Visual Programming Lab

CSL-313

Lab Journal
(2+3)
Complete Package

Student Name: M. Anas Baig
Enrolment No.: 01-134152-037
Class and Section: BS(CS)-5A

Department of Computer Science
BAHRIA UNIVERSITY, ISLAMABAD

by M. Anas Baig (01-134152-037)

My Virtual Bank

In the lab task you will simulate a virtual Bank just like traditional Bank using OOP concepts and C# Console Application. Implementation details are as follows;

a) Account class with attributes {AccountNO, AccountTitle, CNIC, ContactNumber, Balance}
b) Two types of account 1. Current Account with attribute {WithdrawalLimit}, 2. Saving Account with attribute {ProfitPercentage}; both types should be inherited from the Account Class.
c) Program should allow user to open a new account with all information of account holder and type of account he wants to open. (CRUD Operations)
d) You are required to enable an account holder to perform transactions withdrawal and deposit.
e) Validate your inputs; especially for withdrawal and deposit.

Procedure/Program:

program.cs Class:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.IO;
6.  using System.Collections;
7.
8.  namespace ConsoleApplication1
9.  {
10.     class program
11.     {
12.         static void Main(string[] args)
13.         {
14.             Console.BackgroundColor = ConsoleColor.Gray;
15.             Console.Clear();
16.             Console.ForegroundColor = ConsoleColor.Black;
17.             Console.WriteLine("================================================================================");
18.             Console.WriteLine("   B A H R I A - U N I V E R S I T Y - V I R T U A L - B A N K - S Y S T E M");
19.             Console.WriteLine("================================================================================");
20.             consoleMenu c = new consoleMenu();
21.             c.startMenu();
22.             Console.ReadLine();
23.         }
24.     }
25. }
```

account.cs Class:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.
6.  namespace ConsoleApplication1
7.  {
8.      class account
9.      {
10.         protected string accountNo;
11.         protected string accountTitle;
12.         protected string cnic;
13.         protected string contactNo;
14.         protected double balance;
```

```
15.
16.         public account()
17.         {
18.         }
19.
20.         public account(string accountNo, string accountTitle, string cnic, string contactNo, doubl
    e balance)
21.         {
22.             this.accountNo = accountNo;
23.             this.accountTitle = accountTitle;
24.             this.cnic = cnic;
25.             this.contactNo = contactNo;
26.             this.balance = balance;
27.         }
28.
29.         public virtual double withdraw(double amount) //virtual function
30.         {
31.             return (this.balance -= amount);
32.         }
33.
34.         public virtual double deposit(double amount) //virtual function
35.         {
36.             return (this.balance += amount);
37.         }
38.     }
39. }
```

currentAccount.cs Class:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.
6.  namespace ConsoleApplication1
7.  {
8.      class currentAccount : account //inheritence
9.      {
10.         private double withdrawalLimit; //extra variable to store withdrawal limit
11.
12.         public currentAccount()
13.         {
14.         }
15.
16.         public currentAccount( string accountNo, string accountTitle, string cnic, string contactN
    o, double balance, double withdrawalLimit )
17.             : base(accountNo, accountTitle, cnic, contactNo, balance) //base class parametrized co
    nstructor call combine with child class
18.         {
19.             this.withdrawalLimit = withdrawalLimit;
20.         }
21.
22.         public string accountNoProperty //C# property: just like get set method
23.         {
24.             get { return accountNo; }
25.             set { accountNo = value; }
26.         }
27.
28.         public string accountTitleProperty //C# property: just like get set method
29.         {
30.             get { return accountTitle; }
31.             set { accountTitle = value; }
32.         }
33.
34.         public string cnicProperty //C# property: just like get set method
35.         {
36.             get { return cnic; }
```

```
            set { cnic = value; }
        }

        public string contactNoProperty //C# property: just like get set method
        {
            get { return contactNo; }
            set { contactNo = value; }
        }

        public double balanceProperty //C# property: just like get set method
        {
            get { return balance; }
            set { balance = value; }
        }

        public double withdrawalLimitProperty //C# property: just like get set method
        {
            get { return withdrawalLimit; }
            set { withdrawalLimit = value; }
        }

        public override double deposit(double amount) //virtual funtion overridden in child class

        {
            return base.deposit(amount);
        }

        public override double withdraw(double amount) //virtual funtion overridden in child class

        {
            if (amount <= this.withdrawalLimit)
            {
                return base.withdraw(amount);
            }
            return -1;
        }
    }
}
```

savingAccount.cs Class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class savingAccount : account //inheritence
    {
        private float profitPercentage; //extra variable to store profit percentage

        public savingAccount()
        {
        }

        public savingAccount( string accountNo, string accountTitle, string cnic, string contactNo
, double balance, float profitPercentage )
            : base(accountNo, accountTitle, cnic, contactNo, balance) //base class parametrized co
nstructor call combine with child class
        {
            this.profitPercentage = profitPercentage;
        }

        public string accountNoProperty //C# property: just like get set method
        {
            get { return accountNo; }
```

```
            set { accountNo = value; }
        }

        public string accountTitleProperty //C# property: just like get set method
        {
            get { return accountTitle; }
            set { accountTitle = value; }
        }

        public string cnicProperty //C# property: just like get set method
        {
            get { return cnic; }
            set { cnic = value; }
        }

        public string contactNoProperty //C# property: just like get set method
        {
            get { return contactNo; }
            set { contactNo = value; }
        }

        public double balanceProperty //C# property: just like get set method
        {
            get { return balance; }
            set { balance = value; }
        }

        public float profitPercentageProperty //C# property: just like get set method
        {
            get { return profitPercentage; }
            set { profitPercentage = value; }
        }

        public override double deposit(double amount) //virtual funtion overridden in child class

        {
            if (amount >= 10000)
            {
                profitPercentage++;
            }
            return base.deposit(amount);
        }

        public override double withdraw(double amount) //virtual funtion overridden in child class

        {
            if (amount >= 10000)
            {
                profitPercentage--;
            }
            return base.withdraw(amount);
        }
    }
}
```

