++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 № 3**

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

**тема “C# .Net. Колекції. Серіалізація”**

|  |  |  |
| --- | --- | --- |
| **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 можливостями мови C# щодо обробки колекцій даних, їх серіалізації, а також з анонімними методами, методами-розширеннями.

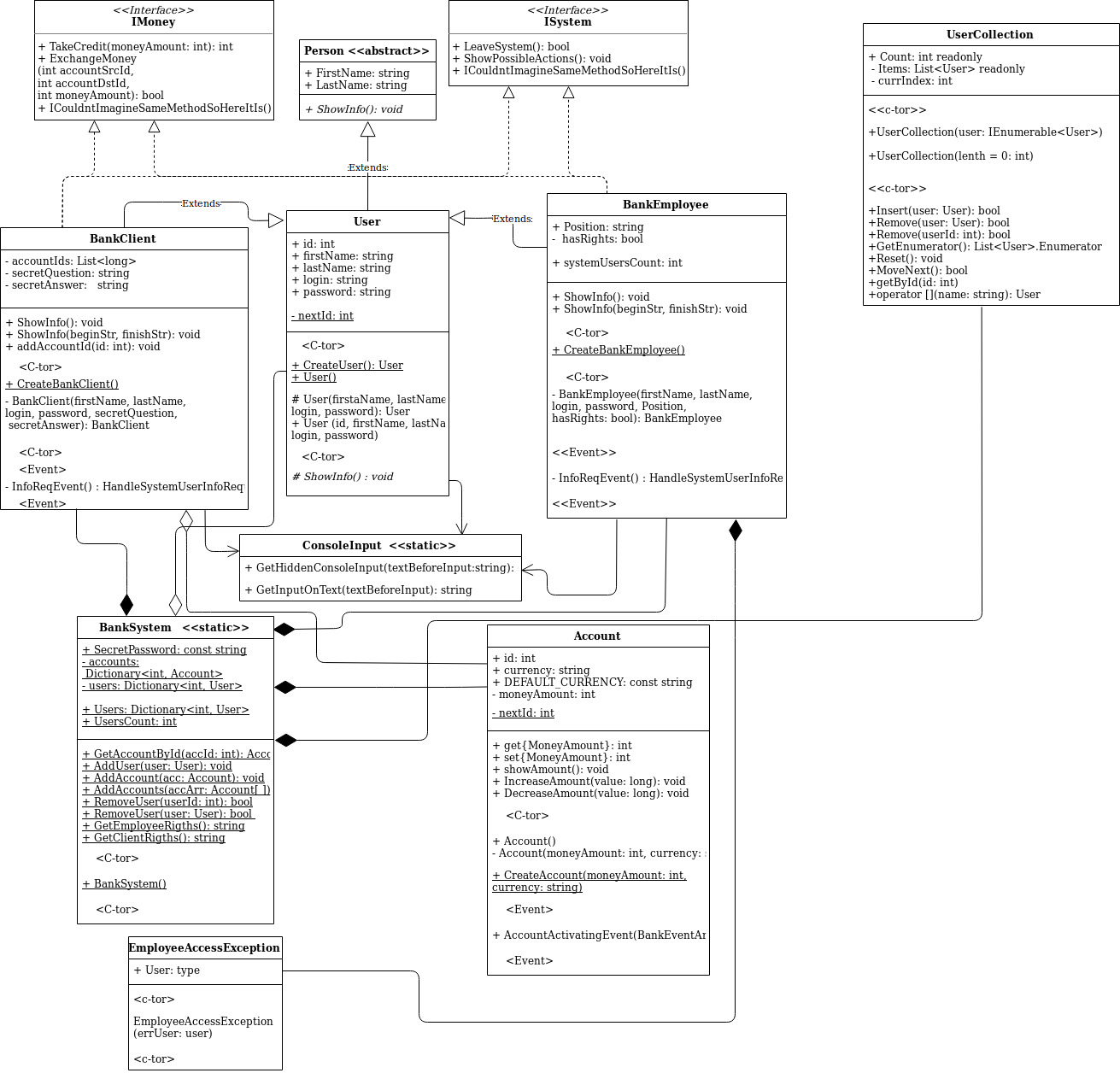
**Постановка завдання**

1. Перетворити код, який забезпечує роботу з подіями та обробниками подій в лабораторній роботі №2, на код, що використовує (\*):
   1. анонімні методи;
   2. lambda-вирази;
   3. типи Action та Func (кожен з них).

(\*) - допускається реалізація коду однієї події різними способами, необов”язково різних подій.

1. Ввести до класів (розроблених у попередніх лабораторних роботах) колекції об’єктів (використати уже існуючі типи даних), продемонструвати роботу з цими даними (додавання, видалення, пошук).
2. Створити власний тип – колекцію об’єктів. Розробити індексатори для звертання до необхідних елементів (з індексом типу не int!).
3. Реалізувати IComparable, IEnumerable, IEnumerator інтерфейси для забезпечення можливостей сортування елементів та їх обходу.
4. Створити метод-розширення класу-колекції.
5. Реалізувати у коді можливість запису поточного стану об’єктів та масивів об’єктів у файл та читання його з файлу (серіалізація). Продемонструвати роботу двох видів серіалізаторів.
6. Створити власний тип даних на основі типів-узагальнень Generic<T>.

