rr

**МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**

**ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Факультет прикладної математики**

**Кафедра програмного забезпечення комп’ютерних систем**

**Лабораторна робота № 2**

**з дисципліни “Об’єктно орієнтоване програмування”**

**тема “C# .Net. Розширені можливості реалізації ООП у мові C#. Події.”**

|  |  |  |
| --- | --- | --- |
| **Виконав(ла)**  **студент(ка) II курсу**  **групи КП-83**  **Коваль Андрій Олекснадрович**  **(прізвище, ім’я, по батькові)** |  | **Перевірив**  **“\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.**  **викладач**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **(прізвище, ім’я, по батькові)** |

**Київ 2019**

**Мета роботи**

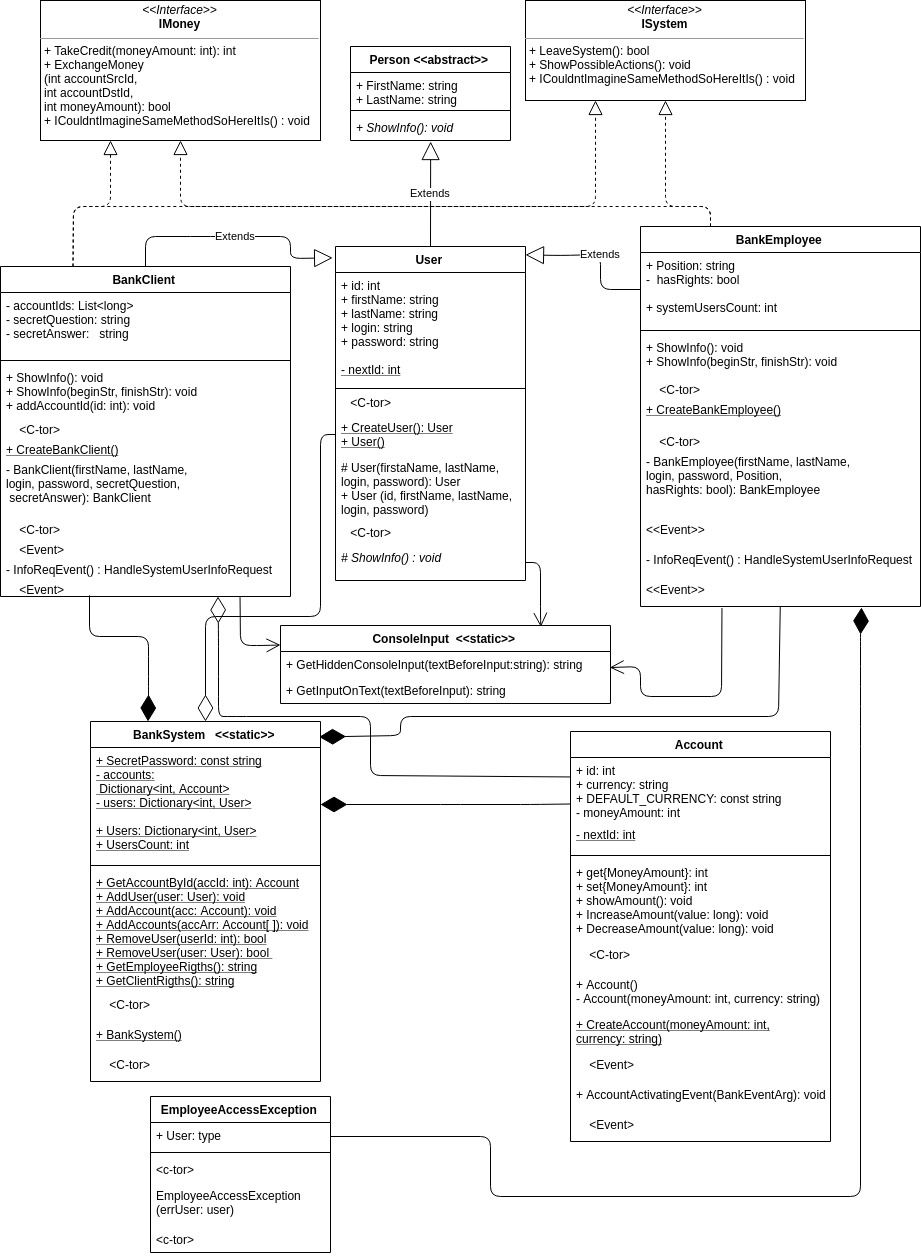
Ознайомитися з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Вивчити механізми обробки подій у C#.

