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”

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

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) II курсу  групи КП-01  Грабовська Анастасія Дмитрівна  (*прізвище, ім’я, по батькові*) |  | 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 задачі**

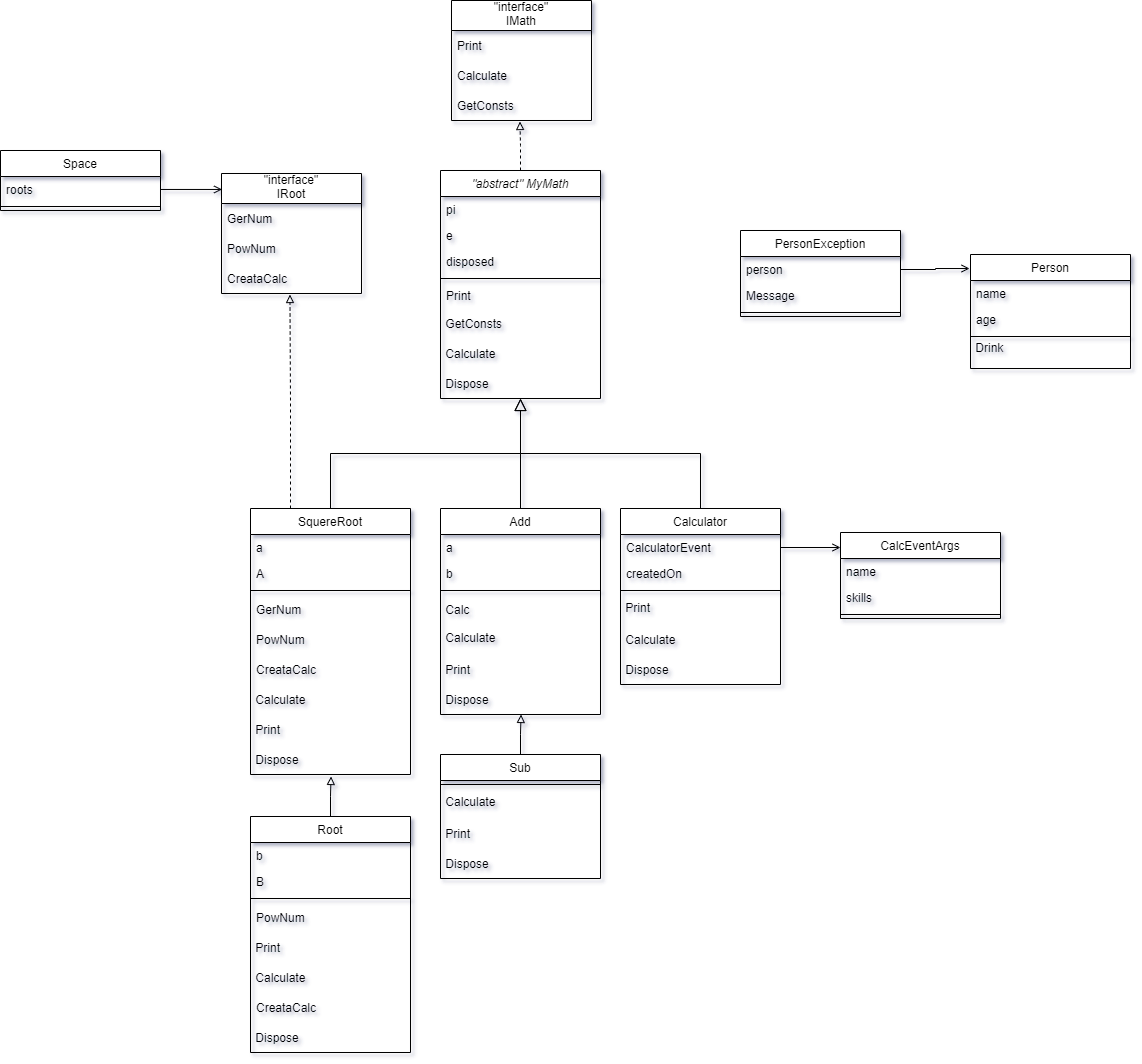
Для ієрархії класів, побудованої в лабораторній роботі №1, реалізувати:

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:**

1.

|  |
| --- |
| interface IMath      {          void Print();          void Calculate();          void GetConsts();      }      interface IRoot      {          double GetNum();          double PowNum();          void CreateCalc(Calculator calculator, CalcEventArgs args);      } |
| abstract class MyMath : IMath, IDisposable      {          protected bool disposed;  …  } |
| public void GetConsts()          {              WriteLine($"pi is {pi}, e is {e}.");          }          void IMath.GetConsts()          {              WriteLine("I know no constants");          } |