consoleMenu.cs Class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Collections;

namespace ConsoleApplication1
{
    class consoleMenu
```

```
    {
        accountManager manager = new accountManager();

        public void startMenu()
        {
            printOptions();
            int num = int.Parse(Console.ReadLine());
            if( num == 1 )
            {
                accountCreation();
            }
            if( num == 2 )
            {
                searchAccount();
            }
            if (num == 3)
            {
                updateAccount();
            }
            if (num == 4)
            {
                deleteAccount();
            }
            if (num == 5)
            {
                displayAccount();
            }
            if (num == 6)
            {
                transaction();
            }
        }

        public void printOptions()
        {
            Console.WriteLine("Enter your desired operation:");
            Console.WriteLine("1. Create New Account.");
            Console.WriteLine("2. Search Account by Account Number.");
            Console.WriteLine("3. Update Account by Account Number.");
            Console.WriteLine("4. Delete Account by Account Number.");
            Console.WriteLine("5. Display Account by Account Number.");
            Console.WriteLine("6. Deposit / Withdraw by Account Number.");
        }

        public void accountCreation()
        {
            int num;
            do
            {
                Console.Write("\nEnter Account Number:\n");
                string An = Console.ReadLine();

                Console.Write("\nEnter Account Title:\n");
                string At = Console.ReadLine();

                Console.Write("\nEnter CNIC:\n");
                string cnic = Console.ReadLine();

                Console.Write("\nEnter Contact Number:\n");
                string cn = Console.ReadLine();

                Console.Write("\nEnter Account Balance:\n");
                double bal = double.Parse(Console.ReadLine());

                Console.Write("\nChoose Account Type:\n");
                Console.Write("1.Saving Account.    2.Current Account.\n");
                int option = int.Parse(Console.ReadLine());

                if (option == 1)
```

```
80.                 {
81.                     Console.WriteLine("\nEnter Profit Percentage:");
82.                     float pp = float.Parse(Console.ReadLine());
83.
84.                     savingAccount savingAcc = new savingAccount(An, At, cnic, cn, bal, pp);
85.                     manager.createNewSavingAccount(savingAcc);
86.                 }
87.                 if (option == 2)
88.                 {
89.                     Console.WriteLine("\nEnter Withdrawal Limit:");
90.                     double wl = double.Parse(Console.ReadLine());
91.
92.                     currentAccount currentAcc = new currentAccount(An, At, cnic, cn, bal, wl);
93.                     manager.createNewCurrentAccount(currentAcc);
94.                 }
95.                 Console.WriteLine("\nPress 1 to continue creation of Accounts.");
96.                 num = int.Parse(Console.ReadLine());
97.             }
98.             while (num == 1);
99.             manager.writeAllinFile();
100.            }
101.
102.            public void searchAccount()
103.            {
104.                Console.Write("\nChoose Account Type:\n");
105.                Console.Write("1.Current Account.    2.Saving Account.\n");
106.                int num = int.Parse(Console.ReadLine());
107.
108.                if ( num == 1)
109.                {
110.                    Console.WriteLine("\nEnter Account Number:");
111.                    string accnum = Console.ReadLine();
112.                    manager.searchCurrentAccount(accnum);
113.                }
114.                else if( num  == 2)
115.                {
116.                    Console.WriteLine("\nEnter Account Number:");
117.                    string accnum = Console.ReadLine();
118.                    manager.searchSavingAccount(accnum);
119.                }
120.                else
121.                {
122.                    Console.WriteLine("Invalid Input\n");
123.                }
124.            }
125.
126.            public void updateAccount()
127.            {
128.                Console.Write("\nChoose Account Type:\n");
129.                Console.Write("1.Current Account.   2.Saving Account.\n");
130.                int num = int.Parse(Console.ReadLine());
131.
132.                if (num == 1)
133.                {
134.                    Console.WriteLine("\nEnter Account Number:");
135.                    string accnum = Console.ReadLine();
136.
137.                    accountFile obj = new accountFile();
138.                    List<currentAccount> accounts = obj.readAllCurrentAccount(); //simple list:
    saves same type of objects
139.                    int totalAccounts = accounts.Count; //built-in count list function
140.                    bool found = false;
141.
142.                    for (int i = 0; i < totalAccounts; i++)
143.                    {
144.                        if (accounts[i].accountNoProperty == accnum)
145.                        {
146.                            found = true;
147.                            Console.WriteLine("\nAccount Found.\n");
```

