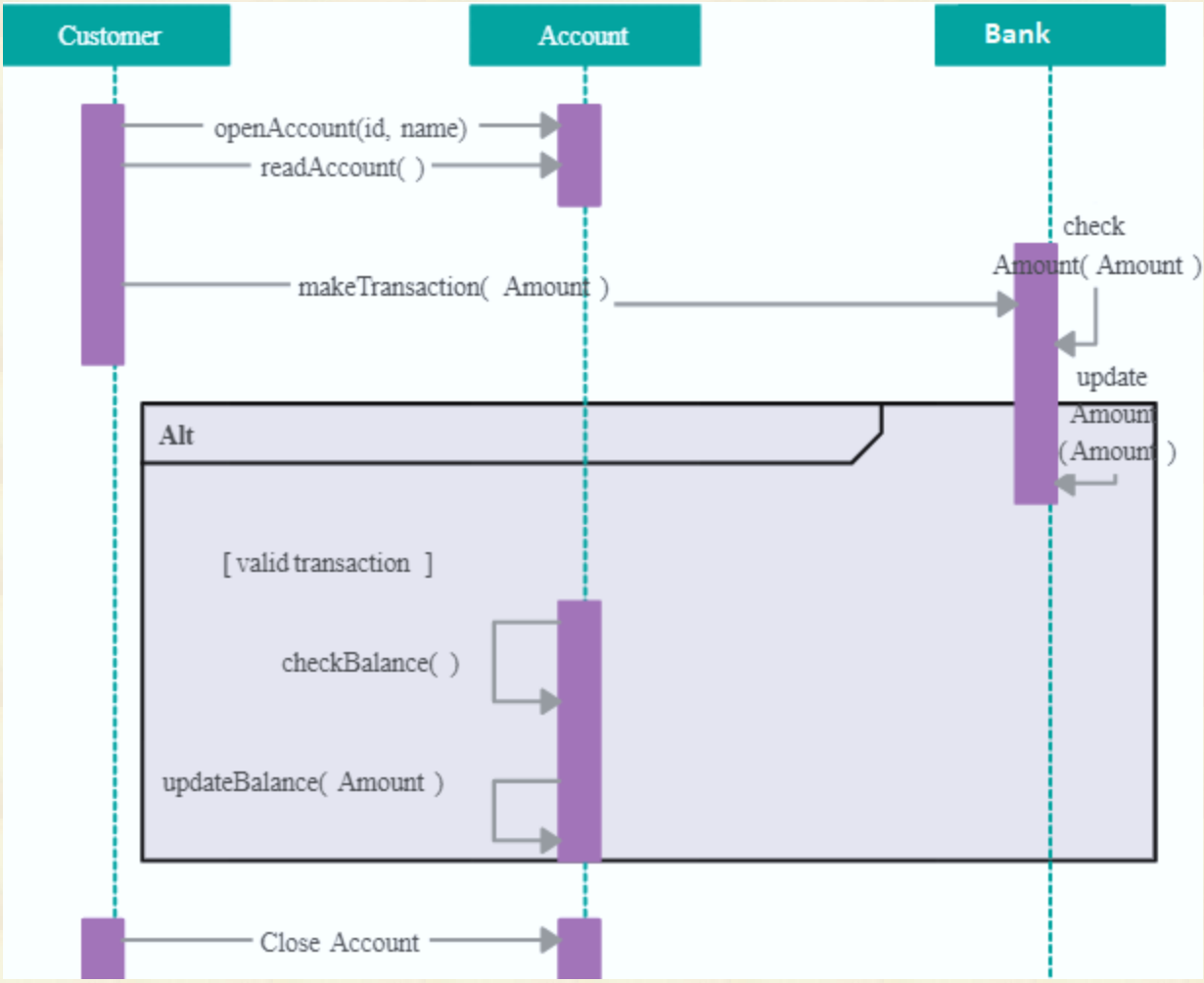
Customer class has attributes name, id , cnic , phone no , account_type ,balance ,city and member functions check_balance, create account, deposit money, withdraw money , view_transaction_history and ask_for_help.

