Programming Logic and Design

CA-PLDES

Project

Student Name: SHILPA SATHYA NARAYANA

Student Number: 674365769

Instructor:

Date: 27 February 2023

Results:

Total: ________/100

Programming Logic and Design – Project

Automatic Teller Simulator

Introduction

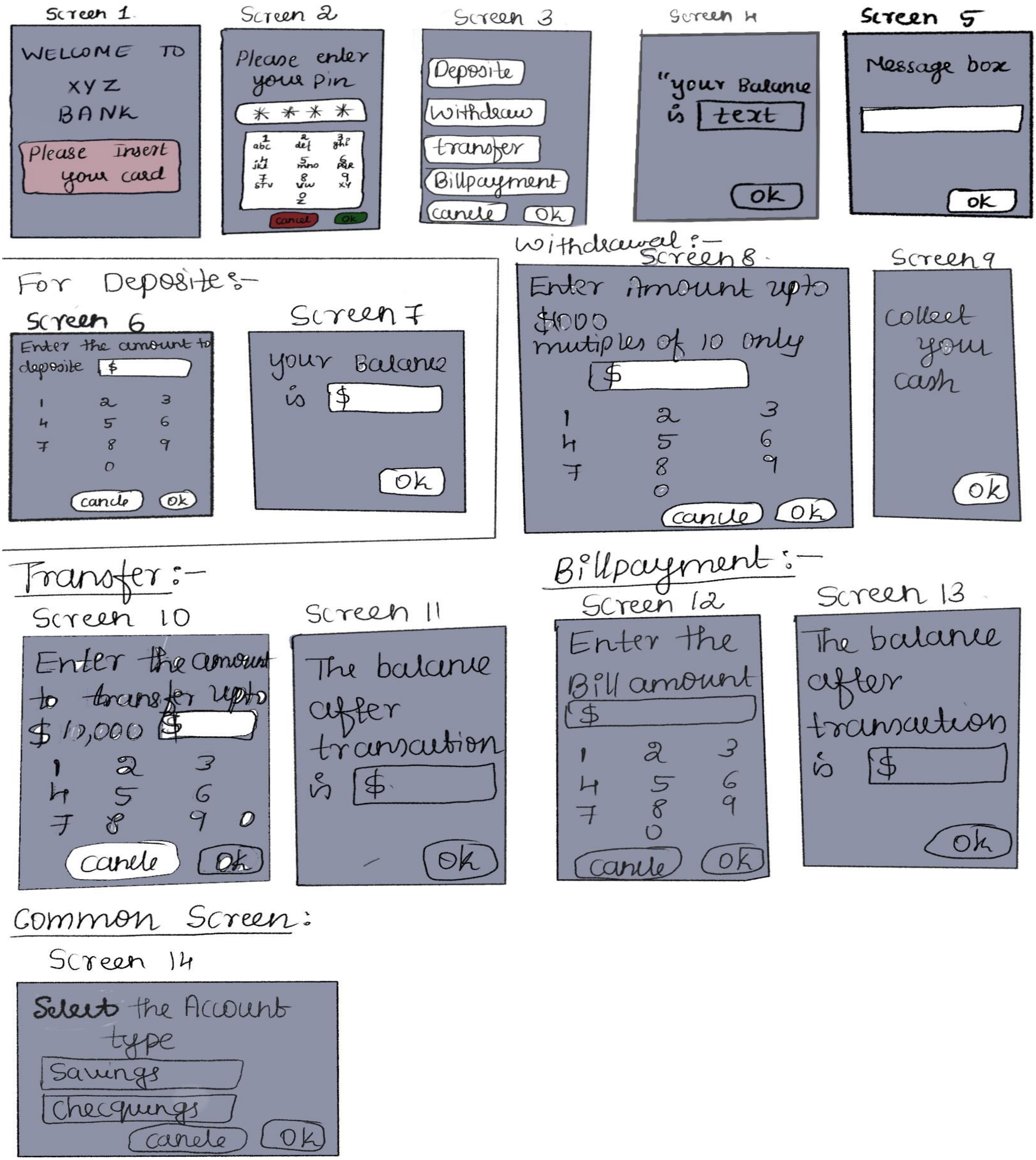
This project will allow you to apply your knowledge and skill in program logic and design. You will need to read and analyze the information provided and extract the vital pieces of information that will allow you to design the program logic and features of the solution to the problem presented. For the purposes of this project, you will need to leverage the major components, features techniques and procedures that you learned in this course in order to complete the requirements of this project. You will need to create flowcharts, write pseudocode, prepare diagrams, apply modularization techniques, use the common programming structures of sequence, selection and looping and design the mainline logic as part of your final solution.

Objectives

The main objectives of this project are to:

- Interpret specifications and analysis performed.
- Design a solution based on the requirements and specifications.
- Design the logic required for a complete program design solution.

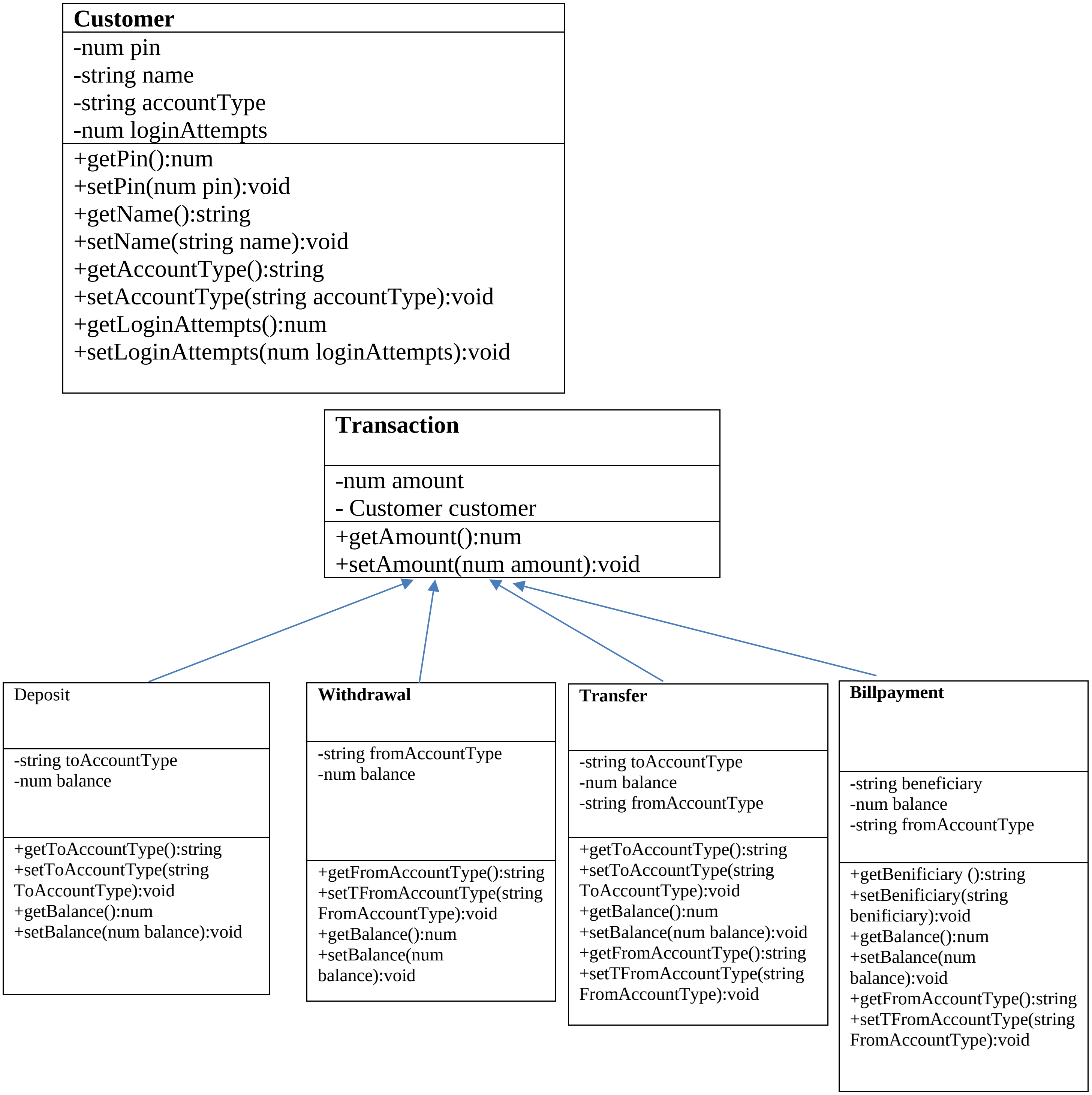
GUI Design:


Screen 1
WELCOME TO
XYZ
BANK
Please Insert your card
Screen 2
Please enter your Pin
* * * *
1 abc
2 def
3 ghi
4 jkl
5 mno
6 pqr
7 stv
8 vw
9 xy
0 z
cancel
ok
Screen 3
Deposite
Withdraw
transfer
Billpayment
canele
ok
Screen 4
"your Balance is
text
ok
Screen 5
Message box
ok
For Deposite:-
Screen 6
Enter the amount to deposite
$
1 2 3
4 5 6
7 8 9
0
canele
ok
Screen 7
your Balance is
$
ok
Withdrawal:-
Screen 8
Enter Amount upto $1000
mutiples of 10 only
$
1 2 3
4 5 6
7 8 9
0
canele
ok
Screen 9
collect you cash
ok
Transfer:-
Screen 10
Enter the amount to transfer upto $10,000
$
1 2 3
4 5 6
7 8 9 0
canele
ok
Screen 11
The balance after transaction is
$
ok
Billpayment:-
Screen 12
Enter the Bill amount
$
1 2 3
4 5 6
7 8 9
0
canele
ok
Screen 13
The balance after transaction is
$
ok
Common Screen:
Screen 14
Select the Account type
Savings
Checquings
canele
ok

Object Dictionary:

Object Type	Name	Screen Num	Variable Affected	Script
Label	welcomeMessage	1	none	none
Lable	message	1	none	none
Lable	message	2	none	none
TextInput	pin	2	customer.pin	none
Button	okButton	2	none	LoginScreen.showMainMenu()
Button	cancleButton	2	none	none
Button Array	transactionTypes	3	none	Deposit(), Transfer(), Withdraw(), Billpayment()
Button	OkButton	3	none	none
Button	CancleButton	3	none	none
Lable	balanceLabel	4	none	none
Button	okButton	4	none	none
Lable	messageLabel	5	none	none
Button	okButton	5	None	None
Label	messageLabel	6	none	none
TextInput	amount	6	Transaction.amount	none
Button	okButton	6	none	Transaction.perform()
Keypad	keypad	6	none	none
Button	cancelButton	6	none	none
Radio	savingsAccount	14	Deposit.fromAccountType Withdraw.fromAccountType Tranfer.toAccountType Transfer.fromAccountType Billpayment.toAccountType Billpayment.fromAccountType	none
Radio	chequingAccount	14	Deposit.fromAccountType Withdraw.fromAccountType Tranfer.toAccountType Transfer.fromAccountType Billpayment.toAccountType Billpayment.fromAccountType	none
Button	CancleButton	14	none	
Button	okButton	14	none	Deposit.setFromAccountType Withdraw.setFromAccountType Tranfer.setToAccountType Transfer.setFromAccountType Billpayment.setToAccountType Billpayment.setFromAccountType

Class Diagram



Customer

-num pin
-string name
-string accountType
-num loginAttempts

+getPin():num
+setPin(num pin):void
+getName():string
+setName(string name):void
+getAccountType():string
+setAccountType(string accountType):void
+getLoginAttempts():num
+setLoginAttempts(num loginAttempts):void

Transaction

-num amount
- Customer customer

+getAmount():num
+setAmount(num amount):void

Deposit

-string toAccountType
-num balance

+getToAccountType():string
+setToAccountType(string ToAccountType):void
+getBalance():num
+setBalance(num balance):void

Withdrawal

-string fromAccountType
-num balance

+getFromAccountType():string
+setTFromAccountType(string FromAccountType):void
+getBalance():num
+setBalance(num balance):void