```
148.
149.                                 Console.Write("\nEnter New Account Number:\n");
150.                                 accounts[i].accountNoProperty = Console.ReadLine();
151.
152.                                 Console.Write("\nEnter New Account Title:\n");
153.                                 accounts[i].accountTitleProperty = Console.ReadLine();
154.
155.                                 Console.Write("\nEnter New CNIC:\n");
156.                                 accounts[i].cnicProperty = Console.ReadLine();
157.
158.                                 Console.Write("\nEnter New Contact Number:\n");
159.                                 accounts[i].contactNoProperty = Console.ReadLine();
160.
161.                                 Console.Write("\nEnter New Account Balance:\n");
162.                                 accounts[i].balanceProperty = double.Parse(Console.ReadLine());
163.
164.                                 Console.WriteLine("\nEnter New Withdrawal Limit:");
165.                                 accounts[i].withdrawalLimitProperty = double.Parse(Console.ReadLine
    ());
166.                             }
167.                         }
168.
169.                         StreamWriter writeCurrent = new StreamWriter("CurrentAccount.txt"); //write
     to file object
170.                         //int totalAccount = accounts.Count;
171.                         for (int j = 0; j < totalAccounts; j++)
172.                         {
173.                             currentAccount account = accounts[j];
174.                             writeCurrent.WriteLine(account.accountNoProperty);
175.                             writeCurrent.WriteLine(account.accountTitleProperty);
176.                             writeCurrent.WriteLine(account.balanceProperty);
177.                             writeCurrent.WriteLine(account.cnicProperty);
178.                             writeCurrent.WriteLine(account.contactNoProperty);
179.                             writeCurrent.WriteLine(account.withdrawalLimitProperty);
180.                         }
181.                         writeCurrent.Close();
182.                         return;
183.                     }
184.
185.                     else if (num == 2)
186.                     {
187.                         Console.WriteLine("\nEnter Account Number:");
188.                         string accnum = Console.ReadLine();
189.
190.                         accountFile obj = new accountFile();
191.                         ArrayList accounts = obj.readAllSavingAccount(); //array list: saves diffee
    nt types of objects
192.                         //List<currentAccount> accounts = obj.readAllCurrentAccount(); //simple lis
    t: saves same type of objects
193.                         int totalAccounts = accounts.Count; //built-in count list function
194.                         bool found = false;
195.
196.                         for (int i = 0; i < totalAccounts; i++)
197.                         {
198.                             if ((accounts[i] as savingAccount).accountNoProperty == accnum)
199.                             {
200.                                 found = true;
201.                                 Console.WriteLine("\nAccount Found.\n");
202.
203.                                 Console.Write("\nEnter New Account Number:\n");
204.                                 (accounts[i] as savingAccount).accountNoProperty = Console.ReadLine
    ();
205.
206.                                 Console.Write("\nEnter New Account Title:\n");
207.                                 (accounts[i] as savingAccount).accountTitleProperty = Console.ReadL
    ine();
208.
209.                                 Console.Write("\nEnter New CNIC:\n");
210.                                 (accounts[i] as savingAccount).cnicProperty = Console.ReadLine();
```

```
211.
212.                                 Console.Write("\nEnter New Contact Number:\n");
213.                                 (accounts[i] as savingAccount).contactNoProperty = Console.ReadLine
    ();
214.
215.                                 Console.Write("\nEnter New Account Balance:\n");
216.                                 (accounts[i] as savingAccount).balanceProperty = double.Parse(Conso
    le.ReadLine());
217.
218.                                 Console.WriteLine("\nEnter New Profit Percentage:");
219.                                 (accounts[i] as savingAccount).profitPercentageProperty = float.Par
    se(Console.ReadLine());
220.                             }
221.                         }
222.
223.                         StreamWriter writeSaving = new StreamWriter("SavingAccount.txt"); //write t
    o file object
224.                         //int totalAccount = accounts.Count;
225.                         for (int j = 0; j < totalAccounts; j++)
226.                         {
227.                             savingAccount account = accounts[j] as savingAccount;
228.                             writeSaving.WriteLine(account.accountNoProperty);
229.                             writeSaving.WriteLine(account.accountTitleProperty);
230.                             writeSaving.WriteLine(account.balanceProperty);
231.                             writeSaving.WriteLine(account.cnicProperty);
232.                             writeSaving.WriteLine(account.contactNoProperty);
233.                             writeSaving.WriteLine(account.profitPercentageProperty);
234.                         }
235.                         writeSaving.Close();
236.                         return;
237.                     }
238.                     else
239.                     {
240.                         Console.WriteLine("Invalid Input\n");
241.                     }
242.                 }
243.
244.                 public void deleteAccount()
245.                 {
246.                     Console.Write("\nChoose Account Type:\n");
247.                     Console.Write("1.Current Account.   2.Saving Account.\n");
248.                     int num = int.Parse(Console.ReadLine());
249.
250.                     if (num == 1)
251.                     {
252.                         Console.WriteLine("\nEnter Account Number:");
253.                         string accnum = Console.ReadLine();
254.                         accountFile obj = new accountFile();
255.                         List<currentAccount> accounts = obj.readAllCurrentAccount(); //simple list:
    saves same type of objects
256.                         int totalAccounts = accounts.Count; //built-in count list function
257.                         bool found = false;
258.
259.                         for (int i = 0; i < totalAccounts; i++)
260.                         {
261.                             if (accounts[i].accountNoProperty == accnum)
262.                             {
263.                                 found = true;
264.                                 Console.WriteLine("\nAccount Found.\n");
265.
266.                                 accounts.Remove(accounts[i]); //deletes that object with i index
267.                                 totalAccounts--;
268.                             }
269.                         }
270.
271.                         StreamWriter writeCurrent = new StreamWriter("CurrentAccount.txt"); //write
    to file object
272.
273.                         for (int j = 0; j < totalAccounts; j++)
```