- ACCOUNT
  1. SAVING
  2. CURRENT

  Account class has attributes number, balance, type. Account is a generalized class having two subclasses having and current.

SEQUENCE DIAGRAM

SEQUENCE DIAGRAM



SEQUENCE DIAGRAM FOR BANK MANAGEMENT SYSTEM

CHAPTER 5

SDA PRINCIPLES

SDA principles

The following are the SDA principles:

- Problem partitioning
- Abstraction

- Modularity
- Top down & bottom up strategy

Problem partitioning

All the problems can be solved at a time but it is difficult to solve all the problems so we divide it into smaller portion like a lot of transactions are being held at one time it is difficult to see whole data at a time so we break it into smaller chunks so that we can easily see & find our required transaction.

These chunks are not independent to each other as they are part of one whole system.

Benefits of problem partitioning

- Software is easy to understand
- Software becomes simple
- Software is easy to test
- Software is easy to modify
- Software is easy to maintain
- Software is easy to expand

ABSTRACTION

Abstraction is process of hiding certain details and showing only essential information to user. For example, in banking system user will only see their own transactions and balance but the CEO or the bank management system will see all the details of every account. So only essential data that the user want to see is provided to user and hide other details.

Functional Abstraction

In functional abstraction, details of the algorithms to accomplish the function are not visible to the consumer of the function. For example, the system and user

perform their own functions in banking management system but admin see the details of the user but user only see their details not system.

Data Abstraction

Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. For example, we only provide the details of user to it not all the details that are performing in backend.

MODULARITY

Modularity specifies to the division of software into separate modules which are differently named and addressed and are integrated later on in to obtain the completely functional software. It is the only property that allows a program to be intellectually manageable. Single large programs are difficult to understand and read due to a large number of reference variables, control paths, global variables, etc.

For example, the system is BANK MANAGEMENT SYSTEM. The modules are:

- Create account
- Check balance
- Withdraw money
- Deposit money
- View transaction history

Top Down & Bottom Up strategy

Top down

This approach starts with the identification of the main components and then decomposing them into their more detailed sub-components.
For example, account is decomposed into saving account &current account etc.

Bottom up

A bottom-up approach begins with the lower details and moves towards up the hierarchy. For example, saving account & current account are the subclasses of class account.