**Діаграма класів**



**Фрагменти коду програми**

|  |
| --- |
| Анонімні функції та Action з Func |
| GetEmployeeRigths = () =  {  return "\nAddAccountId()\n" +  "TakeCredit(moneyValue) - the money is assigned to the first account with the same or more money amount\n" +  "ShowInfo() - show employee info\n" +  "systemUsersCount - get users count in system. Permission reqired\n";  }; |
| GetClientRigths = delegate()  {  return "\nAddAccountId()\n" +  "TakeCredit(moneyValue) - the money is assigned to the first account with the same or more money amount\n" +  "ShowInfo() - show user info\n";  }; |
| public static Action<BankEventArg> AddAccountOnEvent;  public static Func<string> GetEmployeeRigths;  public static Func<string> GetClientRigths; |

|  |
| --- |
| Існуючі колекції |
| private static Dictionary<int, Account> accounts; |

|  |
| --- |
| Власна колекція UserCollection та розширення її |
| public class UserCollection : IEnumerable, IEnumerator  {  public int Count { get { return this.items.Count; } }  private List<User> items; //@todo ask about LinkedList and so on  private int currIndex;  public List<User> Values {  get => this.items;  }  public UserCollection(int length = 0)  {  this.items = new List<User>(length);  this.currIndex = 0;  }  public UserCollection(IEnumerable<User> en) : this()  {  this.items = new List<User>(en);  }  public bool Insert(User user)  {  try  {  this.items.Add(user);  return true;  }  catch  {  Console.WriteLine("Insertion Error");  throw;  }  }  public void Remove(User user)  {  try  {  this.items.Remove(user);  }  catch  {  Console.WriteLine("Removing Error");  throw;  }  }  public void Remove(int userId) {  int index = this.items.FindIndex(user => user.id == userId);  try  {  this.items.RemoveAt(index);  }  catch  {  Console.WriteLine("Removing Error");  throw;  }  }  public IEnumerator GetEnumerator()  {  return items.GetEnumerator();  }  Object IEnumerator.Current  {  get { return this.items[currIndex]; }  }  bool IEnumerator.MoveNext()  {  this.currIndex++;  var indexIsOk = (this.currIndex < this.items.Count && this.currIndex >= 0);  if (!indexIsOk)  currIndex--;  return indexIsOk;  }  void IEnumerator.Reset()  {  this.currIndex = 0;  }  public User this[string name]  {  get  {  var nameLowered = name.ToLower();  try  {  return items.Find((user) =>  {  return user.FirstName.ToLower().Contains(nameLowered) ||  user.LastName.ToLower().Contains(nameLowered) ||  user.FullName.ToLower().Contains(nameLowered);  });  }  catch  {  return null;  }  }  }  public User GetById(int id)  {  try  {  return this.items.Find(user => user.id == id);  }  catch  {  return null;  }  }  } |
| public static class UserCollectionExtension  {  //no property extension provided  public static int GetBankClientsCount(this UserCollection userCollection)  {  int count = 0;  foreach(User user in userCollection)  {  if (user is BankClient) count++;  }  return count;  }  public static int GetBankEmployeesCount(this UserCollection userCollection)  {  int count = 0;  foreach(User user in userCollection)  {  if (user is BankEmployee) count++;  }  return count;  }  } |

|  |
| --- |
| Серіалізація Account |
| [Serializable]  public class Account  {  … |
| <?xml version="1.0"?>  <ArrayOfAccount xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">  <Account>  <currency>uah</currency>  <MoneyAmount>100</MoneyAmount>  </Account>  <Account>  <currency>uah</currency>  <MoneyAmount>5454</MoneyAmount>  </Account>  <Account>  <currency>usd</currency>  <MoneyAmount>200</MoneyAmount>  </Account>  </ArrayOfAccount> |
| public static void SerializationDemoXML()  {  Account[] accArray = new Account[BankSystem.Accounts.Count];  BankSystem.Accounts.Values.CopyTo(accArray, 0);    XmlSerializer formatter = new XmlSerializer(typeof(Account[]));    using (FileStream fs = new FileStream("accounts.xml", FileMode.OpenOrCreate))  {  try  {  formatter.Serialize(fs, accArray);  Console.WriteLine("Serialization done");  }  catch (Exception exp)  {  Console.WriteLine("Serialization Error");  Console.WriteLine(exp.Message);  }  }    using (FileStream fs = new FileStream("accounts.xml", FileMode.OpenOrCreate))  {  try  {  var accountsDeserialized = (Account[])formatter.Deserialize(fs);  Console.WriteLine("Deserialization done");  foreach(var acc in accountsDeserialized)  {  acc.ShowAmount();  }  }  catch (Exception exp)  {  Console.WriteLine("Deserialization error");  Console.WriteLine(exp.Message);  }  }  } |

|  |
| --- |
| Створення власного стеку на основі Generic типів |
| class Stack<T>  {  private int MaxSize = Int32.MaxValue;  private List<T> items;  public Stack(List<T> items)  {  this.items = items;  }  public Stack(T[] val) : this(new List<T>(val))  {}  public Stack() : this(new List<T>())  {}  public void Push(T value)  {  if (items.Count >= MaxSize)  throw new Exception("Stack Overflow Exception =))");  items.Add(value);  }  public T pop()  {  if (items.Count == 0)  throw new Exception("Stack is empty");  var removeIndex = items.Count - 1;  T returnValue = items[removeIndex];  items.RemoveAt(removeIndex);  return returnValue;  }  public int Count => items.Count;  public bool isEmpty() {  return items.Count == 0;  }  } |

**Висновки**

Я ознайомився з можливостями мови C# щодо обробки колекцій даних, їх серіалізації, а також з анонімними методами, методами-розширеннями.