```
                    {
                        currentAccount account = accounts[j];
                        writeCurrent.WriteLine(account.accountNoProperty);
                        writeCurrent.WriteLine(account.accountTitleProperty);
                        writeCurrent.WriteLine(account.balanceProperty);
                        writeCurrent.WriteLine(account.cnicProperty);
                        writeCurrent.WriteLine(account.contactNoProperty);
                        writeCurrent.WriteLine(account.withdrawalLimitProperty);
                    }
                    writeCurrent.Close();
                    return;
                }

                else if (num == 2)
                {
                    Console.WriteLine("\nEnter Account Number:");
                    string accnum = Console.ReadLine();
                    accountFile obj = new accountFile();
                    ArrayList accounts = obj.readAllSavingAccount(); //array list: saves diffee
    nt types of objects
                    int totalAccounts = accounts.Count; //built-in count list function
                    bool found = false;

                    for (int i = 0; i < totalAccounts; i++)
                    {
                        if ((accounts[i] as savingAccount).accountNoProperty == accnum)
                        {
                            found = true;
                            Console.WriteLine("\nAccount Found.\n");

                            accounts.Remove(accounts[i]); //deletes that object with i index
                            totalAccounts--;
                        }
                    }

                    StreamWriter writeSaving = new StreamWriter("SavingAccount.txt"); //write t
    o file object

                    for (int j = 0; j < totalAccounts; j++)
                    {
                        savingAccount account = accounts[j] as savingAccount;
                        writeSaving.WriteLine(account.accountNoProperty);
                        writeSaving.WriteLine(account.accountTitleProperty);
                        writeSaving.WriteLine(account.balanceProperty);
                        writeSaving.WriteLine(account.cnicProperty);
                        writeSaving.WriteLine(account.contactNoProperty);
                        writeSaving.WriteLine(account.profitPercentageProperty);
                    }
                    writeSaving.Close();
                    return;
                }
                else
                {
                    Console.WriteLine("Invalid Input\n");
                }
            }


            public void displayAccount()
            {
                Console.Write("\nChoose Account Type:\n");
                Console.Write("1.Current Account.   2.Saving Account.\n");
                int num = int.Parse(Console.ReadLine());

                if (num == 1)
                {
                    Console.WriteLine("\nEnter Account Number:");
                    string accnum = Console.ReadLine();

```

```
341.                        accountFile obj = new accountFile();
342.                        List<currentAccount> accounts = obj.readAllCurrentAccount(); //simple list:
    saves same type of objects
343.                        int totalAccounts = accounts.Count; //built-in count list function
344.                        bool found = false;
345.
346.                        for (int i = 0; i < totalAccounts; i++)
347.                        {
348.                            if (accounts[i].accountNoProperty == accnum)
349.                            {
350.                                found = true;
351.                                Console.WriteLine("\nAccount Found.\n");
352.
353.                                Console.Write("\nAccount Number:    ");
354.                                Console.Write(accounts[i].accountNoProperty);
355.
356.                                Console.Write("\nAccount Title: ");
357.                                Console.Write(accounts[i].accountTitleProperty);
358.
359.                                Console.Write("\nCNIC:  ");
360.                                Console.Write(accounts[i].cnicProperty);
361.
362.                                Console.Write("\nContact Number:    ");
363.                                Console.Write(accounts[i].contactNoProperty);
364.
365.                                Console.Write("\nAccount Balance:   ");
366.                                Console.Write(accounts[i].balanceProperty);
367.
368.                                Console.WriteLine("\nWithdrawal Limit:  ");
369.                                Console.Write(accounts[i].withdrawalLimitProperty);
370.                            }
371.                        }
372.                    }
373.
374.                    else if (num == 2)
375.                    {
376.                        Console.WriteLine("\nEnter Account Number:");
377.                        string accnum = Console.ReadLine();
378.
379.                        accountFile obj = new accountFile();
380.                        ArrayList accounts = obj.readAllSavingAccount(); //array list: saves diffee
    nt types of objects
381.
382.                        int totalAccounts = accounts.Count; //built-in count list function
383.                        bool found = false;
384.
385.                        for (int i = 0; i < totalAccounts; i++)
386.                        {
387.                            if ((accounts[i] as savingAccount).accountNoProperty == accnum)
388.                            {
389.                                found = true;
390.                                Console.WriteLine("\nAccount Found.\n");
391.
392.                                Console.Write("\nAccount Number:");
393.                                Console.Write((accounts[i] as savingAccount).accountNoProperty);
394.
395.                                Console.Write("\nAccount Title:");
396.                                Console.Write((accounts[i] as savingAccount).accountTitleProperty);
397.
398.                                Console.Write("\nCNIC:");
399.                                Console.Write((accounts[i] as savingAccount).cnicProperty);
400.
401.                                Console.Write("\nContact Number:");
402.                                Console.Write((accounts[i] as savingAccount).contactNoProperty);
403.
404.                                Console.Write("\nAccount Balance:");
405.                                Console.Write((accounts[i] as savingAccount).balanceProperty);
406.
```