Transfer

-string toAccountType
-num balance
-string fromAccountType

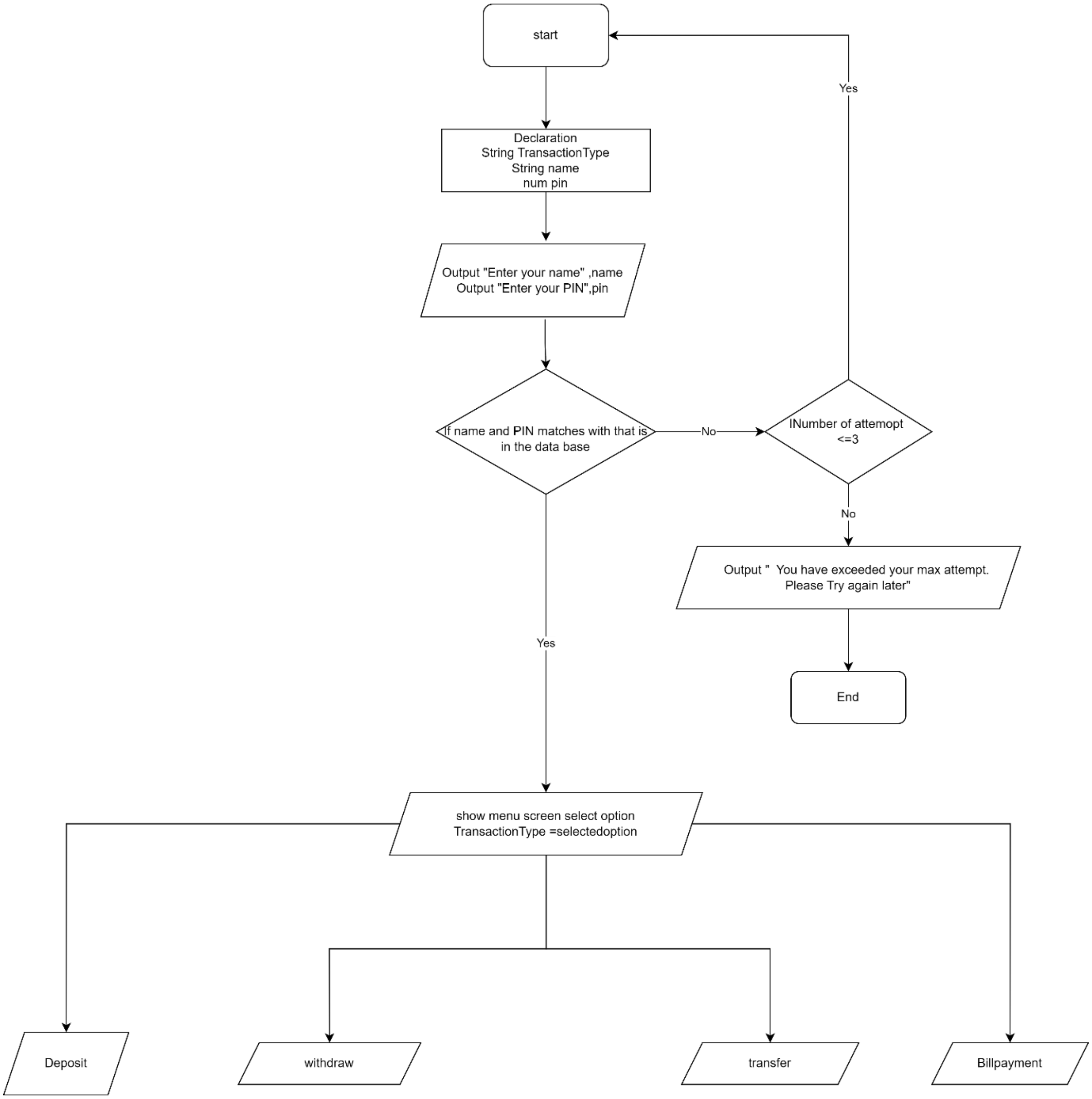
+getToAccountType():string
+setToAccountType(string ToAccountType):void
+getBalance():num
+setBalance(num balance):void
+getFromAccountType():string
+setTFromAccountType(string FromAccountType):void

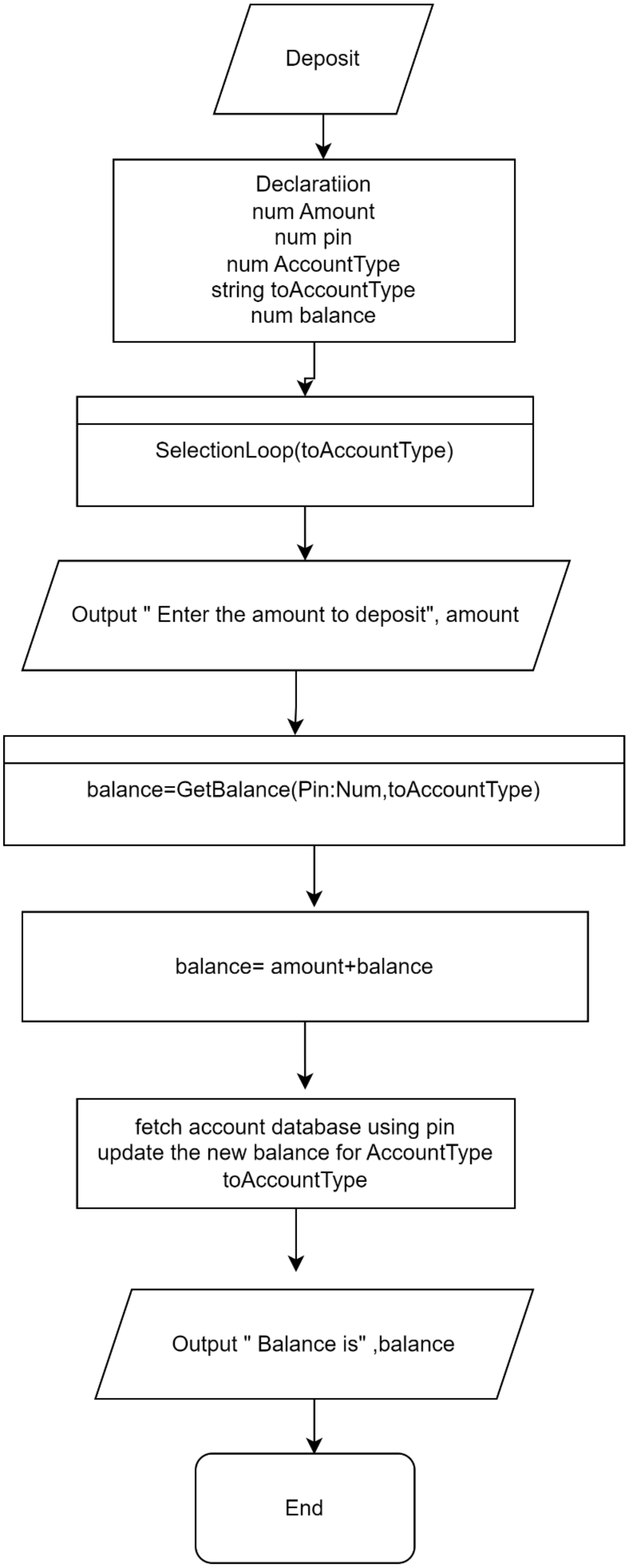
Billpayment

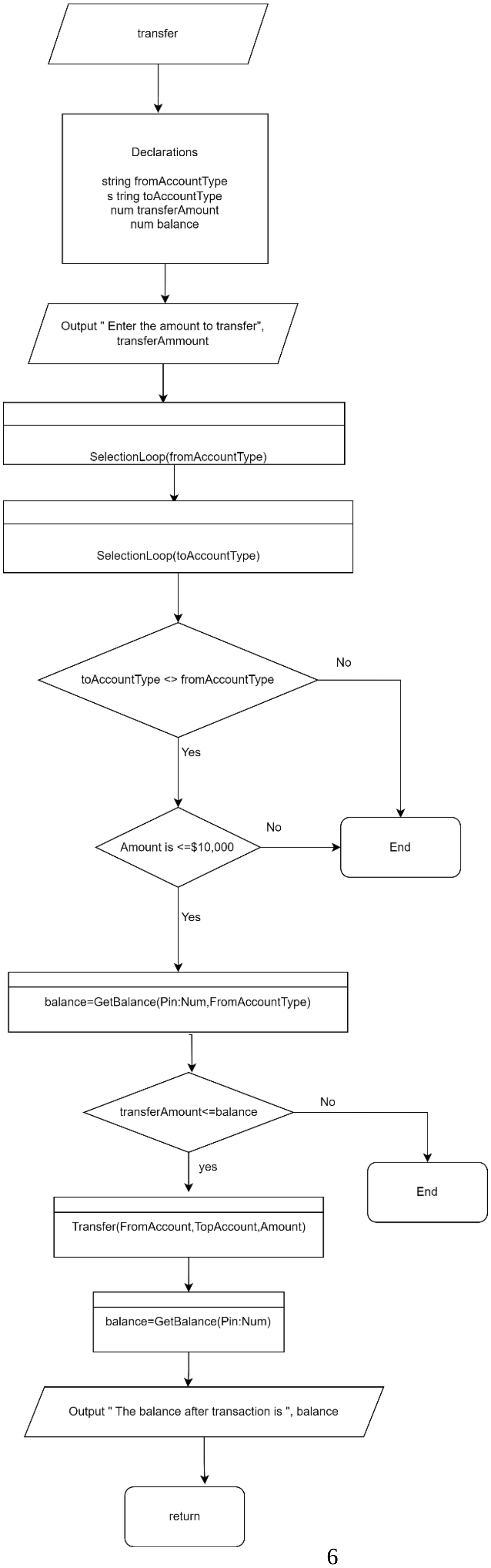
-string beneficiary
-num balance
-string fromAccountType

