Deakin University

Object Oriented Development

OnTrack Submission

# Implementing Multiple Bank Accounts

*Submitted By:*
Connor Gent
gentco
2021/05/03 20:23

*Tutor:*
Nayyar Zaidi

| **Outcome** | **Weight** |
|---|---|
| Evaluate Code | ♦♦♦♦◊ |
| Principles | ♦♦♦♦◊ |
| Build Programs | ♦♦♦♦◊ |
| Design | ♦♦♦♦◊ |
| Justify | ♦♦♦♦◊ |

This task was mainly about implementing the transaction methods in bank class and using it in the banksystem method

May 3, 2021

```
using System;

namespace Connor_Gent_6._2_final
{
    public enum MenuOption

    {

        Withdraw = 1,

        Deposit = 2,

        Transfer = 3,

        AddAccount = 4,

        Print = 5,

        Quit = 6

    }
    class Program
    {
        static MenuOption ReadUserOption()

        {

            int choice = 0;

            do

            {

                Console.WriteLine("1. Withdraw");

                Console.WriteLine("2. Deposit");

                Console.WriteLine("3. Transfer");

                Console.WriteLine("4. Add new account");

                Console.WriteLine("5. Print");

                Console.WriteLine("6. Quit");

                Console.WriteLine("Enter choice: ");

                try

                {

                    choice = Convert.ToInt32(Console.ReadLine());

```

```
54                  }
55
56                  catch (Exception e) { }
57
58              } while (choice < 1 || choice > 6);
59
60              return (MenuOption)choice;
61
62          }
63
64          static void Main(string[] args)
65
66          {
67
68              Bank bank = new Bank();
69
70              Account Jason = new Account(420, "Jason");
71
72              bank.AddAccount(Jason);
73
74              Jason.Deposit(200);
75
76              Jason.Withdraw(500);
77
78              Jason.Print();
79
80              Account James = new Account(420, "James");
81
82              bank.AddAccount(James);
83
84              James.Deposit(300);
85
86              James.Withdraw(40);
87
88              James.Print();
89
90              while (true)
91
92              {
93
94                  switch (ReadUserOption())
95
96                  {
97
98                      case MenuOption.Withdraw:
99
100                         DoWithdraw(bank);
101
102                         break;
103
104                     case MenuOption.Deposit:
105
106                         DoDeposit(bank);
```

```
                    break;

                case MenuOption.Transfer:

                    DoTransfer(bank);

                    break;

                case MenuOption.AddAccount:

                    bank.AddAccount(GetAccount());

                    break;

                case MenuOption.Print:

                    DoPrint(bank);

                    break;

                case MenuOption.Quit:

                    Environment.Exit(0);

                    break;

                default:

                    Jason.Quit();

                    break;

            }

        }

    }

    static Account GetAccount()

    {

        Console.WriteLine("Enter account name: ");

        string name = Console.ReadLine();

        Console.WriteLine("Enter starting balance: ");

        decimal balance = Convert.ToDecimal(Console.ReadLine());

        return new Account(balance, name);