CHAPTER 6

IMPLEMENTATION

CODE

SOFTWARE USED

ONLINE COMPILER

https://www.programiz.com/java-programming/online-compiler/

CODE IN JAVA

```java
import java.util.*;
class Bank
{
    public static void main(String args[])
    {
        Account a=new Account();
        a.menu();
    }

}
class Account
{
    int balance=0,trans_detail;
    String cust_name;
    String cust_id;
    String cnic;
    String phone_no;
    String city;
    String account_type;

    void deposit(int amount)
    {
        if(amount>0)
        {
            balance=balance+amount;
            trans_detail=amount;
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("DEPOSITED SUCCESSFULLY!!");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
```

```java
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }
        else
        {

            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("AMOUNT MUST BE ABOVE 0");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }
    }

    void withdraw(int amount)
    {
        if(amount>0)
        {
            balance=balance-amount;
            trans_detail=-amount;
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("WITHDRAW SUCCESSFULLY!!");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }
        else
        {
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("MOUNT MUST BE ABOVE 0");
```

```java
                System.out.println("MOUNT MUST BE ABOVE 0");
                System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            }
    }

    void bank_trans_detail()
    {
        if(trans_detail>0)
        {
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("DEPOSITED MONEY:::"+trans_detail);
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }
        else if(trans_detail<0)
        {
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("DEPOSITED AMOUNT :::"+Math.abs(trans_detail));
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }
        else
        {
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("NO TRANSACTION HAS BEEN DONE!!");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
```

```java
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }
    }

    void menu()
    {
        int ch;
        Scanner s =new Scanner(System.in);
        System.out.println("*******************************************");
        System.out.println("*******************************************");
        System.out.println("*******************************************");
        System.out.println("*******************************************");
        System.out.println("~~~~WELCOME TO BANK MANAGEMENT SYSTEM ~~~~");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        System.out.println("~~~~~~~~~~~~~~~GROUP MEMBERS~~~~~~~~~~~~~~");
        System.out.println("~~~~~~~~~~RABIA LATIF 20-SE-024~~~~~~~~~~~");
        System.out.println("~~~~~~~~~~~M. AWAIS  20-SE-001~~~~~~~~~~~~");
        System.out.println("~~~~~SYED HAIDER ZUBAIR 20-SE-002~~~~~~~~~");
        System.out.println("~~~~~~~~USMAN ALI SHAH 20-SE-016~~~~~~~~~~");
        System.out.println("~~~~~~~~~~~~M. USMAN 20-SE-018~~~~~~~~~~~~");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        System.out.println("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        System.out.println("~~~~ SOFTWARE DESIGN AND ARCHITECTURE ~~~~");
        System.out.println("*******************************************");
        System.out.println("*******************************************");
        System.out.println("*******************************************");
        System.out.println("*******************************************");
        do
        {
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
```

```java
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("\n``````````````````MAIN MENU```````````````");
            System.out.println("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("1].CREATE ACCOUNT");
            System.out.println("2].BALANCE ENQUIRY");
            System.out.println("3].DEPOSIT MONEY");
            System.out.println("4].WITHDRAW MONEY");
            System.out.println("5].LAST TRANSACTION HISTORY");
            System.out.println("6].EXIT");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
            System.out.println("ENTER YOUR CHOICE:::");
            ch=s.nextInt();

            switch(ch)
            {
                case 1:
                    System.out.println("ENTER YOUR  NAME::: ");
                    cust_name=s.next();
                    System.out.println("ENTER CUSTOMER ID::: ");
                    cust_id=s.next();
                    System.out.println("ENTER CNIC NUMBER::: ");
                    cnic=s.next();
                    System.out.println("ENTER YOUR PHONE NUMBER::: ");
                    phone_no=s.next();
                    System.out.println("ENTER YOUR CITY::: ");
                    city=s.next();
                    System.out.println("ENTER YOUR ACCOUNT TYPE::: ");
                    account_type=s.next();
```

```
                    account_type=s.next();
                    System.out.println("\nYOUR ACCOUNT HAS SUCCESSFULLY BEEN CREATED");
                    System.out.println("\nYOUR ACCONUT DETAILS ARE:");
                    System.out.println("\nNAME::: "+cust_name);
                    System.out.println("\nID::: "+cust_id);
                    System.out.println("\nCNIC NUMBER::: "+cnic);
                    System.out.println("\nPHONE NUMBER::: "+phone_no);
                    System.out.println("\nCITY::: "+city);
                    System.out.println("\nACCOUNT TYPE ::: "+account_type);
                    System.out.println("\nTHANK YOU FOR CREATING ACCOUNT");
                    break;
                case 2:

                    System.out.println("ACCOUNT BALANCE::: "+balance);

                    System.out.println("\n");
                    break;
                case 3:

                    System.out.println("ENTER THE AMOUNT TO DEPOSIT::: ");
                    int amount1=s.nextInt();
                    deposit(amount1);
                    System.out.println("\n");
                    break;
                case 4:

                    System.out.println("AVAILABLE BALANCE IS::: "+balance);

                    System.out.println("ENTER THE AMOUNT YOU WANT TO WITHDRAW:::");
                    int amount2=s.nextInt();
                    withdraw(amount2);
```

```
                    withdraw(amount2);
                    System.out.println("\n");
                    break;
                case 5:
                    bank_trans_detail();
                    System.out.println("\n");
                    break;
                case 6:
                    System.out.println("########################################");
                    break;
                default:
                    System.out.println("INVALID CHOICE!!! PLEASE ENTER VALID CHOICE ");
                    System.out.println("\n");
            }
        }while(ch!=5);
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        System.out.println("~~~~~~~~THANKS FOR VISITING OUR BANK~~~~~~~");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
    }
}
```