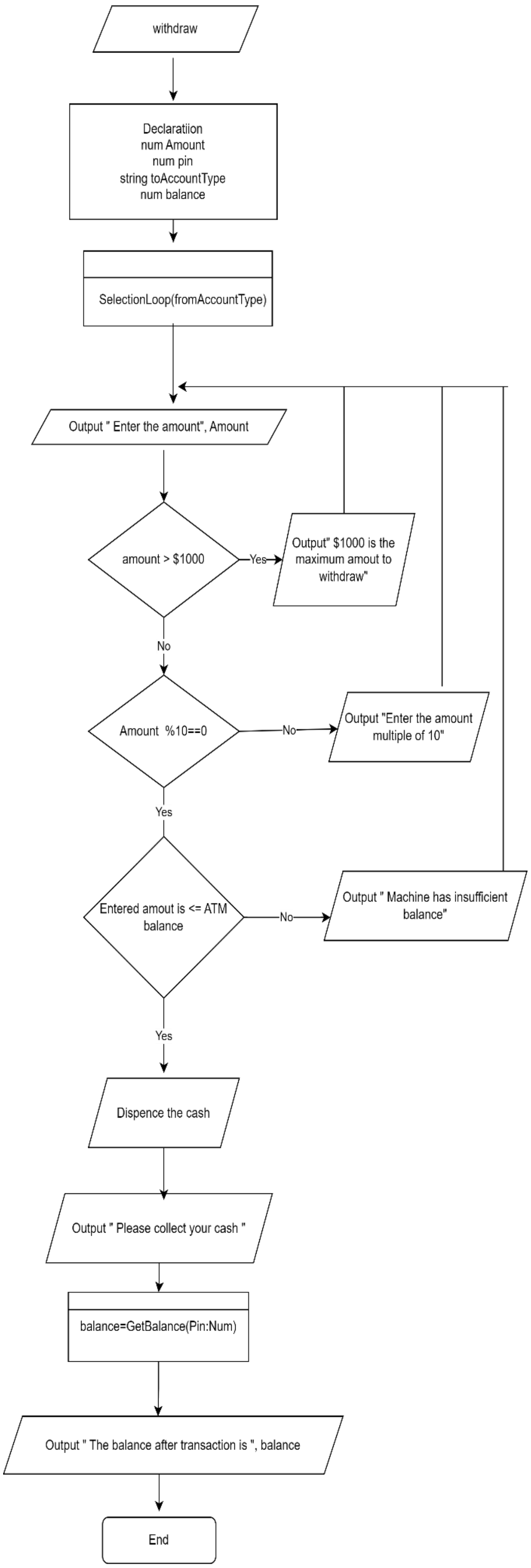
+getBenificiary ():string
+setBenificiary(string benificiary):void
+getBalance():num
+setBalance(num balance):void
+getFromAccountType():string
+setTFromAccountType(string FromAccountType):void

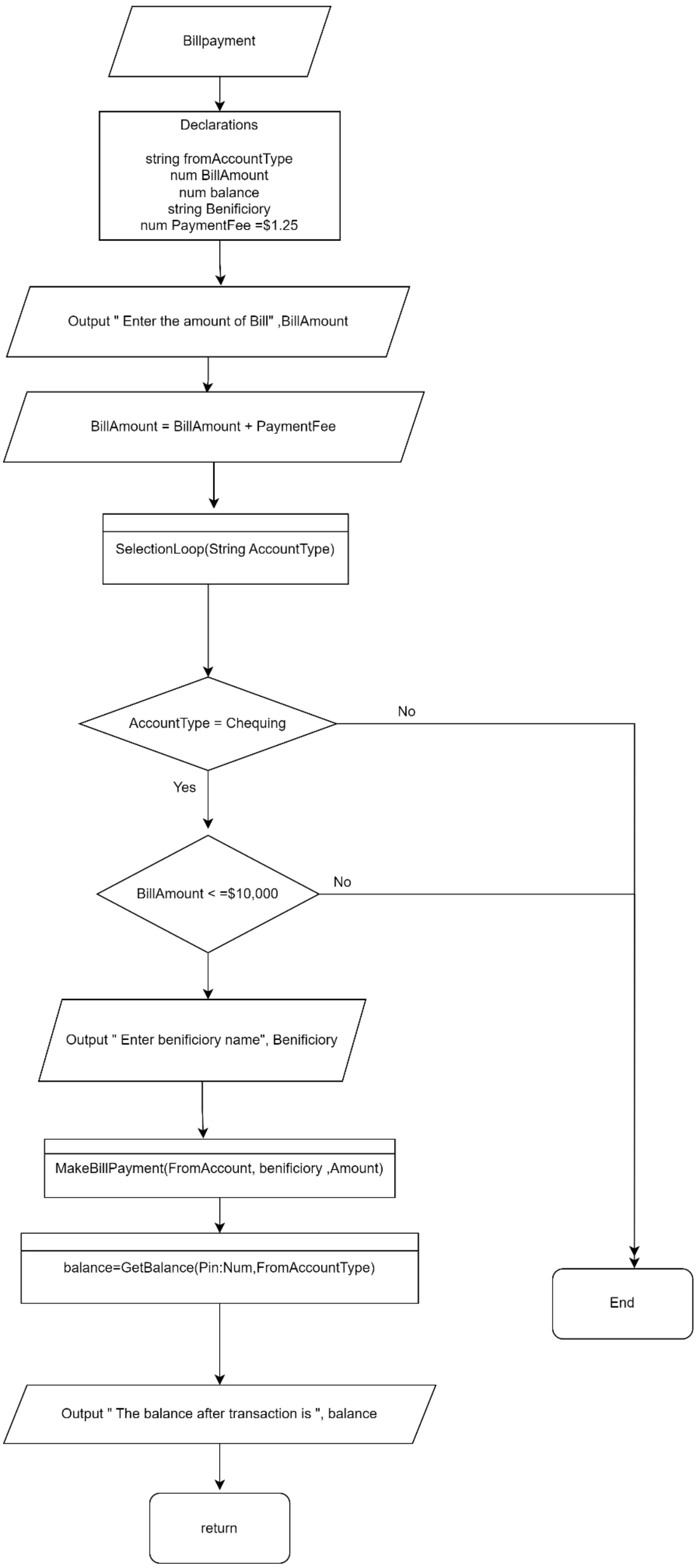
Flowchart:

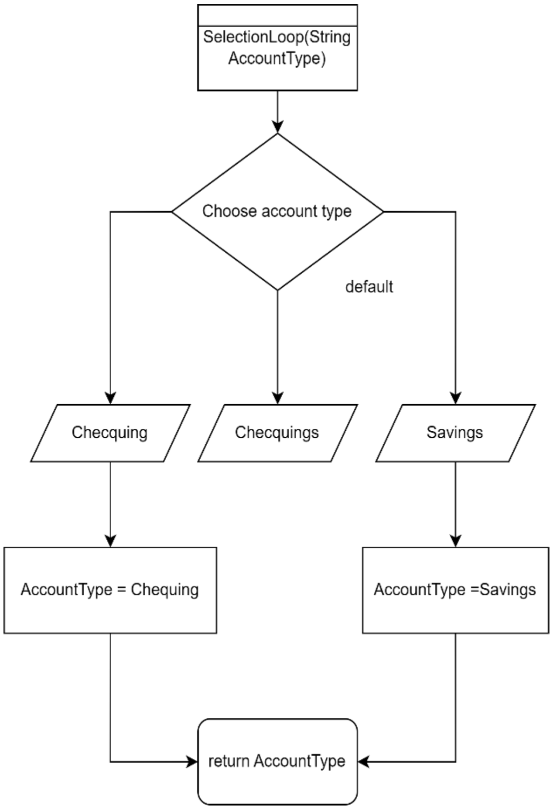

start
Declaration
String TransactionType
String name
num pin
Output "Enter your name" ,name
Output "Enter your PIN",pin
If name and PIN matches with that is in the data base
No
INumber of attemopt <=3
Yes
No
Output " You have exceeded your max attempt. Please Try again later"
End
Yes
show menu screen select option
TransactionType =selectedoption
Deposit
withdraw
transfer
Billpayment

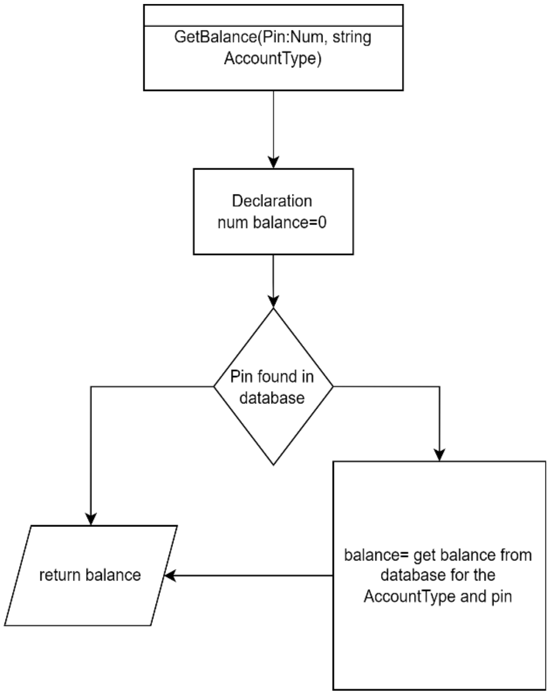

Deposit
Declaratiion
num Amount
num pin
num AccountType
string toAccountType
num balance
SelectionLoop(toAccountType)
Output " Enter the amount to deposit", amount
balance=GetBalance(Pin:Num,toAccountType)
balance= amount+balance
fetch account database using pin
update the new balance for AccountType
toAccountType
Output " Balance is" ,balance
End

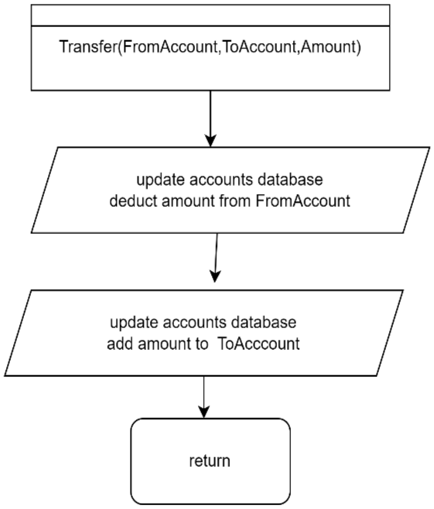

transfer
Declarations
string fromAccountType
s tring toAccountType
num transferAmount
num balance
Output " Enter the amount to transfer",
transferAmmount
SelectionLoop(fromAccountType)
SelectionLoop(toAccountType)
toAccountType <> fromAccountType
No
Yes
Amount is <=$10,000
No
End
Yes
balance=GetBalance(Pin:Num,FromAccountType)
transferAmount<=balance
No
yes
End
Transfer(FromAccount,TopAccount,Amount)
balance=GetBalance(Pin:Num)
Output " The balance after transaction is ", balance
return

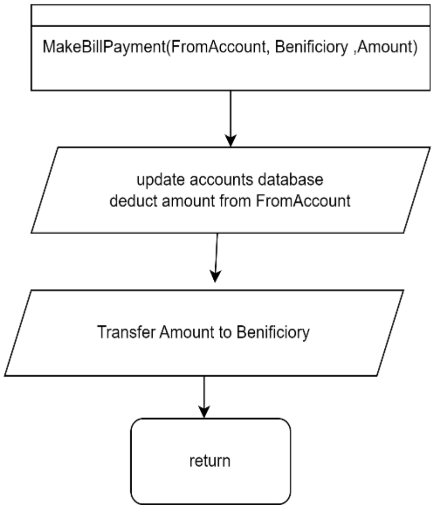

withdraw
Declaratiion
num Amount
num pin
string toAccountType
num balance
SelectionLoop(fromAccountType)
Output " Enter the amount", Amount
amount > $1000
Yes
Output" $1000 is the maximum amout to withdraw"
No
Amount %10==0
No
Output "Enter the amount multiple of 10"
Yes
Entered amout is <= ATM balance
No
Output " Machine has insufficient balance"
Yes
Dispence the cash
Output " Please collect your cash "
balance=GetBalance(Pin:Num)
Output " The balance after transaction is ", balance
End


Billpayment
Declarations
string fromAccountType
num BillAmount
num balance
string Benificiory
num PaymentFee =$1.25
Output " Enter the amount of Bill" ,BillAmount
BillAmount = BillAmount + PaymentFee
SelectionLoop(String AccountType)
AccountType = Chequing
No
Yes
BillAmount < =$10,000
No
Output " Enter benificiory name", Benificiory
MakeBillPayment(FromAccount, benificiory ,Amount)
balance=GetBalance(Pin:Num,FromAccountType)
Output " The balance after transaction is ", balance
return
End