2.

|  |
| --- |
| abstract class MyMath : IMath, IDisposable      {          ...          public abstract void Print();          public abstract void Calculate();          public abstract void Dispose();      } |

3.

|  |
| --- |
| delegate void CalcHandle(Calculator calculator, CalcEventArgs args);      class CalcEventArgs : EventArgs      {          public string name;          public int skills;          public CalcEventArgs(string name, int skills)          {              this.name = name;              this.skills = skills;          }      }      interface IRoot      {          double GetNum();          double PowNum();          void CreateCalc(Calculator calculator, CalcEventArgs args);      }  class SquereRoot : MyMath, IRoot      {          ...          public virtual void CreateCalc(Calculator calculator, CalcEventArgs args)          {              if(args.skills >= 10)              {                  WriteLine($"{args.name} can calculate. My skills is {args.skills}");                  calculator.GetConsts();              }              else              {                  WriteLine($"{args.name} can't calculate. My skills is {args.skills}, but must be bigger than 10");                  calculator.Print();              }          }          ...      }  class Root : SquereRoot      {  ...          public override void CreateCalc(Calculator calculator, CalcEventArgs args)          {              if(args.skills >= 15)              {                  WriteLine($"{args.name} can calculate. My skills is {args.skills}");                  calculator.GetConsts();              }              else              {                  WriteLine($"{args.name} can't calculate. My skills is {args.skills}, but must be bigger than 15");                  calculator.Print();              }          }  ...      }  class Calculator : MyMath      {          public event CalcHandle CalculatorEvent;          private static string createdOn;          public Calculator()          {              createdOn = DateTime.Now.ToLongTimeString();          }          public override void Print()          {              WriteLine($"I'm created at {createdOn}.");          }          public override void Calculate()          {              CalcEventArgs args = new CalcEventArgs("Bimo", 10);              if(CalculatorEvent != null) CalculatorEvent(this, args);          }  ...      }  class Space      {          IRoot[] roots;          public Space(Calculator calculator)          {              roots = new IRoot[2];              roots[0] = new SquereRoot();              roots[1] = new Root();              foreach(IRoot r in roots)              {                  calculator.CalculatorEvent += new CalcHandle(r.CreateCalc);              }          }      } |
| Calculator calculator = new Calculator();              Space space = new Space(calculator);              calculator.Calculate(); |

4.

|  |
| --- |
| Calculator calculator = new Calculator();  CalcEventArgs calcArgs = new CalcEventArgs("Pipi", 17);              CalcHandle calcHandle = delegate(Calculator calculator, CalcEventArgs calcAgs)              {                  if(calcArgs.skills >= 15)                  {                      WriteLine($"{calcArgs.name} can calculate. My skills is {calcArgs.skills}");                      calculator.GetConsts();                  }                  else                  {                      WriteLine($"{calcArgs.name} can't calculate. My skills is {calcArgs.skills}, but must be bigger than 15");                      calculator.Print();                  }              };              calcHandle(calculator, calcArgs);              Action<string> line = name => Console.WriteLine("{0} invoked an action", name);              Func<int, int, string> result = (a, b) => $"{a} + {b} is {a+b}";              line("Anastasia");              WriteLine(result(3,8)); |

5.

|  |
| --- |
| class Person      {          public string name;          public int age;          public Person(string name, int age)          {              this.name = name;              this.age = age;          }          public void Drink()          {              if(age < 18) throw new PersonException(this);              WriteLine($"{this.name} drink vodka.");          }      }      class PersonException : Exception      {          public Person args;          public PersonException(Person args) : base()          {              this.args = args;          }          public override string Message => $"Person {args.name} is too young. Come back in {18 - args.age}";      } |
| Person p1 = new Person("John", 16);              try              {                  p1.Drink();              }              catch(PersonException e)              {                  WriteLine(e.Message);              }              finally              {                  Person p2 = new Person("Mike", 19);                  p2.Drink();              }                  // Person John is too young. Come back in 2                  // Mike drink vodka. |

6.

|  |
| --- |
| static class MyExtentions      {          public static void Announce(this Sub sub)          {              WriteLine("This is extention method");          }          public static int SubTwice(this Sub sub, int c)          {              return sub.Calc() - c;          }          public static int SubAdd(this Sub sub, int c)          {              return sub.Calc() + c;          }      } |

**Висновки:**

Під час виконання лабораторної роботи я ознайомилася з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Також я вивчила механізми оброблення подій у C# та мож можливості, які мають методи-розширення.