INPUT

```
import java.util.*;
```

```
class Bank
{
	public static void main(String args[])
	{
		Account a=new Account();
		a.menu();
	}

}
class Account
{
	int balance=0,trans_detail;
	String cust_name;
	String cust_id;
	String cnic;
	String phone_no;
	String city;
	String account_type;

	void deposit(int amount)
	{
		if(amount>0)
		{
			balance=balance+amount;
			trans_detail=amount;

	System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

	System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
			System.out.println("DEPOSITED SUCCESSFULLY!!");

	System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

	System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
		}
		else
		{


	System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

	System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
			System.out.println("AMOUNT MUST BE ABOVE 0");
```

```
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                }
        }

        void withdraw(int amount)
        {
                if(amount>0)
                {
                        balance=balance-amount;
                        trans_detail=-amount;

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("WITHDRAW SUCCESSFULLY!!");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                }
                else
                {

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("MOUNT MUST BE ABOVE 0");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                }
        }

        void bank_trans_detail()
        {
                if(trans_detail>0)
                {

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
```

```
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("DEPOSITED MONEY:::"+trans_detail);

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                }
                else if(trans_detail<0)
                {

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("DEPOSITED AMOUNT
:::"+Math.abs(trans_detail));

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                }
                else
                {

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("NO TRANSACTION HAS BEEN DONE!!");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                }
        }

        void menu()
        {
                int ch;
                Scanner s =new Scanner(System.in);
                System.out.println("****************************************");
                System.out.println("****************************************");
                System.out.println("****************************************");
                System.out.println("****************************************");
```

```
                System.out.println("~~~~WELCOME TO BANK MANAGEMENT
SYSTEM ~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                System.out.println("~~~~~~~~~~~~~~~~GROUP
MEMBERS~~~~~~~~~~~~~~~");
                System.out.println("~~~~~~~~~~~RABIA LATIF 20-SE-024~~~~~~~~~~~");
                System.out.println("~~~~~~~~~~~M. AWAIS  20-SE-001~~~~~~~~~~~~");
                System.out.println("~~~~~SYED HAIDER ZUBAIR 20-SE-
002~~~~~~~~~");
                System.out.println("~~~~~~~~USMAN ALI SHAH 20-SE-
016~~~~~~~~~~");
                System.out.println("~~~~~~~~~~~~M. USMAN 20-SE-
018~~~~~~~~~~~~");
            System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                System.out.println("~~~~ SOFTWARE DESIGN AND ARCHITECTURE
~~~~");
                System.out.println("****************************************");
                System.out.println("****************************************");
                System.out.println("****************************************");
                System.out.println("****************************************");
                do
                {

System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("\n``````````````````MAIN
MENU``````````````");

        System.out.println("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("1].CREATE ACCOUNT");
                        System.out.println("2].BALANCE ENQUIRY");
                        System.out.println("3].DEPOSIT MONEY");
                        System.out.println("4].WITHDRAW MONEY");
                        System.out.println("5].LAST TRANSACTION HISTORY");
                        System.out.println("6].EXIT");
```

```
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                        System.out.println("ENTER YOUR CHOICE:::");
                        ch=s.nextInt();

                        switch(ch)
                        {
                           case 1:
                              System.out.println("ENTER YOUR  NAME::: ");
                        cust_name=s.next();
                        System.out.println("ENTER CUSTOMER ID::: ");
                    cust_id=s.next();
                    System.out.println("ENTER CNIC NUMBER::: ");
                        cnic=s.next();
                        System.out.println("ENTER YOUR PHONE NUMBER::: ");
                    phone_no=s.next();
                    System.out.println("ENTER YOUR CITY::: ");
                        city=s.next();
                        System.out.println("ENTER YOUR ACCOUNT TYPE::: ");
                    account_type=s.next();
                                System.out.println("\nYOUR ACCOUNT HAS
SUCCESSFULLY BEEN CREATED");
                                System.out.println("\nYOUR ACCONUT DETAILS ARE:");
                                System.out.println("\nNAME::: "+cust_name);
                                System.out.println("\nID::: "+cust_id);
                                System.out.println("\nCNIC NUMBER::: "+cnic);
                                System.out.println("\nPHONE NUMBER::: "+phone_no);
                                System.out.println("\nCITY::: "+city);
                                System.out.println("\nACCOUNT TYPE ::: "+account_type);
                                System.out.println("\nTHANK YOU FOR CREATING
ACCOUNT");
                        break;
                                case 2:

                                        System.out.println("ACCOUNT BALANCE:::
"+balance);

                                        System.out.println("\n");
                                        break;
                                case 3:
```

```
                                        System.out.println("ENTER THE AMOUNT TO
DEPOSIT::: ");
                                        int amount1=s.nextInt();
                                        deposit(amount1);
                                        System.out.println("\n");
                                        break;
                                case 4:

                                        System.out.println("AVAILABLE BALANCE IS:::
"+balance);

                                        System.out.println("ENTER THE AMOUNT YOU
WANT TO WITHDRAW:::");
                                        int amount2=s.nextInt();
                                        withdraw(amount2);
                                        System.out.println("\n");
                                        break;
                                case 5:

                                        System.out.println("YOUR TRANSACTION
HISTORY IS:::");
                                       bank_trans_detail();
                                        System.out.println("\n");
                                        break;
                                case 6:

        System.out.println("########################################");
                                        break;
                                default:
                                        System.out.println("INVALID CHOICE!!! PLEASE
ENTER VALID CHOICE ");
                                        System.out.println("\n");
                        }
                }while(ch!=5);

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

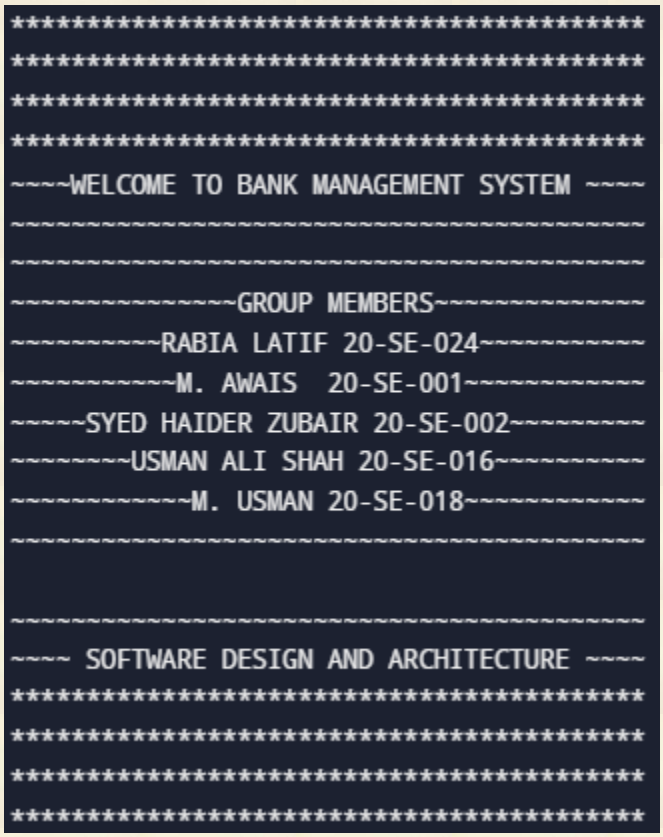
        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
                System.out.println("~~~~~~~~THANKS FOR VISITING OUR
BANK~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");

        System.out.println("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~");
```

```
        }
}
```

OUTPUT OF THE CODE



MAIN MENU

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

`````````````````MAIN MENU```````````````

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1].CREATE ACCOUNT
2].BALANCE ENQUIRY
3].DEPOSIT MONEY4].WITHDRAW MONEY
5].LAST TRANSACTION HISTORY
6].EXIT
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ENTER YOUR CHOICE:::
```

CREATE ACCOUNT