SelectionLoop(String AccountType)
Choose account type
default
Chequing
Chequings
Savings
AccountType = Chequing
AccountType =Savings
return AccountType


GetBalance(Pin:Num, string AccountType)
Declaration
num balance=0
Pin found in database
return balance
balance= get balance from database for the AccountType and pin


Transfer(FromAccount,ToAccount,Amount)
update accounts database
deduct amount from FromAccount
update accounts database
add amount to ToAcccount
return


MakeBillPayment(FromAccount, Benificiory ,Amount)
update accounts database
deduct amount from FromAccount
Transfer Amount to Benificiory
return

Pseudo Code:

```
Class  Customer
Declaration
        num pin
        string name
        string accountType
        num loginAttempts

public num getPin()
        return this.pin

public void setPin(num pin)
        this.pin =pin
        return

public void getName()
        return this.name

public void setName(string name)
        this.name=name
        return

public string getAccountType()
        return this.accountType

public void setAccountType(string accountType)
        this.accountType= accountType
        return

public num getLoginAttempts()
        return this.loginAttempts

public void setLoginAttempts(num loginAttempts)
        this.loginAttempts= loginAttempts
        return
```

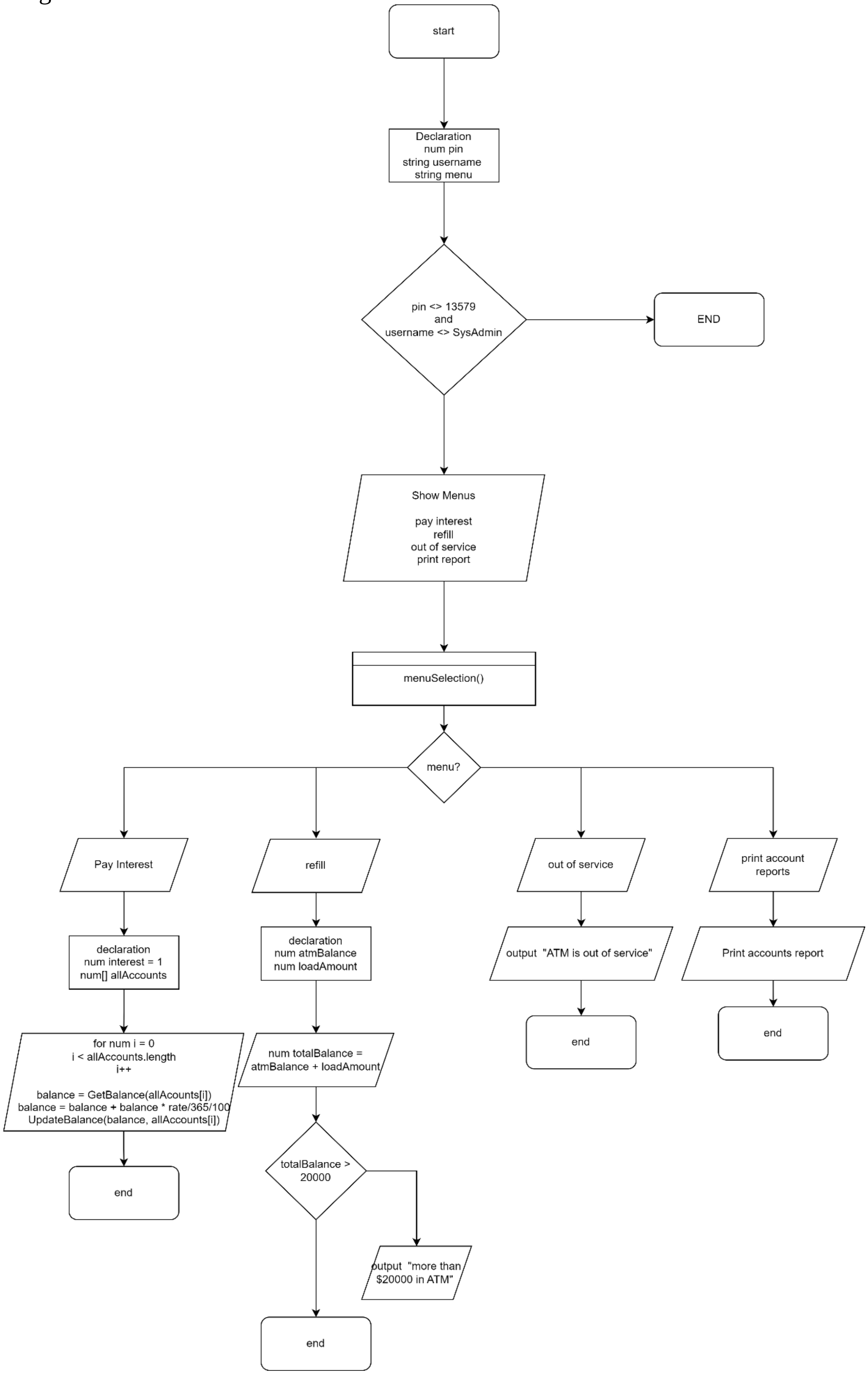
```
Class Tranasction
    Declaration
        String name
        num amount
        Customer customer
        Num balance
Public num getName()
    return this.name

public void setName(string name)
    this.name = name
    return

Public num getAmount()
    return this.amount

public void setAmount(num amount)
    this.amount = amount
    return

Public num getCustomer ()
    return this.customer

public void setCustomer(Customer customer)
    this. customer = customer
    return
Public num getBalance()
    Return this.balance

Public void setBalance(num balance)
    this.balance = balance
    return
public void perform()



Class Deposit inherits Transaction
    Deposit()
        selectAccount(toAccountType)
        Label label()
        Label.setText(“Enter the amount to deposit”)
        amountPromopt.setMessageLabel(label)
        amountPromopt.getOkButton().registerListener(perform)
        This.display()
    Private string toAccountType
    Private AmountPromopt amountPromopt
    Private BalanceScreen balanceScreen
    Public string getToAccountType()
        Return toAccountType
    Public void setToAccountType(string toAccountType)
        this.toAccountType = toAccountType
        return
    private void perform()
        balance = GetBalance(pin, accountType)
        balance = balance + amountPromopt.getAmount().getText()
        UpdateNewBalance(balance)
        balanceScreen.display()
```

```
Class Withdrawal inherits Transaction
    Withdrawal()
        selectAccount(fromAccountType)
        Label label()
        Label.setText(“Enter the amount to withdraw”)
        amountPromopt.setMessageLabel(label)
        amountPromopt.getOkButton().registerListener(perform)
        This.display()

        This.display()
    Private string fromAccountType
    Private AmountPromopt amountPromopt
    Private BalanceScreen balanceScreen
    Public string getFromAccountType()
        Return fromAccountType

    Public void setFromAccountType(string fromAccountType)
        this.fromAccountType = fromAccountType
        return
    private void perform()
        num amount = amountPromopt.getAmount().getText()
        num atmBalance = getAtmBalance()
        balance = GetBalance(pin, accountType)
        if amount > balance
            MessageScreen message()
            Label label()
            Label.settext(“Insuffcient fund”)
            message.setLabel(label)
            message.display()
        else if amount > 1000
            MessageScreen message()
            Label label()
            Label.settext(“Max $1000 is allowed to withdraw”)
            message.setLabel(label)
            message.display()
        else if amount %10 == 0
            MessageScreen message()
            Label label()
            Label.settext(“Enter the amount in multiples of $10”)
            message.setLabel(label)
            message.display()
        else if amount > atmBalance
            MessageScreen message()
            Label label()
            Label.settext(“ATM has insufficient cash”)
            message.setLabel(label)
            message.display()
        else
            dispenseCash()
            balance = GetBalance(pin, accountType)
            balance = balance – amount
            updateNewBalance(balance)
            balanceScreen.display()




Class Transfer inherits Transaction
    Transfer()
        selectAccount(fromAccountType)
        selectAccount(toAccountType)

        Label label()
        Label.setText(“Enter the amount to transfer”)
        amountPromopt.setMessageLabel(label)
        amountPromopt.getOkButton().registerListener(perform)
        This.display()

    Private string toAccountType
    Private string fromAccountType
    Private AmountPromopt amountPromopt
    Public string getToAccountType()
        Return toAccountType
    Public void setToAccountType(string toAccountType)
        this.toAccountType = toAccountType
        return

    Public string getFromAccountType()
        return fromAccountType

    Public void setFromAccountType(string fromAccountType)
        this.fromAccountType = fromAccountType
        return
    private perform()
        balance = GetBalance(pin, accountType)
        if amount > balance
            MessageScreen message()
            Label label()
            Label.settext(“Insuffcient fund”)
            message.setLabel(label)
            message.display()

        else if fromAccountType == toAccountType
            MessageScreen message()
            Label label()
            Label.settext(“From account cannot be same as to account”)
            message.setLabel(label)
            message.display()
        else if amount > 10000
            MessageScreen message()
            Label label()
            Label.settext(“Max $10000 is allowed for transfer”)
            message.setLabel(label)
            message.display()
        else
            transfer()
            balance = GetBalance(pin, accountType)
            balance = balance – amount
            updateNewBalance(balance)
            balanceScreen.display()




Class Billpayment inherits Transaction
    Billpayment()
        selectAccount(fromAccountType)
        selectBeneficiary(beneficiary)

        Label label()
        Label.setText(“Enter the amount to pay”)
        amountPromopt.setMessageLabel(label)
        amountPromopt.getOkButton().registerListener(perform)
        This.display()

    Private string beneficiary
    Private string fromAccountType
    Private AmountPromopt amountPromopt
    Public string getBeneficiary()
        Return beneficiary

    Public void setBeneficiary (string beneficiary)
        this. beneficiary= beneficiary
        return

    Public string getFromAccountType()
        return fromAccountType

    Public void setFromAccountType(string fromAccountType)
        this.fromAccountType = fromAccountType
        return
    private perform()
        balance = GetBalance(pin, accountType)
        if amount > balance
            MessageScreen message()
            Label label()
            Label.settext(“Insuffcient fund”)
            message.setLabel(label)
            message.display()
        else if fromAccountType <> chequing
            MessageScreen message()
            Label label()
            Label.settext(“Bill payment from chequing account only”)
            message.setLabel(label)
            message.display()
        if amount > 10000
            MessageScreen message()
            Label label()
            Label.settext(“Max $10000 is allowed for bill payment”)
            message.setLabel(label)
            message.display()
        else

            billPayment(amount)
            UpdateNewBalance(balance – (amount+1.25))
            balanceScreen.display()

Class WelcomeScreen
    Private static Label welcomeMessage
    Private static Label message

    Public static Label getWelcomeMessage()
        Return welcomeMessage

    Public static Label getMessage()
        Return message

Class LoginScreen
    LoginScreen()
        okButton.registerListener(showMainMenu())
    Private static Label message
    Private Button okButton
    Private Button cancelButton
    Private Text pin
    Private Customer customer

    Public static Label getMessage()
        Return message
    Private showMainMenu()
        Customer customer()
        customer.setPin(pin.getText())
        customer.setName(name.getText())
        If customer.getPin() and customer .getName() exist in CustomerInfo database
            MainMenu mainMenu()
            Button deposit()
            Button withdrwal()
            Button transfer()
            Button billPayment()
            Button[] transactionTypes
            transactionTypes[0] = deposit
            transactionTypes[1] = withdrwal
            transactionTypes[2] = transfer
            transactionTypes[3] = billPayment
            mainMenu.display()
        else
            customer. setLoginAttempts(customer. getLoginAttempts()+1)
            if customer. getLoginAttempts() > 3
                MessageScreen message()
                Label label()
                label.settext("You have exceeded 3 attempts. Please try again later")
                message.setLabel(label)
                message.display()
            end if
        end if



Class MainMenu
    Private Button[] transactionTypes
    Private Button okButton
```

