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#.

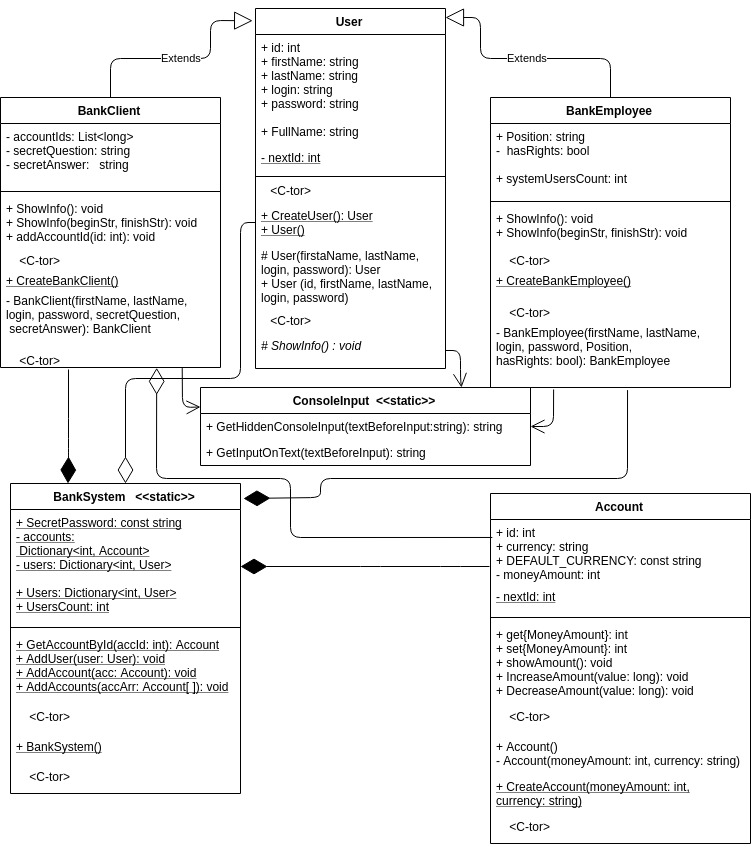
**Постановка завдання**

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**

[Посилання](https://www.draw.io/?lightbox=1&highlight=0000ff&edit=_blank&layers=1&nav=1&title=lab1.svg#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1tQcz9VJlao5VFvyhBjaUR_kD8odUx-iS%26export%3Ddownload) на якісне зображення

****

**Фрагменти коду програми**

|  |
| --- |
| readonly та protected поля |
| namespace Human  {  // basic user with no rights  class User  {  public readonly long id;  public readonly string firstName;  public readonly string lastName;  public string login { get; }  public string password { get; }  protected static long nextId;  //..  } |

|  |
| --- |
| get та set |
| public long MoneyAmount  {  get { return moneyAmount; }  set  {  if (value < 0)  {  Console.WriteLine("Error: invalid money amount value");  return;  }  moneyAmount = value;  }  } |

|  |
| --- |
| Деякі конструктори |
| public Account()  {  this.moneyAmount = 0;  this.currency = DEFAULT\_CURRENCY;  this.id = nextId++;  }  public static Account CreateAccount(long moneyAmount, string currency)  {  Account acc = new Account(moneyAmount, currency);  BankSystem.AddAccount(acc);  return acc;  }  private Account(long moneyAmount, string currency)  {  this.id = nextId++;  if (moneyAmount < 0)  {  this.moneyAmount = 0;  }  else  {  this.moneyAmount = moneyAmount;  }  this.currency = currency;  } |
| private User() { }  public static User CreateUser()  {  string firstName = ConsoleInput.GetInputOnText("Enter first name");  string lastName = ConsoleInput.GetInputOnText("Enter last name");  string login = ConsoleInput.GetInputOnText("Enter login");  string password = "";  do  {  password = ConsoleInput.GetHiddenConsoleInput("Enter password");  string passwordToCheck = ConsoleInput.GetHiddenConsoleInput("Enter password again");  if (passwordToCheck != password)  {  Console.WriteLine("Error: passwords differ");  continue;  }  break;  } while (true);  User newUser = new User(firstName, lastName, login, password);  return newUser;  }  static User()  {  nextId = 0;  }  // protected for inherited classes  protected User(string firstName, string lastName, string login, string password)  {  this.id = nextId++;  this.firstName = firstName;  this.lastName = lastName;  this.password = password;  }  // for explicit creating user  public User(long id, string firstName, string lastName, string login, string password)  {  if (id < nextId)  {  Console.WriteLine($"Unable to create user with id {id}");  Console.WriteLine($"Assigning id value {nextId}");  id = nextId;  }  this.id = id;  long newNextId = id + 1;  nextId = newNextId;  this.firstName = firstName;  this.lastName = lastName;  this.login = login;  this.password = password;  } |

|  |
| --- |
| virtual та override |
| //method to override  public virtual void ShowInfo()  {  Console.WriteLine($"User info {this.id} - '{this.FullName}'");  } |
| public override void ShowInfo()  {  Console.WriteLine("Client info-------");  base.ShowInfo();  foreach (var accId in accountsIds)  {  Bank.BankSystem.GetAccountById(accId).ShowAmount();  }  Console.WriteLine("Client info-------");  }  public void ShowInfo(string beginStr, string finishStr)  {  Console.WriteLine(beginStr);  base.ShowInfo();  foreach (var accId in accountsIds)  {  Bank.BankSystem.GetAccountById(accId).ShowAmount();  }  Console.WriteLine(finishStr);  } |
| public override void ShowInfo()  {  Console.WriteLine("Employee info-----");  base.ShowInfo();  Console.WriteLine($"Position {this.Position}");  Console.WriteLine("Employee info-----");  }  public void ShowInfo(string beginStr, string finishStr)  {  Console.WriteLine(beginStr);  base.ShowInfo();  Console.WriteLine($"Position {this.Position}");  Console.WriteLine(finishStr);  } |

**Висновки**

Я ознайомився з основами об’єктно-орієнтованого підходу до створення ПЗ у мові С#, створенням класів, об’єктів, механізмами інкапсуляції, наслідування та поліморфізму. Завдяки новим знанням я побудував невелику ієрархію класів з відношеннями між ними.