```
                                Console.WriteLine("\nProfit Percentage:");
                                Console.Write((accounts[i] as savingAccount).profitPercentageProperty);
                            }
                        }
                    }
                    else
                    {
                        Console.WriteLine("Invalid Input\n");
                    }
                }

                public void transaction()
                {
                    Console.Write("\nChoose Account Type:\n");
                    Console.Write("1.Current Account.   2.Saving Account.\n");
                    int num = int.Parse(Console.ReadLine());

                    if (num == 1)
                    {
                        Console.WriteLine("\nEnter Account Number:");
                        string accnum = Console.ReadLine();

                        accountFile obj = new accountFile();
                        List<currentAccount> accounts = obj.readAllCurrentAccount(); //simple list: saves same type of objects
                        int totalAccounts = accounts.Count; //built-in count list function
                        bool found = false;

                        for (int i = 0; i < totalAccounts; i++)
                        {
                            if (accounts[i].accountNoProperty == accnum)
                            {
                                found = true;
                                Console.WriteLine("\nAccount Found.\n");

                                Console.Write("\nChoose Transaction Type:\n");
                                Console.Write("1.Deposit Amount.   2.Withdraw Amount.\n");
                                int option = int.Parse(Console.ReadLine());

                                if (option == 1)
                                {
                                    Console.Write("Enter Deposit Amount:\n");
                                    int depositAmount = int.Parse(Console.ReadLine());
                                    accounts[i].balanceProperty = (accounts[i].balanceProperty + depositAmount);
                                    accounts[i].deposit(depositAmount);
                                }
                                if (option == 2)
                                {
                                    Console.Write("Enter Withdrawal Amount:\n");
                                    int withdrawalAmount = int.Parse(Console.ReadLine());
                                    if (withdrawalAmount <= accounts[i].withdrawalLimitProperty)
                                    {
                                        accounts[i].balanceProperty = (accounts[i].balanceProperty + withdrawalAmount);
                                        accounts[i].withdraw(withdrawalAmount);
                                    }
                                    else
                                    {
                                        Console.Write("ERROR!!! Amount more than Withdrawal Limit.\n");
                                    }
                                }
                            }
                        }

                        StreamWriter writeCurrent = new StreamWriter("CurrentAccount.txt"); //write to file object
```

```
470.
471.                         for (int j = 0; j < totalAccounts; j++)
472.                         {
473.                             currentAccount account = accounts[j];
474.                             writeCurrent.WriteLine(account.accountNoProperty);
475.                             writeCurrent.WriteLine(account.accountTitleProperty);
476.                             writeCurrent.WriteLine(account.balanceProperty);
477.                             writeCurrent.WriteLine(account.cnicProperty);
478.                             writeCurrent.WriteLine(account.contactNoProperty);
479.                             writeCurrent.WriteLine(account.withdrawalLimitProperty);
480.                         }
481.                         writeCurrent.Close();
482.                         return;
483.                     }
484.
485.                     else if (num == 2)
486.                     {
487.                         Console.WriteLine("\nEnter Account Number:");
488.                         string accnum = Console.ReadLine();
489.
490.                         accountFile obj = new accountFile();
491.                         ArrayList accounts = obj.readAllSavingAccount(); //array list: saves diffee
    nt types of objects
492.
493.                         int totalAccounts = accounts.Count; //built-in count list function
494.                         bool found = false;
495.
496.                         for (int i = 0; i < totalAccounts; i++)
497.                         {
498.                             if ((accounts[i] as savingAccount).accountNoProperty == accnum)
499.                             {
500.                                 found = true;
501.                                 Console.WriteLine("\nAccount Found.\n");
502.
503.                                 Console.Write("\nChoose Transaction Type:\n");
504.                                 Console.Write("1.Deposit Amount.   2.Withdraw Amount.\n");
505.                                 int option = int.Parse(Console.ReadLine());
506.
507.                                 if (option == 1)
508.                                 {
509.                                     Console.Write("Enter Deposit Amount:\n");
510.                                     int depositAmount = int.Parse(Console.ReadLine());
511.                                     (accounts[i] as savingAccount).balanceProperty = ((accounts[i]
    as savingAccount).balanceProperty + depositAmount);
512.                                     (accounts[i] as savingAccount).deposit(depositAmount);
513.                                 }
514.                                 if (option == 2)
515.                                 {
516.                                     Console.Write("Enter Withdrawal Amount:\n");
517.                                     int withdrawalAmount = int.Parse(Console.ReadLine());
518.                                     (accounts[i] as savingAccount).balanceProperty = ((accounts[i]
    as savingAccount).balanceProperty + withdrawalAmount);
519.                                     (accounts[i] as savingAccount).withdraw(withdrawalAmount);
520.                                 }
521.                             }
522.                         }
523.
524.                         StreamWriter writeSaving = new StreamWriter("SavingAccount.txt"); //write t
    o file object
525.
526.                         for (int j = 0; j < totalAccounts; j++)
527.                         {
528.                             savingAccount account = accounts[j] as savingAccount;
529.                             writeSaving.WriteLine(account.accountNoProperty);
530.                             writeSaving.WriteLine(account.accountTitleProperty);
531.                             writeSaving.WriteLine(account.balanceProperty);
532.                             writeSaving.WriteLine(account.cnicProperty);
533.                             writeSaving.WriteLine(account.contactNoProperty);
534.                             writeSaving.WriteLine(account.profitPercentageProperty);
```

