Deakin University

Object Oriented Development

OnTrack Submission

Validating Accounts

Submitted By:
Peter Stacey
pstacey
2020/03/21 08:40

Tutor:
Dipto Pratyaksa

Outcome	Weight
Evaluate Code	♦♦◊◊◊
Principles	♦♦◊◊◊
Build Programs	♦♦♦◊◊
Design	♦♦♦◊◊
Justify	♦♦♦◊◊

The task involves evaluating both the existing code and the new requirements, to design additional features and changes to the existing design from task 2.2. Additionally, it involves applying the principles of object oriented programming, and specifically encapsulation - by hiding the implementation of the underlying types in private variables with a public interface for users of the program. To implement the changes, there is a reasonable amount of new code, in addition to a small number of code changes and between the submitted code and the video I will link, this aligns well with meeting the outcomes of the subject.

March 21, 2020

```
using System;

namespace Task_3._2P
{
    enum MenuOption
    {
        Withdraw,
        Deposit,
        Print,
        Quit
    }

    class BankSystem
    {
        // Reads string input in the console
        /// <summary>
        /// Reads string input in the console
        /// </summary>
        /// <returns>
        /// The string input of the user
        /// </returns>
        /// <param name="prompt">The string prompt for the user</param>
        public static String ReadString(String prompt)
        {
            Console.Write(prompt + ": ");
            return Console.ReadLine();
        }

        // Reads integer input in the console
        /// <summary>
        /// Reads integerinput in the console
        /// </summary>
        /// <returns>
        /// The input of the user as an integer
        /// </returns>
        /// <param name="prompt">The string prompt for the user</param>
        public static int ReadInteger(String prompt)
        {
            int number = 0;
            string numberInput = ReadString(prompt);
            while (!(int.TryParse(numberInput, out number)))
            {
                Console.WriteLine("Please enter a whole number");
                numberInput = ReadString(prompt);
            }
            return Convert.ToInt32(numberInput);
        }

        // Reads integer input in the console between two numbers
        /// <summary>
        /// Reads integer input in the console between two numbers
        /// </summary>
        /// <returns>
```

```
54          /// The input of the user as an integer
55          /// </returns>
56          /// <param name="prompt">The string prompt for the user</param>
57          /// <param name="minimum">The minimum number allowed</param>
58          /// <param name="maximum">The maximum number allowed</param>
59          public static int ReadInteger(String prompt, int minimum, int maximum)
60          {
61              int number = ReadInteger(prompt);
62              while (number < minimum || number > maximum)
63              {
64                  Console.WriteLine("Please enter a whole number from " +
65                                    minimum + " to " + maximum);
66                  number = ReadInteger(prompt);
67              }
68              return number;
69          }
70
71          // Reads decimal input in the console
72          /// <summary>
73          /// Reads decimal input in the console
74          /// </summary>
75          /// <returns>
76          /// The input of the user as a decimal
77          /// </returns>
78          /// <param name="prompt">The string prompt for the user</param>
79          public static decimal ReadDecimal(String prompt)
80          {
81              decimal number = 0;
82              string numberInput = ReadString(prompt);
83              while (!(decimal.TryParse(numberInput, out number)))
84              {
85                  Console.WriteLine("Please enter a decimal number");
86                  numberInput = ReadString(prompt);
87              }
88              return Convert.ToDecimal(numberInput);
89          }
90
91          /// <summary>
92          /// Displays a menu of possible actions for the user to choose
93          /// </summary>
94          private static void DisplayMenu()
95          {
96              Console.WriteLine("\n**********************");
97              Console.WriteLine("*       Menu         *");
98              Console.WriteLine("**********************");
99              Console.WriteLine("*  1. Withdraw       *");
100             Console.WriteLine("*  2. Deposit        *");
101             Console.WriteLine("*  3. Print          *");
102             Console.WriteLine("*  4. Quit           *");
103             Console.WriteLine("**********************");
104         }
105
106         private static void DisplayResult(MenuOption action, Boolean result)
```

```
107         {
108             String output = action + " "
109                 + (result == true ? "succeeded" : "failed. Invalid amount.");