**Постановка завдання**

Для ієрархії класів, побудованої в лабораторній роботі №1, реалізувати:

1. Множину інтерфейсів. При чому один з класів повинен реалізовувати щонайменше 2 інтерфейси. Також продемонструвати реалізацію explicit implementation інтерфейса.
2. Абстрактний клас. Забезпечити його наслідування.
3. Механізм «делегат – подія – обробник події»**.**
4. Механізм створення та обробки власних помилок:
   1. створити новий клас виключної ситуації;
   2. створити новий клас аргументів для передачі їх до обробника виключної ситуації;
   3. забезпечити ініціювання створеної виключної ситуації та продемонструвати, як працює обробник даної помилки.

**Діаграма класів**

****

**Фрагменти коду програми**

|  |
| --- |
| Інтерфейси |
| interface IMoney {  int TakeCredit(int moneyAmount);  bool ExchangeMoney(int accountSrcId, int accountDstId, int moneyAmount);  void ICouldntImagineSameMethodSoHereItIs();  }  interface ISystem {  bool LeaveSystem();  void ShowPossibleActions();  void ICouldntImagineSameMethodSoHereItIs();  } |
| class BankClient : User, IMoney, ISystem  {  …  public int TakeCredit(int moneyValue)  {  if (moneyValue <= 0) return 0;  int accountIdWithCredit = -1;  foreach (var accId in this.accountsIds)  {  var acc = BankSystem.GetAccountById(accId);  if (acc.MoneyAmount > moneyValue)  {  acc.IncreaseAmount(moneyValue);  accountIdWithCredit = (int)acc.id;  break;  }  }  return accountIdWithCredit;  }  public bool ExchangeMoney(int accountSrcId, int accountDstId, int MoneyAmount)  {  var accSrc = BankSystem.GetAccountById(accountSrcId);  var accDst = BankSystem.GetAccountById(accountDstId);  if (!this.accountsIds.Contains(accountSrcId))  return false;  if (accSrc == null || accDst == null)  return false;  if (accSrc.DecreaseAmount(MoneyAmount))  {  accDst.IncreaseAmount(MoneyAmount);  }  else  {  return false;  }  return true;  }  public bool LeaveSystem()  {  return BankSystem.RemoveUser((int)this.id);  }  public void ShowPossibleActions()  {  if (InfoReqEvent != null)  {  string actions = InfoReqEvent();  Console.WriteLine(actions);  }  else  {  Console.WriteLine("No answer from system");  }  }  void IMoney.ICouldntImagineSameMethodSoHereItIs()  {  Console.WriteLine("Some IMoney method overriden in BankCLient");  }  void ISystem.ICouldntImagineSameMethodSoHereItIs()  {  Console.WriteLine("Some ISystem method overriden in BankCLient");  } |

|  |
| --- |
| Абстрактний клас |
| abstract class Person  {  public abstract string FirstName { get; }  public abstract string LastName { get; }  public abstract void ShowInfo();  } |

|  |
| --- |
| Механізм «делегат – подія – обробник події»**.** 1 приклад |
| public class Account  {  ...  private event AccountHandle AccountActivatingEvent;  public Account(long moneyAmount, string currency)  {  this.id = nextId++;  if (moneyAmount < 0)  {  this.moneyAmount = 0;  }  else  {  this.moneyAmount = moneyAmount;  }  this.currency = currency;  this.AccountActivatingEvent += new AccountHandle(BankSystem.AddAccount);  }  public void Activate(string bankPassword) {  Console.WriteLine($"Activating new acconut with id - {this.id}");  if (bankPassword == BankSystem.SecretPassword) {  if (AccountActivatingEvent != null) {  AccountActivatingEvent(new BankEventArg(this));  }  Console.WriteLine("Account succesfully activated");  } else {  Console.WriteLine("ERROR: bank password is wrong, aborting account activating");  }  }  } |
|  |

|  |
| --- |
| Механізм створення та обробки власних помилок |
| class EmployeeAccessExceptionArgs  {  public User User;  public string Message;  public EmployeeAccessExceptionArgs(string Msg, User user) {  this.Message = Msg;  this.User = user;  }  public EmployeeAccessExceptionArgs(User user, string Msg = "") {  this.Message = Msg;  this.User = user;  }  }  class EmployeeAccessException : Exception  {  public EmployeeAccessExceptionArgs args;  public EmployeeAccessException(EmployeeAccessExceptionArgs args) : base(args.Message)  {  this.args = args;  }  } |
| public int systemUsersCount  {  get  {  if (this.hasRights)  {  return BankSystem.UsersСount;  }  else  {  throw new EmployeeAccessException(new EmployeeAccessExceptionArgs(this, "Current employee has no rights"));  }  }  } |

**Висновки**

Ознайомився з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати та вивчив механізми обробки подій у C#.