```
                    }
                    writeSaving.Close();
                    return;
                }
                else
                {
                    Console.WriteLine("Invalid Input\n");
                }
            }
        }
    }
```

accountManager.cs Class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.IO;

namespace ConsoleApplication1
{
    class accountManager
    {
        ArrayList savingAccountList = new ArrayList(); //array list: saves different types of objects
        List<currentAccount> currentAccountList = new List<currentAccount>(); //simple list: saves same type of objects

        public void createNewSavingAccount(savingAccount sa)
        {
            this.savingAccountList.Add(sa); //adding object to array list
        }

        public void createNewCurrentAccount(currentAccount ca)
        {
            this.currentAccountList.Add(ca); //adding object to simple list
        }

        public void writeAllinFile()
        {
            accountFile write = new accountFile();
            write.writeAllSavingAccount(savingAccountList);
            write.writeAllCurrentAccount(currentAccountList);
        }

        public void searchCurrentAccount(string An)
        {
            accountFile obj = new accountFile();
            List<currentAccount> accounts = obj.readAllCurrentAccount(); //simple list: saves same type of objects
            int totalAccounts = accounts.Count; //built-in count list function
            bool found = false;

            for (int i = 0; i < totalAccounts; i++)
            {
                if (accounts[i].accountNoProperty == An)
                {
                    found = true;
                    Console.WriteLine("\nAccount Found.\n");
                    return;
                }
            }
            Console.WriteLine("\nAccount Not Found.\n");
        }

```

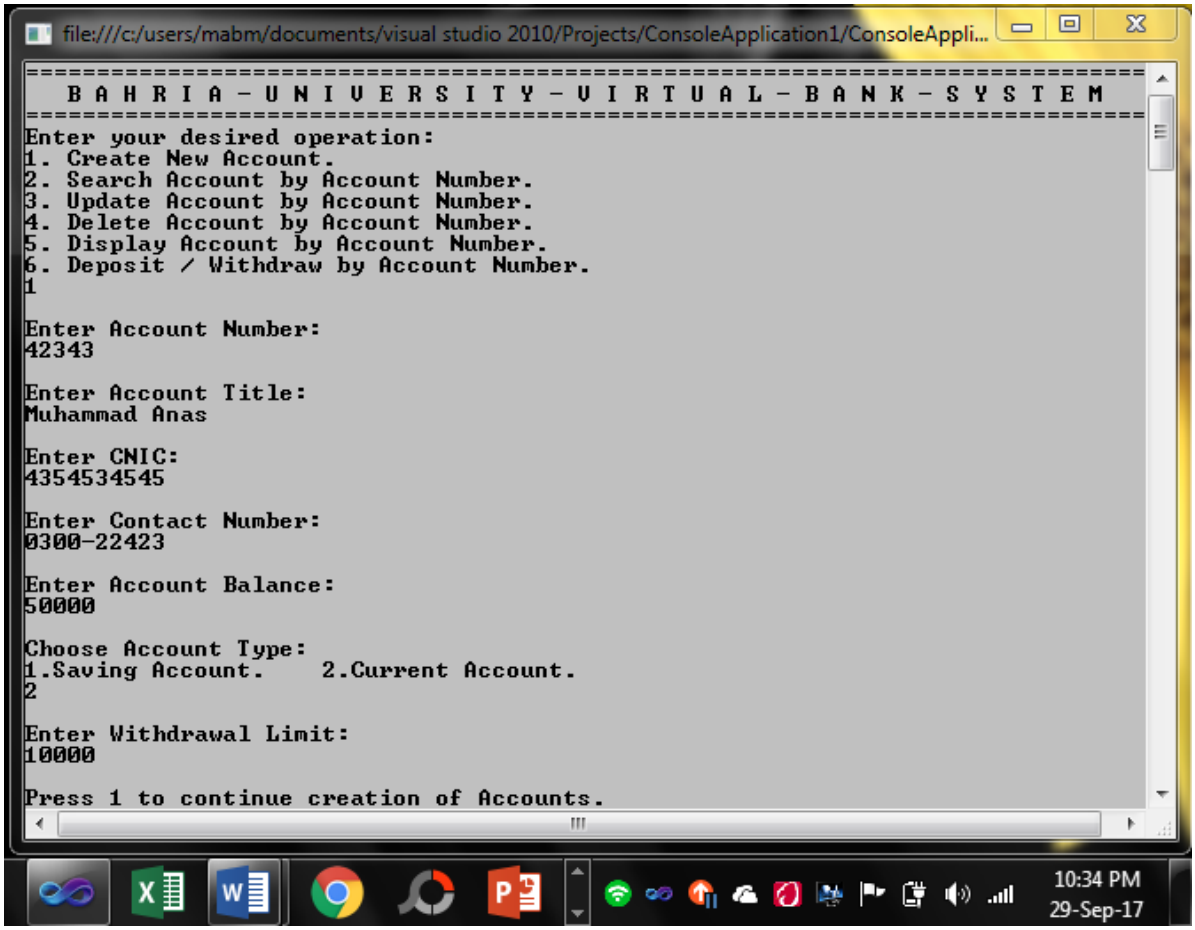
```
51.         public void searchSavingAccount(string An)
52.         {
53.             accountFile obj = new accountFile();
54.             ArrayList accounts = obj.readAllSavingAccount(); //array list: saves diffeent types of
     objects
55.             int totalAccounts = accounts.Count; //built-in count list function
56.             bool found = false;
57.
58.             for (int i = 0; i < totalAccounts; i++)
59.             {
60.                 if ((accounts[i] as savingAccount).accountNoProperty == An)
61.                 {
62.                     found = true;
63.                     Console.WriteLine("\nAccount Found.\n");
64.                     return;
65.                 }
66.             }
67.             Console.WriteLine("\nAccount Not Found.\n");
68.         }
69.     }
70. }
```