110             Console.WriteLine(output);
111         }
112 
113         /// <summary>
114         /// Returns a menu option chosen by the user
115         /// </summary>
116         /// <returns>
117         /// MenuOption chosen by the user
118         /// </returns>
119         static MenuOption ReadUserOption()
120         {
121             DisplayMenu();
122             int option = ReadInteger("Choose an option", 1,
123                 Enum.GetNames(typeof(MenuOption)).Length);
124             return (MenuOption)(option - 1);
125         }
126 
127         /// <summary>
128         /// Attempts to deposit funds into an account
129         /// </summary>
130         /// <param name="account">The account to deposit into</param>
131         static void DoDeposit(Account account)
132         {
133             decimal amount = ReadDecimal("Enter the amount");
134             bool result = account.Deposit(amount);
135             DisplayResult(MenuOption.Deposit, result);
136         }
137 
138         /// <summary>
139         /// Attempts to withdraw funds from an account
140         /// </summary>
141         /// <param name="account">The account to withdraw from</param>
142         static void DoWithdraw(Account account)
143         {
144             decimal amount = ReadDecimal("Enter the amount");
145             Boolean result = account.Withdraw(amount);
146             DisplayResult(MenuOption.Withdraw, result);
147         }
148 
149         /// <summary>
150         /// Outputs the account name and balance
151         /// </summary>
152         /// <param name="account">The account to print</param>
153         static void DoPrint(Account account)
154         {
155             account.Print();
156         }
157 
158         static void Main(string[] args)
159         {
```

```
160             Account acc = new Account("Peter Stacey", -100);
161             acc.Print(); // confirm balance not set to negative
162 
163             do
164             {
165                 MenuOption chosen = ReadUserOption();
166                 switch (chosen)
167                 {
168                     case MenuOption.Withdraw:
169                         DoWithdraw(acc); break;
170                     case MenuOption.Deposit:
171                         DoDeposit(acc); break;
172                     case MenuOption.Print:
173                         DoPrint(acc); break;
174                     case MenuOption.Quit:
175                     default:
176                         Console.WriteLine("Goodbye");
177                         System.Environment.Exit(0); // terminates the program
178                         break; // unreachable
179                 }
180             } while (true);
181         }
182     }
183 }
```

```cs
using System;

namespace Task_3._2P
{
    /// <summary>
    /// A bank account class to hold the account name and balance details
    /// </summary>
    class Account
    {
        // Instance variables
        private String _name;
        private decimal _balance;

        // Read-only properties
        public String Name { get { return _name; } }


        /// <summary>
        /// Class constructor
        /// </summary>
        /// <param name="name">The name string for the account</param>
        /// <param name="balance">The decimal balance of the account</param>
        public Account(String name, decimal balance = 0)
        {
            _name = name;
            if (balance <= 0)
                return;
            _balance = balance;
        }

        /// <summary>
        /// Deposits money into the account
        /// </summary>
        /// <returns>
        /// Boolean whether the deposit was successful (true) or not (false)
        /// </returns>
        /// <param name="amount">The decimal amount to add to the balance</param>
        public Boolean Deposit(decimal amount)
        {
            if (amount <= 0)
                return false;

            _balance += amount;
            return true;
        }

        /// <summary>
        /// Withdraws money from the account (with no overdraw protection currently)
        /// </summary>
        /// <returns>
        /// Boolean whether the withdrawal was successful (true) or not (false)
        /// </returns>
        /// <param name="amount">The amount to subtract from the balance</param>
```

```
54          public Boolean Withdraw(decimal amount)
55          {
56              if ((amount <= 0) || (amount > _balance))
57                  return false;
58
59              _balance -= amount;
60              return true;
61          }
62
63          /// <summary>
64          /// Outputs the account name and current balance as a string
65          /// </summary>
66          public void Print()
67          {
68              Console.WriteLine("Account Name: {0}, Balance: {1}",
69                  _name, _balance.ToString("C"));
70          }
71      }
72  }
```