```
ENTER YOUR CHOICE:::
1
ENTER YOUR  NAME:::
RABIA
ENTER CUSTOMER ID::: 24
ENTER CNIC NUMBER:::
3830238668686
ENTER YOUR PHONE NUMBER:::
0300000000
ENTER YOUR CITY:::
TAXILA
ENTER YOUR ACCOUNT TYPE:::
SAVING
YOUR ACCOUNT HAS SUCCESSFULLY BEEN CREATED

YOUR ACCONUT DETAILS ARE:
NAME::: RABIA
ID::: 24
CNIC NUMBER::: 3830238668686
PHONE NUMBER::: 0300000000
CITY::: TAXILA
ACCOUNT TYPE ::: SAVING
THANK YOU FOR CREATING ACCOUNT
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

BALANCE ENQUIRY
```

```
ENTER YOUR CHOICE:::
2
ACCOUNT BALANCE::: 20000
```

DEPOSIT MONEY

```
ENTER YOUR CHOICE:::
3
ENTER THE AMOUNT TO DEPOSIT:::
20000
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
DEPOSITED SUCCESSFULLY!!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

WITHDRAW MONEY

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ENTER YOUR CHOICE:::
4
AVAILABLE BALANCE IS::: 20000
ENTER THE AMOUNT YOU WANT TO WITHDRAW:::
5000
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
WITHDRAW SUCCESSFULLY!!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

VIEW TRANSACTION HISTORY

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ENTER YOUR CHOICE:::
5
YOUR TRANSACTION HISTORY IS:::
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
DEPOSITED MONEY:::20000
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

EXIT

```
ENTER YOUR CHOICE:::
6
########################################
```

DEFAULT SWITCH CASE

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ENTER YOUR CHOICE:::
7
INVALID CHOICE!!! PLEASE ENTER VALID CHOICE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~THANKS FOR VISITING OUR BANK~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

CHAPTER 7

CONCLUSIONS

CONCLUSIONS

This project has been designed using the principles of **OOP, SE, SRE, PF and SDA.**

The major programming concepts used in this project are:

- Classes
- Objects
- Functions
- Encapsulation
- Abstraction
- Inheritance
- Switch case statements
- Conditional statements
- Loops

REFRENCES

- www.w3schools.com/java/java_abstract.asp
- www.javatpoint.com/software-engineering-software-design-principles
- www.freeprojectz.com/uml-diagram/internet-banking-system-uml-diagram
- www.dbforums.com
- www.ibm.com

- Chihi, H., Chainbi, W., Ghdira, K. (2016) 'Cloud computing architecture and migration strategy for universities and higher education', Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, vol.2016-. DOI: 10.1109/AICCSA.2015.7507140.

- Crotty, J., Horrocks, I. (2017) 'Managing legacy system costs: A case study of a meta-assessment model to identify solutions in a large financial services company', Applied Computing and Informatics, 13(2), pp.175-183. DOI: 10.1016/j.aci.2016.12.001.

GLOSSARY

TERMS TO UNDERSTAND

TERM	DEFINITION
Class	a blueprint of data member and functions or we can say a template which has attributes and functions
Object	Instance of class
Abstraction	used to hide background details or any unnecessary implementation about the data so that users only see the required information
Inheritance	the process by which genetic information is passed on from parent to child i.e, from base class to derived class
Association	a "using" relationship between two or more objects in which the objects have their own lifetime and there is no owner
Generalization	the technique of extracting the essential characteristics (these include attributes, properties and methods) from two or more subclasses and then combining them inside a generalized base class (also called a superclass)
Aggregation	a relation that exists between two or more two objects which individually have their own individual life cycle along with the ownership
Encapsulation	the bundling of data, along with the methods that operate on that data, into a single unit , data hiding

ABBREVIATIONS

ABBREVIATION	STANDS FOR
OOP	OBJECT ORIENTED PROGRAMMING
SDA	SOFTWARE DESIGN ND ARCHITECTURE
PF	PROGRAMMING FUNDAMENTALS
SRE	SOFTWARE REQUIREMENT ENGINEERING
SE	SOFTWARE ENGINEERING

****