```

```
160         }
161
162         static Account FindAccount(Bank bank)
163
164         {
165
166             Console.WriteLine("Enter account name: ");
167
168             string name = Console.ReadLine();
169
170             var account = bank.GetAccount(name);
171
172             if (account == null)
173
174             {
175
176                 Console.WriteLine("Account wiht name " + name + " not found");
177
178             }
179
180             return account;
181
182         }
183
184         static void DoWithdraw(Bank bank)
185
186         {
187
188             var account = FindAccount(bank);
189
190             if (account == null)
191
192                 return;
193
194             Console.WriteLine("Enter value: ");
195
196             decimal amount = Convert.ToDecimal(Console.ReadLine());
197
198             WithdrawTransaction withdrawTransaction = new
                ↪  WithdrawTransaction(account, amount);
199
200             bank.ExecuteTransaction(withdrawTransaction);
201
202         }
203
204         static void DoDeposit(Bank bank)
205
206         {
207
208             var account = FindAccount(bank);
209
210             if (account == null)
211
```

```
212                 return;
213
214             Console.WriteLine("Enter value: ");
215
216             decimal amount = Convert.ToDecimal(Console.ReadLine());
217
218             DepositTransaction depositTransaction = new DepositTransaction(account,
                ↪  amount);
219
220             bank.ExecuteTransaction(depositTransaction);
221
222         }
223
224         static void DoTransfer(Bank bank)
225
226         {
227
228             Console.WriteLine("From Account:");
229
230             var fromAccount = FindAccount(bank);
231
232             if (fromAccount == null)
233
234                 return;
235
236             Console.WriteLine("To Account:");
237
238             var toAccount = FindAccount(bank);
239
240             if (toAccount == null)
241
242             {
243
244                 return;
245
246             }
247
248             Console.WriteLine("Enter amount: ");
249
250             decimal amount = Convert.ToDecimal(Console.ReadLine());
251
252             TransferTransaction transferTransaction = new
                ↪  TransferTransaction(toAccount, fromAccount, amount);
253
254             bank.ExecuteTransaction(transferTransaction);
255
256         }
257
258         static void DoPrint(Bank bank)
259
260         {
261
262             var account = FindAccount(bank);
```

```
263
264             if (account != null)
265
266             {
267
268                 account.Print();
269
270             }
271
272             else
273
274             {
275
276                 Console.WriteLine("Account not found");
277
278             }
279
280         }
281     }
282 }
```

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

namespace Connor_Gent_6._2_final
{
    class Bank
    {
        private List<Account> accountList;

        public Bank()

        {

            accountList = new List<Account>();

        }

        public void AddAccount(Account account)

        {

            accountList.Add(account);

        }

        public Account GetAccount(String name)

        {

            return accountList.FirstOrDefault(a => a.Name == name);

        }

        public void ExecuteTransaction(DepositTransaction transaction)

        {

            transaction.Execute();

        }

        public void ExecuteTransaction(WithdrawTransaction transaction)

        {

            transaction.Execute();

        }

        public void ExecuteTransaction(TransferTransaction transaction)
```

```
54
55          {
56
57              transaction.Execute();
58
59          }
60
61      }
62  }
```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Connor_Gent_6._2_final
{
    class Account
    {
        public decimal Balance { get; set; }
        public String Name { get; set; }
        public Account(decimal balance, string name)
        {
            Balance = balance;
            Name = name;
        }

        public bool Deposit(decimal amount)
        {
            if(amount <= 0)
            {
                Console.WriteLine("Deposit not successful, Please enter a valid
                ↪  value");
                return false;
            }
            Balance += amount;
            Console.WriteLine("The new balance is " + Balance);
            return true;
        }

        public bool Withdraw(decimal amount)

        {

            if (amount <= 0 || amount > Balance)

            {

                Console.WriteLine("Withdraw not successful, Please enter a valid
                ↪  value");

                return false;

            }

            Balance -= amount;

            Console.WriteLine("The new balance is " + Balance);

            return true;

        }
```

```
52
53          public void Print()
54
55          {
56
57              Console.WriteLine("The balance is " + Balance);
58
59              Console.WriteLine("This account belongs to " + Name);
60
61          }
62
63          public void Quit()
64
65          {
66
67              Environment.Exit(0);
68
69          }
70
71      }
72  }
```

```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Connor_Gent_6._2_final
{
    class WithdrawTransaction
    {
        private Account _account;

        public decimal _amount;

        public bool _executed;

        public bool _success;

        public bool _reversed;

        public bool Executed { get { return _executed; } }

        public bool Success { get { return _success; } }

        public bool Reversed { get { return _reversed; } }

        public WithdrawTransaction(Account account, decimal amount)

        {

            this._account = account;

            this._amount = amount;

        }

        public void Print()

        {

            Console.WriteLine("Transaction Successful: " + Executed + "\nWithdrawn:
            ↪  " + _amount +

                " from " + _account.Name);

        }

        public void Rollback()

        {

            try

