Sample: C# - Banking Application

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

namespace Task76102
{
 enum AccountType { Checking, Savings, Certificate }


 class Program
 {
 static void Main(string[] args) //Entry point of the program
 {

 //Several accounts with different account types
 List<Account> acc = new List<Account>(); //List collection to store accounts
 acc.Add( new Account("Tom Patric", AccountType.Certificate, 1000523));
 acc.Add( new Account("John Smith", AccountType.Savings, 1220));
 acc.Add( new Account("Andrew Jameson", AccountType.Checking, 603));
 //
 Start: //Label for returning to the main menu
 int line = 1; // position of the chosen line
 ConsoleKeyInfo c; // this variable holds Key information
 Console.Clear(); //clearing window
 Console.WriteLine("Select one of the next cases:(Esc-exit)");
 Console.WriteLine("Add an account *");
 Console.WriteLine("Select user");
```

```
Console.SetCursorPosition(26, 1); //Set cursor position for menu selecting

//menu loop
do
{
c = Console.ReadKey(true); //getting key


switch (c.Key) // identifying pressed key
{
case ConsoleKey.UpArrow: //UpArrow was pressed
Console.SetCursorPosition(26, line);
Console.Write(" ");
if (line == 1) line = 2;
else line--;
Console.SetCursorPosition(26, line);
Console.Write("*");

break;
case ConsoleKey.DownArrow://DownArrow was pressed
Console.SetCursorPosition(26, line);
Console.Write(" ");
if (line == 2) line = 1;
else line++;
Console.SetCursorPosition(26, line);
Console.Write("*");
break;  case ConsoleKey.Escape: Environment.Exit(0); break; //Exit the
program
}
```

```
} while (c.Key != ConsoleKey.Enter); //end of main menu loop



if (line == 1) // Creating user
{
Console.Clear();
string name = "";
int type = 0; // Account type in digital form
double mon = 0; //Account money
Console.WriteLine("Enter Holder name:");
name = Console.ReadLine();
Console.WriteLine("Enter account type:(1,2,3)");
Console.WriteLine("1.Checking 2.Savings 3.Certificate");
c = Console.ReadKey(true);
if ((byte)c.Key < 52 && (byte)c.Key > 48) //49 - code of the "1" button,
51 - "3"
type = ((byte)c.Key)-49; //checking the input information
Console.WriteLine("Enter deposit sum:");
try
{
mon = Double.Parse(Console.ReadLine()); //converting string to double
}
catch //in the case of incorrect input
{
mon = 0;
}
if (mon < 0) mon = 0;
```

```
 acc.Add(new Account(name, (AccountType)type, mon)); //adding new acccount
to the collection
 Console.WriteLine("Successfully done. Press any key...");
 Console.ReadKey(true);
 goto Start; // goto to the start label

 }
 else // Selecting created users
 {
 do // another loop for the holding a window
 {
 Console.Clear(); //Clearing window
Console.WriteLine("Choose user(ESC - return):"); //Printing accounts
from te collection
 for (int i = 0; i < acc.Count; i++)
 Console.WriteLine(i + 1 + ". " + acc[i].AcctHolderName);



 //Menu start is the same as the main menu
Console.SetCursorPosition(25, 1);
Console.Write("*");
 line = 1;
do
{  c = Console.ReadKey(true);

 switch (c.Key)
{
```

```
 case ConsoleKey.UpArrow:
 Console.SetCursorPosition(25, line);
Console.Write(" ");
 if (line == 1) line = acc.Count;
else line--;
 Console.SetCursorPosition(25, line);
Console.Write("*");

 break;
 case ConsoleKey.DownArrow:
 Console.SetCursorPosition(25, line);
Console.Write(" ");
 if (line == acc.Count) line = 1;
else line++;
 Console.SetCursorPosition(25, line);
 Console.Write("*");
 break;
 case ConsoleKey.Escape: goto Start;
 }
 } while (c.Key != ConsoleKey.Enter);
 //Menu end


 line--; // line = line -1 -> index fix
 Console.Clear();

 //calling functions from the UI class
UI.Display(acc[line]);
UI.Balance(acc[line]);
```

```
UI.ProcessCheck(acc[line]);
//////////////////////////////////////////////////

