

МІНІСТЕРСТВО ОСВІТИ 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 “Програмування 1. Об'єктно-орієнтоване програмування та шаблони проєктування”

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

|  |  |  |
| --- | --- | --- |
| Виконала  студентка IІ курсу  групи КП-02  Гудзіцька Дарина Сергіївна |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2021

**Мета роботи**

Ознайомитися з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Вивчити механізми оброблення подій у C#, а також можливості, які мають методи-розширення.

**Постановка завдання**

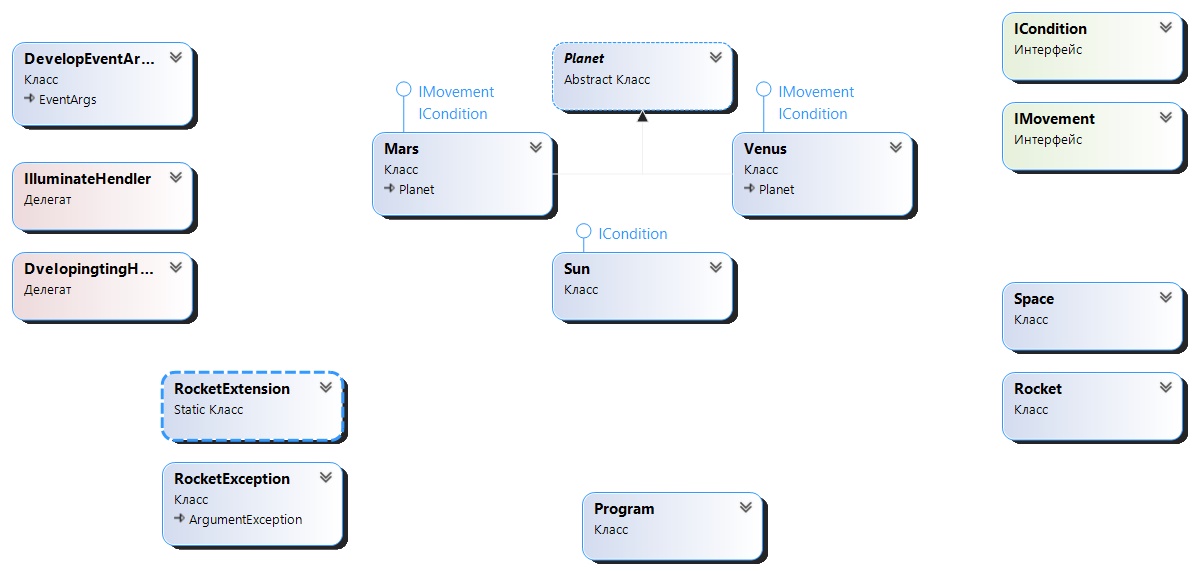
Для ієрархії класів, побудованої в лабораторній роботі №1, реалізувати:

1. Множину інтерфейсів. При чому один з класів повинен реалізовувати щонайменше 2 інтерфейси. Також продемонструвати реалізацію explicit implementation інтерфейса, обгрунтувати її використання **(1 бал)**.
2. Абстрактний клас. Забезпечити його наслідування. Наявність в цьому класі абстрактних методів - обов’язкова **(1 бал)**.
3. Механізм «делегат – подія – обробник події» **(2 бали)**.
4. Перетворити код, який забезпечує роботу з подіями та обробниками подій, на код, що використовує (\*) **(2 бали)**:
   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. Механізм створення та оброблення власних помилок **(2 бали)**:
   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;
   4. реалізувати різні сценарії оброблення помилки.
2. Метод-розширення будь-якого класу **(1 бал)**.

**UML діаграма класів**

****

**Фрагменти коду**

|  |
| --- |
| public interface ICondition  {  void ShowInfo();  }  public interface IMovement  {  void ShowInfo();  }  public class Mars : Planet, IMovement, ICondition |

|  |
| --- |
| public abstract class Planet  {  abstract public void Develop(Rocket r, DevelopEventArgs dEvent);  } |

|  |
| --- |
| public class DevelopEventArgs : EventArgs  {  public int developmentLevel;  public int humanCount;  public DevelopEventArgs(int dL, int hC)  {  this.developmentLevel = dL;  this.humanCount = hC;  }  public DevelopEventArgs() : this(0, 0) { }  }  public delegate void DvelopingtingHandle(Rocket r, DevelopEventArgs dEvent);  public class Rocket  {  public event DvelopingtingHandle RocketEvent;  public string name;  public Rocket(string name)  {  this.name = name;  }  public void StartDeparture()  {  int dL, hC;  DevelopEventArgs dargs;  try  {  Console.Write("Enter level of development: ");  dL = Int32.Parse(Console.ReadLine());  Console.Write("Enter number of astronauts : ");  hC = Int32.Parse(Console.ReadLine());  dargs = new DevelopEventArgs(dL, hC);  }  catch  {  dargs = new DevelopEventArgs();  }  Console.WriteLine($"---Rocket {this.name} takes off, good luck---");  if (RocketEvent != null)  RocketEvent((Rocket)this, dargs);  }  } |

|  |
| --- |
| //анонимный метод  IlluminateHendler handler1 = delegate  {  Console.WriteLine("\tThe sun will shine for another 4 million years") ;  };  handler1(sun);  //lambda-выражение  IlluminateHendler handler2 = (sun) =>  Console.WriteLine("\tThe sun will shine for another 3 million years") ;  handler2(sun);  //Actoin  Action actiondelegate;  actiondelegate = Sun.Illuminate;  //Func  Func<string, string> iluminate = Sun.Flash;  string str = " lasting 15 minutes!!!";  Console.WriteLine(iluminate(str)); |

|  |
| --- |
| private int countHuman;  public int CountHuman  {  get { return countHuman; }  set  {  if (value <= 0)  throw new RocketException("People must participate in the expedition!", value);  else if(value > 100)  throw new RocketException("It`s too many people for 1 expedition!", value);  else  countHuman = value;  }  }  class RocketException : ArgumentException  {  public int Value { get; }  public RocketException()  : base("Incorrect count of human.")  {  }  public RocketException(string message, int val)  : base(message)  {  Value = val;  }  }  Console.WriteLine("\t\t\t--------EXEPTION--------");  try  {  Rocket r = new Rocket("Apol") { CountHuman = -2 };  }  catch (RocketException ex)  {  Console.WriteLine($"Error: {ex.Message}");  Console.WriteLine($"Invalid value: {ex.Value}");  }  try  {  Rocket r = new Rocket("Apol2") { CountHuman = 0 };  }  catch (RocketException ex)  {  Console.WriteLine($"Error: {ex.ToString()}");  } |

|  |
| --- |
| public static class RocketExtension  {  public static string StartExpedition(this Rocket r, string place)  {  return "The rocket goes on an expedition to " + place;  }  } |

**Висновки**

Виконавши дану лабораторну роботу відбулося ознайомлення з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати; вивчено механізми оброблення подій у C#, а також можливості, які мають методи-розширення.