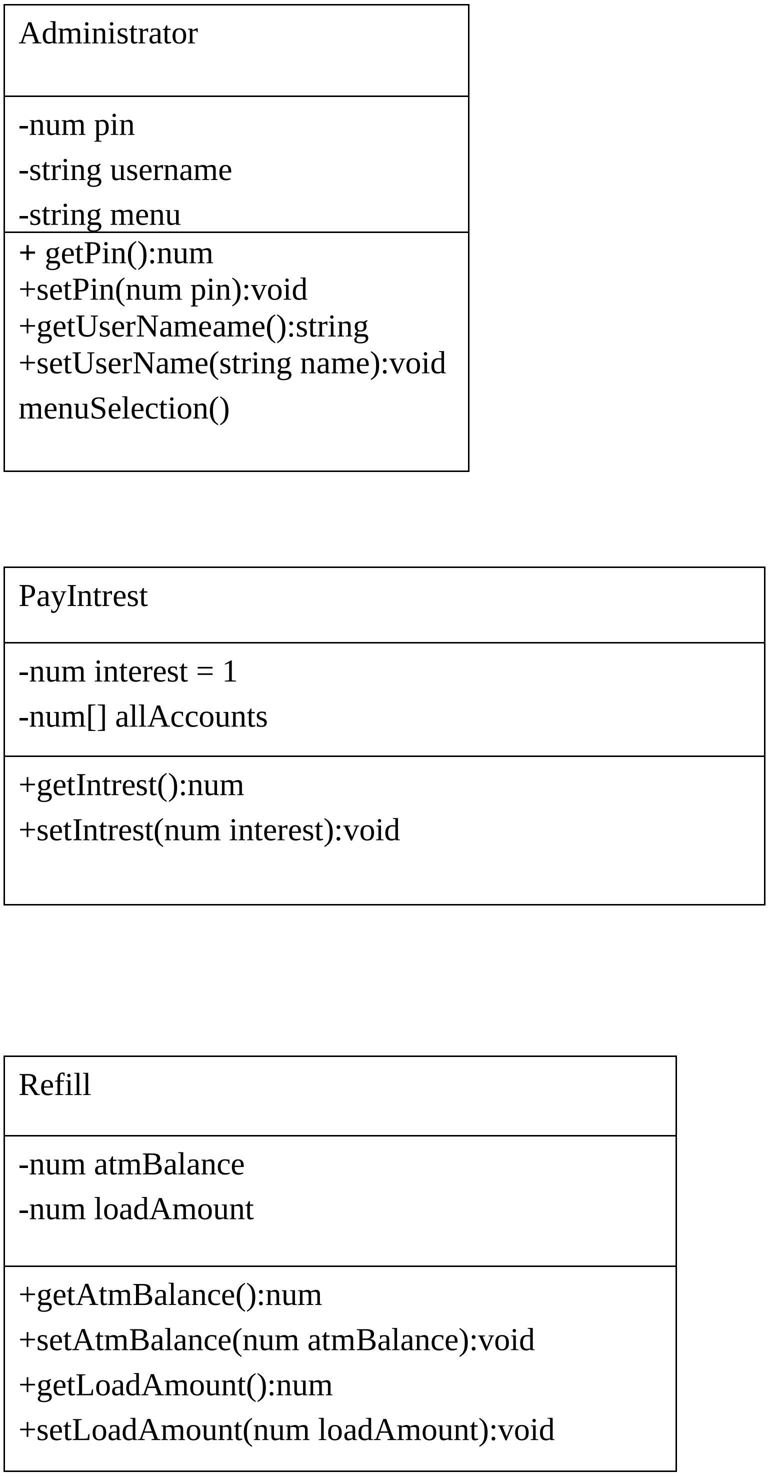
```
          Private Button cancelButton


          Public void setTransactionTypes(Transaction[] transactionTypes)
                    this. transactionTypes = transactionTypes
                    setButtons()
          private setButtons()
                    for(num I, I < transactionTypes.length; i++)
                              buttons[i].setText(transactionTypes[i].name)
end class

Class BalanceScreen
          Private Label balanceLabel
          Private Button okButton

          Public Label getBalanceLabel()
                    Return this.balanceLabel
          Public void setBalanceLabel(Label balanceLabel)
                    This.balanceLabel=balanceLabel
                    Return
          Public Button getOkButton()
                    Return this.okButton
          Public void setOkButton (Button okButton)
                    This.okButton = okButton
                    return
end class

Class MessageScreen
          Private Label messageLabel
          Private Button okButton

          Public Label getMessageLabel ()
                    Return this.messageLabel
          Public void setMessageLabel (Label messageLabel)
                    This.messageLabel = messageLabel
                    Return
          Public Button getOkButton()
                    Return this.okButton
          Public void setOkButton (Button okButton)
                    This.okButton = okButton
                    return
end class

Class AmountPrompt
          Private Label messageLabel
          Private TextInput amount
          Private Button okButton
          Private Button cancelButton
          Private Keypad keypad

          public Label getMessageLabel()
                     return messageLabel

           public void setMessageLabel(Label messageLabel)
                     this.messageLabel = messageLabel

           public Button getOkButton()
```

```
         return okButton

    public void setOkButton(Button okButton)
        this.okButton = okButton

    public Button getCancelButton()
        return cancelButton

    public void setCancelButton(Button cancelButton)
        this.cancelButton = cancelButton

    public Keypad getKeypad()
        return keypad

    public void setKeypad(Keypad keypad)
        this.keypad = keypad
end class

Class SelectAccountType
    Private Radio savingsAccount
    Private radio chequingAccount

    public Radio getSavingsAccount()
        return savingsAccount

    public void setSavingsAccount(Radio savingsAccount)
        this.savingsAccount = savingsAccount

    public Radio getChequingAccount()
        return chequingAccount

    public void setChequingAccount(Radio chequingAccount)
        this.chequingAccount = chequingAccount
End class

Start
    Declarations
        Num loggedIn = 0
        WelcomeScreen welcomeScreen()

    welcomeScreen.display()

    if card inserted
        welcomeScreen.remove()
        LoginScreen loginScreen()
        loginScreen.display()
stop
```

Program for Administrator user: Flowchart



Class Diagram:



Administrator

-num pin
-string username
-string menu

+ getPin():num
+setPin(num pin):void
+getUserNameame():string
+setUserName(string name):void
menuSelection()

PayIntrest

-num interest = 1
-num[] allAccounts

+getIntrest():num
+setIntrest(num interest):void

Refill

-num atmBalance
-num loadAmount

+getAtmBalance():num
+setAtmBalance(num atmBalance):void
+getLoadAmount():num
+setLoadAmount(num loadAmount):void

Psuedo code:

```
Class Administrator
    Privatenum pin
    Private string username
    Private string menu

public num getPin()
    return this.pin

public void setPin(num pin)
    this.pin =pin
    return

public void getUserName()
    return this.username

public void setUseName(string username)
    this.username=username
    return



Class PayIntrest
    Declaration
        Private num interest=1
        Private num[] allAccounts
        Private Button paybutton
    Public void pay()
        For( i=0, i<allAccounts.length, i++)
            balance=GetBalance(allAccounts[i]
            balance=balance+balance*rate/365/100
            atm.updateBalance(balance,allAccounts[i]
Return


Class Refill
    Declaration
        Private num atmBalance
        Private num loadAmount

Public  num getAtmBalance()
    return this.atmbalance
public void setAtmBalance(numatmBalance)
    this.atmBalance =atmBalance
    return

Public  num getLoadAmount()
    return this.loadAmount
```

```
public void setLoadAmount(num loadAmount)
    this.loadAmount =loadAmount
    return
public void refill()
    num totalBalance=atm.getbalance+loadAmount
    if  totalBalance >20,000
            MessageScreen message()
            Label label()
            Label.settext(“More than $20,000 in the atm”)
            message.setLabel(label)
            message.display()
    else if
            MessageScreen message()
            Label label()
            Label.settext(“Money has been loaded
            message.setLabel(label)
            message.display()

    endif



Menu
    Declaration
        Button payInterstButton
        Button refillButton
        Button outOfServiceButton
        Button printAccountReportButton
    Menu()
        payInterstButton.setText(“Pay interest”);
        refillButton.setText(“Refill”);
        outOfServiceButton.setText(“Out of service”);
        printAccountReportButton.setText(“Print report”);

        payInterstButton.registerListener(payInterest)
        refillButton.registerListener(refill)
        outOfServiceButton.registerListener(outofService)
        printAccountReportButton.registerListener(print)

    private payInterest()
        PayInterest payInterest()
        payInterest.pay()
    private refill()
        atm.refill()
    private outofService()
        MessageScreen message()
        Label label()
        Label.settext(“ATM out of service”)
        message.setLabel(label)
        message.display()
    private print()
        atm.printReport()
start
    LoginScreen loginscreen()
    Loginscreen.display()
    Administrator administrator()
        If administrator.getpin<>13579 and administrator.getuserName<> SysAdmin
            MessageScreen message()
            Label label()
            Label.settext("Insuffcient fund")
            message.setLabel(label)
            message.display()

        else
            Menu menu()
            Menu.display()
        End if
stop
```