 Console.WriteLine("\nEnter deposit sum:");
 double mon = 0; //money

 try //construction for the input validation
{
 mon = Double.Parse(Console.ReadLine());
 }
catch //in the case of wrong input
 {
 mon = 0;
 }
if (mon < 0) mon = 0;

 //calling function from the UI class
UI.Deposit(acc[line], mon);

 Console.WriteLine("\nEnter WithDraw sum:");
 mon = 0; //money

 try//construction for the input validation
{
 mon = Double.Parse(Console.ReadLine());  }
catch//in the case of wrong input
{
 mon = 0;
 }
```

```
if (mon < 0) mon = 0;
//calling function from the UI class
UI.WithDraw(acc[line], mon);

 Console.WriteLine("\n Press any key...");
 Console.ReadKey();
 } while (true); //Press esc to return to the main menu

 }




 }
 }



 class Account
 {
 double money=0; // your deposit money closed from the program
(private)
 static int UserNum=0; // general counter for accounts quantity (private)


 // Fields:
 public int AcctNumber { get; private set; }
 public string AcctHolderName { get; private set; }
```

```
public AccountType AcctType { get; private set; }
//

public Account() // Constructor without parameters
{
money = 0;
AcctNumber = ++UserNum;
AcctHolderName = "Guest " + AcctNumber;
AcctType = AccountType.Savings;
}

public Account(string name, AccountType accType, double balance)
{
AcctHolderName = name;
if (name.Length == 0) AcctHolderName = "Guest " + AcctNumber;

AcctType = accType;
if (balance < 0)
{
money = 0;
}
else money = balance;
AcctNumber = ++UserNum;
 } //Constructor with parameters

//Methods

 public Account(Account acc, double balance=0) //Creating new account for existing
user with a new deposit
```

```
{
this.AcctHolderName = acc.AcctHolderName;
this.AcctNumber = ++UserNum;
this.AcctType = acc.AcctType;
this.money = balance;
}

public double Deposit(double money, out bool pass) //returns money quatity after
transaction
//flag "pass" shows correct or uncorrect result of transaction
{
if (money <= 0)
{
pass=false;

}
else
{
this.money += money;
pass = true;
}

return this.money;

}

public double WithDraw(double money, out bool pass) //Parameters are used like in
Deposit() method
{
```

```
if (money <= 0||money>this.money)
{
pass = false;

}
else
{
this.money -= money;
pass = true;
}

return this.money;
}

public bool ProcessCheck() //(i.e. processing a check against the savings
account) - from the task
{
if (AcctType == AccountType.Savings)
return false;
return true;
}

public double CurBalance() //returns current balance  {

return this.money;
}

public void Display() //dispays the main information
{
```

```
Console.WriteLine("Holder name: {0,20}",AcctHolderName);
Console.WriteLine("Account number: {0,17}", AcctNumber);
Console.WriteLine("Account type: {0,19}", AcctType.ToString());
}
//

}


class UI //contains static methods
{

public static void Deposit(Account acc,double money)
{
bool pass; //flag of the correct transaction
double N_money=acc.Deposit(money, out pass);
Console.WriteLine("\n{0}`s Deposit information (+
{1}):",acc.AcctHolderName,money);
if (!pass)
{
Console.WriteLine("Action revoken. Wrong input information.");
}
else
{
Console.WriteLine("Deposit successfully renewed. The
sum:\t{0:###,###.##}$", N_money);
}
}
```

```
public static void WithDraw(Account acc, double money)
{
bool pass;//flag of the correct transaction
money = acc.WithDraw(money, out pass);
Console.WriteLine("\n{0}`s WithDraw information:", acc.AcctHolderName);
if (!pass)
{
Console.WriteLine("Action revoken. Wrong input information or not enough
money.");
}
else
{
Console.WriteLine("Deposit successfully renewed. The
sum:\t{0:###,###.##}$", money);
}
}

public static void Balance(Account acc)
{
if (acc.CurBalance() == 0)
{
Console.WriteLine("Balance: {0,23}$", 0);  return;
}
Console.WriteLine("Balance: {0,23:###,###.##}$", acc.CurBalance());
}

public static void Display(Account acc)
{
acc.Display();
```

```
}

public static void ProcessCheck(Account acc)
{
Console.Write("\n{0} Checking status: ",acc.AcctHolderName);
if(acc.ProcessCheck()) Console.Write("ALLOWED");
else Console.Write("FORBIDDEN");
}

}

}
```