```

```
53              {
54
55                  if (Success == false)
56
57                  {
58
59                      throw new InvalidOperationException("Transaction not
                        ↪  successful");
60
61                  }
62
63                  if (Reversed)
64
65                  {
66
67                      throw new InvalidOperationException("This operation cannot be
                        ↪  done again");
68
69                  }
70
71                  else
72
73                  {
74
75                      _account.Balance += _amount;
76
77                      _reversed = true;
78
79                  }
80
81              }
82
83              catch (InvalidOperationException e)
84
85              {
86
87                  Console.WriteLine("The following error detected: " +
                    ↪  e.GetType().ToString() +
88
89                      " with message \"" + e.Message + "\"");
90
91              }
92
93          }
94
95          public void Execute()
96
97          {
98
99              try
100
101             {
102
```

```
103                 if (_amount > _account.Balance)
104 
105                 {
106 
107                     throw new InvalidOperationException("Insufficient funds");
108 
109                 }
110 
111                 if (Executed)
112 
113                 {
114 
115                     throw new InvalidOperationException("Transaction already
                        ↪  attempted");
116 
117                 }
118 
119                 if (_amount < 0)
120 
121                 {
122 
123                     throw new InvalidOperationException("Please enter a valid
                        ↪  amount");
124 
125                 }
126 
127                 else
128 
129                 {
130 
131                     _account.Balance -= _amount;
132 
133                     _success = true;
134 
135                     _executed = true;
136 
137                     Print();
138 
139                 }
140 
141             }
142 
143             catch (InvalidOperationException e)
144 
145             {
146 
147                 Console.WriteLine("The following error detected: " +
                    ↪  e.GetType().ToString() +
148 
149                     " with message \"" + e.Message + "\"");
150 
151             }
152 
```

```
153             }
154
155         }
156     }
```

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

namespace Connor_Gent_6._2_final
{
    class DepositTransaction
    {
        private Account _account;

        public decimal _amount;

        public bool _executed;

        public bool _success;

        public bool _reversed;

        public bool Executed { get { return _executed; } }

        public bool Success { get { return _success; } }

        public bool Reversed { get { return _reversed; } }

        public DepositTransaction(Account account, decimal amount)

        {

            this._account = account;

            this._amount = amount;

        }

        public void Print()

        {

            Console.WriteLine("Transaction Successful: " + Executed + "\nDeposited:
            ↪  " + _amount +

                " to " + _account.Name);

        }

        public void Rollback()

        {

            try

```

```
53              {
54
55                  if (Success == false)
56
57                  {
58
59                      throw new InvalidOperationException("Transaction not
                        ↪  successful");
60
61                  }
62
63                  if (Reversed)
64
65                  {
66
67                      throw new InvalidOperationException("This operation cannot be
                        ↪  done again");
68
69                  }
70
71                  else
72
73                  {
74
75                      _account.Balance -= _amount;
76
77                      _reversed = true;
78
79                  }
80
81              }
82
83              catch (InvalidOperationException e)
84
85              {
86
87                  Console.WriteLine("The following error detected: " +
                    ↪  e.GetType().ToString() +
88
89                      " with message \"" + e.Message + "\"");
90
91              }
92
93          }
94
95          public void Execute()
96
97          {
98
99              try
100
101             {
102                 if (_amount < 0)
```