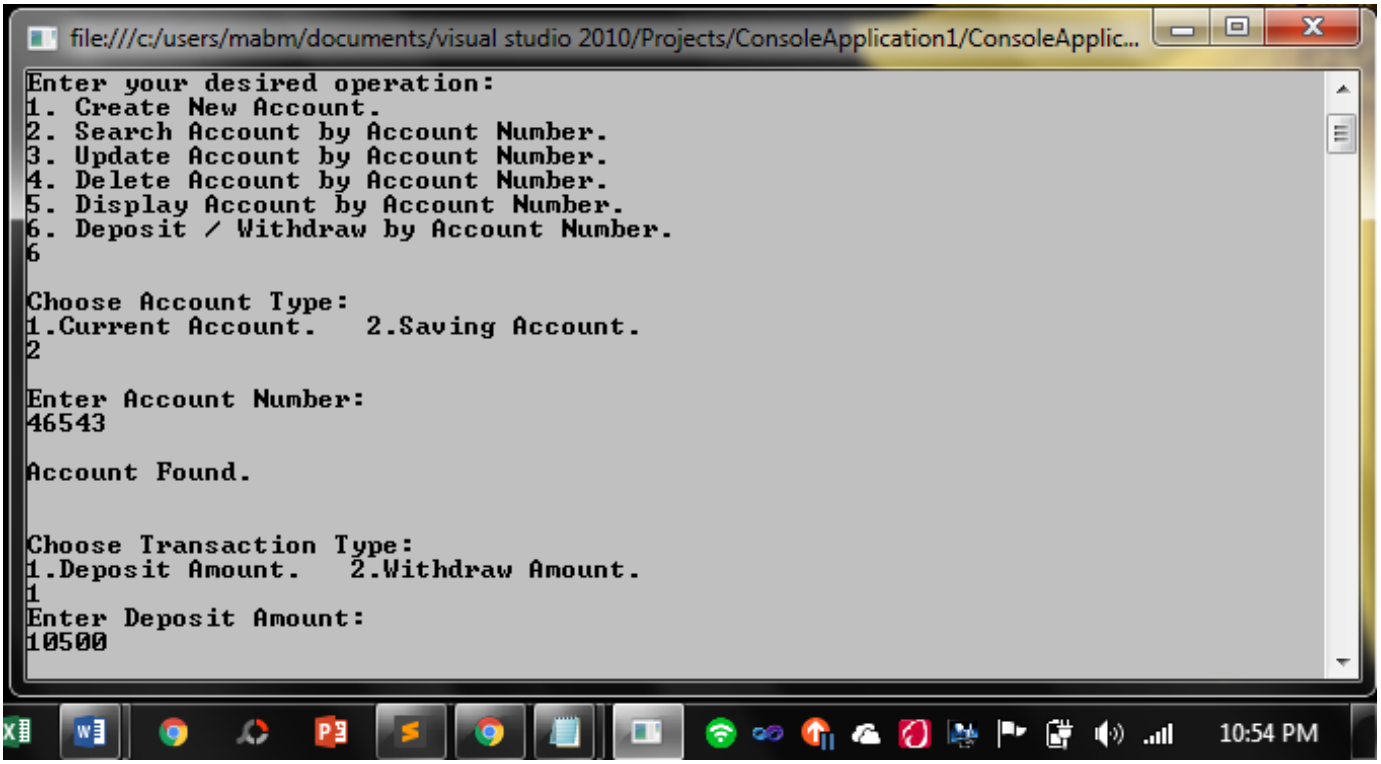
accountFile.cs Class:

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.IO;
6.  using System.Collections;
7.
8.  namespace ConsoleApplication1
9.  {
10.     class accountFile
11.     {
12.         public void writeAllSavingAccount(ArrayList sa)
13.         {
14.             StreamWriter writeSaving = new StreamWriter("SavingAccount.txt"); //write to file obje
    ct
15.             int totalAccount = sa.Count;
16.
17.             for (int i = 0; i < totalAccount; i++)
18.             {
19.                 savingAccount account = sa[i] as savingAccount;
20.                 writeSaving.WriteLine(account.accountNoProperty);
21.                 writeSaving.WriteLine(account.accountTitleProperty);
22.                 writeSaving.WriteLine(account.balanceProperty);
23.                 writeSaving.WriteLine(account.cnicProperty);
24.                 writeSaving.WriteLine(account.contactNoProperty);
25.                 writeSaving.WriteLine(account.profitPercentageProperty);
26.             }
27.             writeSaving.Close();
28.         }
29.
30.         public void writeAllCurrentAccount(List<currentAccount> ca)
31.         {
32.             StreamWriter writeCurrent = new StreamWriter("CurrentAccount.txt"); //write to file ob
    ject
33.             int totalAccount = ca.Count;
34.
35.             for (int i = 0; i < totalAccount; i++)
36.             {
37.                 currentAccount account = ca[i];
38.                 writeCurrent.WriteLine(account.accountNoProperty);
39.                 writeCurrent.WriteLine(account.accountTitleProperty);
40.                 writeCurrent.WriteLine(account.balanceProperty);
41.                 writeCurrent.WriteLine(account.cnicProperty);
```

