**LAB # 11**

**Abstract Data Types**

**OBJECTIVE:**

Understanding and implementing abstract data types.

**LAB TASKS:**

Considering a bank interface, it will have different concrete classes for receiving bill payments, opening of new accounts and contacting those loan takers whose limits of extension have been expired. Now, develop this scenario on the described ADT and decide which ADT should be used for each concrete class, also implement the whole scenario in multiple java classes.

**Solution:**

**Source Code:**

**public interface Bill {**

public void addBillPayer(Account acc);

public void removeBillPayer(Account acc);

public void executeBillPay();

}

**public interface Loan {**

public void addLoanTaker(Account acc);

public void removeLoanTaker(Account acc);

public void showAllLoaners();

}

**public interface OpenNewAccount {**

Account openNewAccount(String name,int accNo,int amount);

}

**public class Account {**

private String name;

private int accNo;

private int amount;

public Account(String name, int accNo,int amount) {

this.amount=amount;

this.name = name;

this.accNo = accNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAccNo() {

return accNo;

}

public void setAccNo(int accNo) {

this.accNo = accNo;

}

public void deposit(int amount) {

this.amount+=amount;

}

public int withdraw(int amount) {

if(amount>this.amount) {

return -1;

}

else {

int amt = this.amount-amount;

return amt;

}}}

import java.util.ArrayList;

import java.util.List;

**public class Bank implements OpenNewAccount{**

private List<Account> accounts;

public Bank() {

accounts = new ArrayList<Account>();

}

@Override

public Account openNewAccount(String name,int accNo,int amount) {

Account ac = new Account(name,accNo,amount);

this.accounts.add(ac);

return ac;

}}

import java.util.LinkedList;

import java.util.Queue;

**public class LoanTaker implements Loan{**

private Queue<Account> loaners;

public LoanTaker() {

loaners = new LinkedList<Account>();

}

@Override

public void addLoanTaker(Account acc) {

this.loaners.add(acc);

}

@Override

public void removeLoanTaker(Account acc) {

this.loaners.remove(acc);

}

@Override

public void showAllLoaners() {

System.out.println("\nShowing all Loaners:");

int size = loaners.size();

for(int i=0;i<size;i++) {

Account acc=loaners.poll();

System.out.println("Account "+(i+1)+":\nName: "+acc.getName()+"\nAccount No: "+acc.getAccNo()+"\n");

}}}

import java.util.LinkedList;

import java.util.Queue;

**public class PayBill implements Bill{**

private Queue<Account> payers;

public PayBill() {

payers = new LinkedList<Account>();

}

@Override

public void addBillPayer(Account acc) {

payers.add(acc);

}

@Override

public void removeBillPayer(Account acc) {

payers.remove(acc);

}

@Override

public void executeBillPay() {

int size=payers.size();

for(int i=0;i<size;i++) {

Account acc=payers.poll();

System.out.println(acc.getName()+" has payed his bills.");

removeBillPayer(acc);

}}}

**public class Main {**

public static void main(String[] args) {

Bank bank = new Bank();

Account account1 =bank.openNewAccount("Asif", 233, 10000);

Account account2 =bank.openNewAccount("Ahsan", 236, 100000);

Account account3 =bank.openNewAccount("Ahmed", 237, 20000);

LoanTaker loan = new LoanTaker();

loan.addLoanTaker(account3);

loan.addLoanTaker(account2);

PayBill paybill = new PayBill();

paybill.addBillPayer(account1);

paybill.addBillPayer(account2);

paybill.addBillPayer(account3);

paybill.executeBillPay();

loan.showAllLoaners();

}}

**Output:**

A screenshot of a computer screen

Description automatically generated