```
103
104                 {
105
106                     throw new InvalidOperationException("Please enter valid
                        ↪ amount");
107                 }
108                 if (Executed)
109                 {
110
111                     throw new InvalidOperationException("Transaction already
                        ↪ attempted");
112
113                 }
114
115                 else
116
117                 {
118
119                     _account.Balance += _amount;
120
121                     _success = true;
122
123                     _executed = true;
124
125                     Print();
126
127                 }
128
129             }
130
131             catch (InvalidOperationException e)
132
133             {
134
135                 Console.WriteLine("The following error detected: " +
                    ↪ e.GetType().ToString() +
136
137                     " with message \"" + e.Message + "\"");
138
139             }
140
141         }
142
143     }
144 }
```

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

namespace Connor_Gent_6._2_final
{
    class TransferTransaction
    {
        private Account _toaccount, _fromaccount;

        public decimal _amount;

        public bool _executed;

        public bool _success;

        public bool _reversed;

        public bool Executed { get { return _executed; } }

        public bool Success { get { return _success; } }

        public bool Reversed { get { return _reversed; } }

        public TransferTransaction(Account toaccount, Account fromaccount, decimal
        ↪ amount)

        {

            this._toaccount = toaccount;

            this._fromaccount = fromaccount;

            this._amount = amount;

        }

        public void Print()

        {

            Console.WriteLine("Transaction Successful: " + Executed +
            ↪ "\nTransferred: " + _amount +

                " from " + _fromaccount.Name + " to " + _toaccount.Name);

        }

        public void Rollback()

        {
```

```
52
53              try
54
55              {
56
57                  if (Success == false)
58
59                  {
60
61                      throw new InvalidOperationException("Transaction not
                        ↪  successful");
62
63                  }
64
65                  if (Reversed)
66
67                  {
68
69                      throw new InvalidOperationException("This operation cabbit be
                        ↪  done again");
70
71                  }
72
73                  if (_amount > _toaccount.Balance)
74
75                  {
76
77                      throw new InvalidOperationException("The " + _toaccount.Name +
                        ↪  " acount does not have enough money");
78
79                  }
80
81                  else
82
83                  {
84
85                      WithdrawTransaction depositRollback = new
                        ↪  WithdrawTransaction(_toaccount, _amount);
86
87                      // Call execute method to withdraw the amount from toaccount
88
89                      depositRollback.Execute();
90
91                      DepositTransaction withdrawRollback = new
                        ↪  DepositTransaction(_fromaccount, _amount);
92
93                      // Call execute method to deposit the amount to fromaccount
94
95                      withdrawRollback.Execute();
96
97                      _reversed = true;
98
99                  }
```

```
100
101             }
102
103             catch (InvalidOperationException e)
104
105             {
106
107                 Console.WriteLine("The following error detected: " +
                    ↪ e.GetType().ToString() +
108
109                     " with message \"" + e.Message + "\"");
110
111             }
112
113         }
114
115         public void Execute()
116
117         {
118
119             try
120
121             {
122
123                 if (_amount > _fromaccount.Balance)
124
125                 {
126
127                     throw new InvalidOperationException("Insufficient funds");
128
129                 }
130
131                 if (Executed)
132
133                 {
134
135                     throw new InvalidOperationException("Transaction already
                        ↪ attempted");
136
137                 }
138
139                 else
140
141                 {
142
143                     WithdrawTransaction withdrawTransaction = new
                        ↪ WithdrawTransaction(_fromaccount, _amount);
144
145                     withdrawTransaction.Execute();
146
147
148
149                     // If withdraw transaction is successful, do the deposit
```

```
150
151                     if (withdrawTransaction.Success)
152
153                     {
154
155                         DepositTransaction depositTransactionTest = new
                            ↪ DepositTransaction(_toaccount, _amount);
156
157                         depositTransactionTest.Execute();
158
159                         // If deposit also successful
160
161                         if (depositTransactionTest.Success)
162
163                         {
164
165                             _success = true;
166
167                             _executed = true;
168
169                             Print();
170
171                         }
172
173                         else  // Deposit falied, rollback the withdrawTransaction
174
175                         {
176
177                             withdrawTransaction.Rollback();
178
179                         }
180
181                     }
182
183                 }
184
185             }
186
187             catch (InvalidOperationException e)
188
189             {
190
191                 Console.WriteLine("The following error detected: " +
                    ↪ e.GetType().ToString() +
192
193                     " with message \"" + e.Message + "\"");
194
195             }
196
197         }
198
199     }
200 }
```