```
42.                 writeCurrent.WriteLine(account.contactNoProperty);
43.                 writeCurrent.WriteLine(account.withdrawalLimitProperty);
44.             }
45.             writeCurrent.Close();
46.         }
47.
48.         public ArrayList readAllSavingAccount()
49.         {
50.             StreamReader read = new StreamReader("SavingAccount.txt"); //read from file object
51.             ArrayList savingAccount = new ArrayList();
52.             savingAccount sv = null;
53.
54.             while (!read.EndOfStream)
55.             {
56.                 string Ac = read.ReadLine();
57.                 string At = read.ReadLine();
58.                 double bal = double.Parse(read.ReadLine());
59.                 string cnic = read.ReadLine();
60.                 string cn = read.ReadLine();
61.                 float pp = float.Parse(read.ReadLine());
62.                 sv = new savingAccount(Ac, At, cnic, cn, bal, pp);
63.                 savingAccount.Add(sv);
64.             }
65.             read.Close();
66.             return savingAccount;
67.         }
68.
69.         public List<currentAccount> readAllCurrentAccount()
70.         {
71.             StreamReader read = new StreamReader("CurrentAccount.txt"); //read from file object
72.             List<currentAccount> currentAccount = new List<currentAccount>();
73.             currentAccount ca = null;
74.
75.             while (!read.EndOfStream)
76.             {
77.                 string Ac = read.ReadLine();
78.                 string At = read.ReadLine();
79.                 double bal = double.Parse(read.ReadLine());
80.                 string cnic = read.ReadLine();
81.                 string cn = read.ReadLine();
82.                 double wl = double.Parse(read.ReadLine());
83.                 ca = new currentAccount(Ac, At, cnic, cn, bal, wl);
84.                 currentAccount.Add(ca);
85.             }
86.             read.Close();
87.             return currentAccount;
88.         }
89.     }
90. }
```

Console Output:

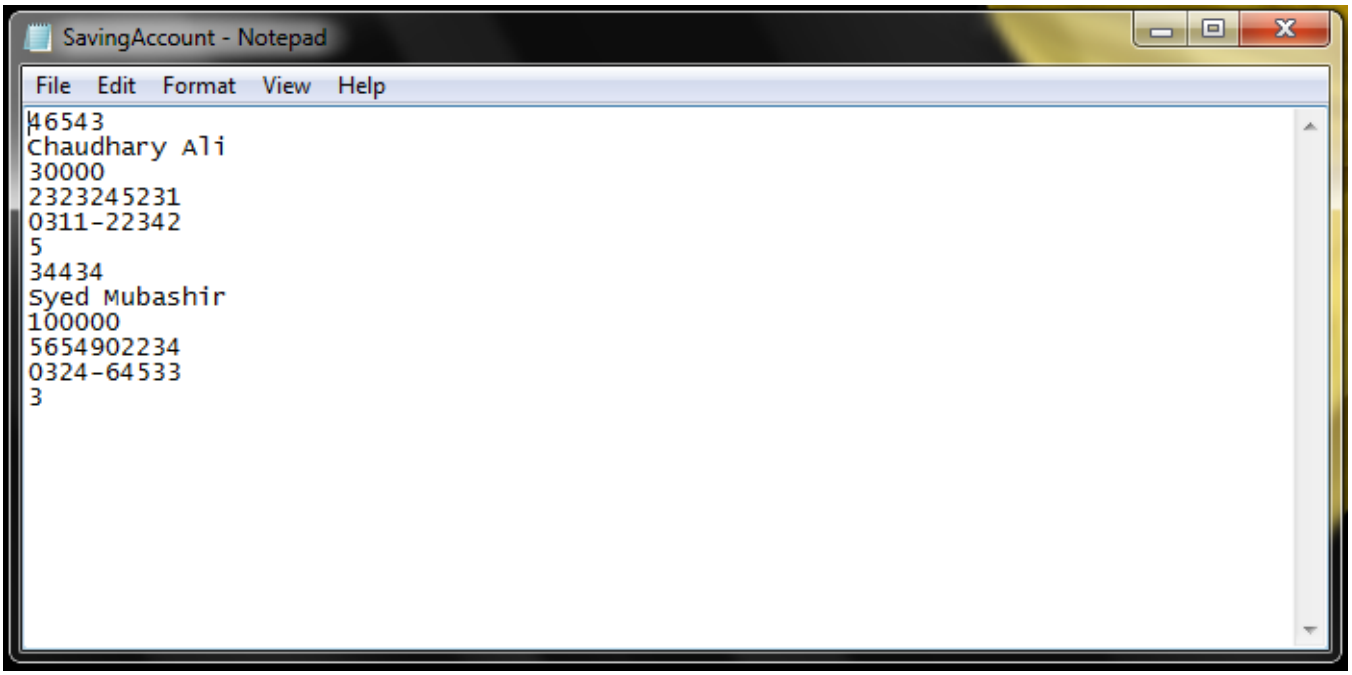




by M. Anas Baig (01-134152-037)

Saved Files:

savingAccount.txt File:



currentAccount.txt File: