

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  Студентк IІ курсу  групи КП-92  Мовчан Максим Олександрович  (*прізвище, ім’я, по батькові*) |  | Перевірив  “8” “листопада” 2020 р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

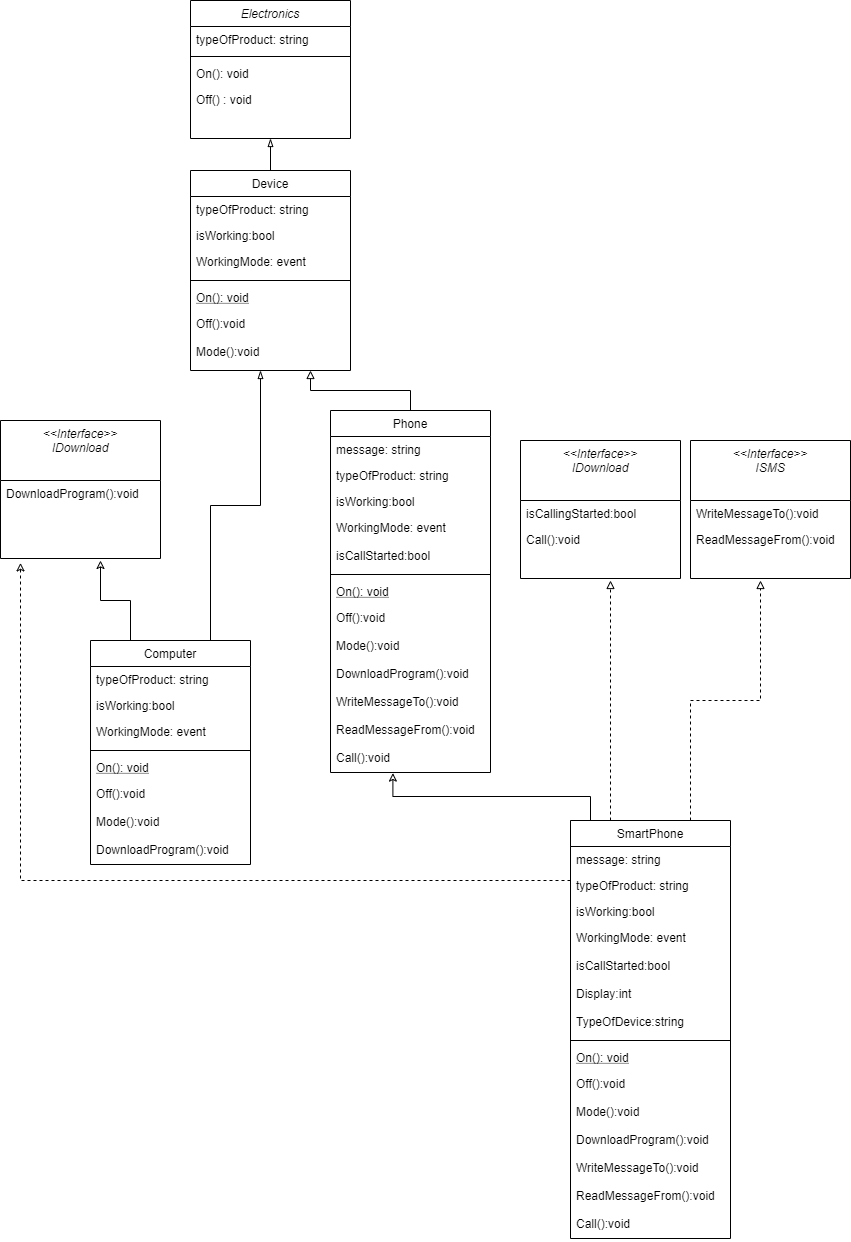
Ознайомитися з такими можливостями мови програмування C# як абстрактні класи, інтерфейси, делегати. Вивчити механізми оброблення подій у C#.

**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. Наявність в цьому класі абстрактних методів - обов’язкова.
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**



**Тексти коду програми**

|  |
| --- |
| Множина інтерфейсів. |
| interface IDownload  {  void DownloadProgram(string name);  }  interface IRinging  {  bool isCallStarted { get; set; }  void Call(string number);    }  interface ISMS  {  void WriteMessageTo(string abonent, string message);  void ReadMessageFrom(string abonent);  } |
| Один з класів реалізує 2 інтерфейси. Також продемонстровано реалізацію explicit implementation інтерфейса. |
| class Phone : Device, ISMS, IRinging, IDownload  {  protected string message;  public string Message  {  set  {  if (value.Length > 0 && value.Length <= 30) message = value;  else throw new OverLengthError(new ExceptionArgs(value, 30));  }  get  {  return message;  }  }  public void DownloadProgram(string name)  {  Console.WriteLine("Program " + name + " is downloaded successfuly");  }  void ISMS.notification()  {  Console.WriteLine($"Notification from SMS");  }  void IRinging.notification()  {  Console.WriteLine($"Notification from Ringing");  }  {  Console.WriteLine($"You have written {abonent}: {message}");  }  public void ReadMessageFrom(string abonent) {  Console.WriteLine($"You have read message from {abonent}");  }  public bool isCallStarted { get; set; }  public void Call(string number) {  Console.WriteLine($"You are calling to {number}");  } |
| Абстрактний клас.  Наявність в цьому класі абстрактних методів - обов’язкова |
| abstract class Electronics  { public string typeOfProduct = "Electronics";  public abstract void On();  public abstract void Off();} |
| Наслідування абстрактного класу. |
| class Device : Electronics  {  public event ModeDelegate workingMode;  public bool isWorking = true;  public Action WorkingMode { get; internal set; }  public override void On()  {  Console.WriteLine("Device is switched on");  }  public override void Off()  {  Console.WriteLine("Device is switched off");  }  public void Mode()  {  if(isWorking)  {  // Console.WriteLine("Working mode: on");  if(workingMode != null) workingMode();  }    }  } |
| Механізм «делегат – подія – обробник події»**.** |
| class Device : Electronics  {  public event ModeDelegate workingMode;  public string modeDevice="ALARM RINGING!!!";  public bool isWorking = true;  public Action WorkingMode { get; internal set; }  public override void On()  {  Console.WriteLine("Device is switched on");  }  public override void Off()  {  Console.WriteLine("Device is switched off");  }  public void Mode()  {  if (isWorking)  {  // Console.WriteLine("Working mode: on");  if (workingMode != null)  {  workingMode(this);  }  }  }  public delegate void ModeDelegate(Device d);  }  class Phone : Device, ISMS, IRinging, IDownload  {  protected string message;  public string Message  {  set  {  if (value.Length > 0 && value.Length <= 30) message = value;  else throw new OverLengthError(new ExceptionArgs(value, 30));  }  get  {  return message;  }  }  public void reaction(Device d)  {  Console.WriteLine("It is working mood: ", d.modeDevice);  }  public void DownloadProgram(string name)  {  Console.WriteLine("Program " + name + " is downloaded successfuly");  }  void ISMS.notification()  {  Console.WriteLine($"Notification from SMS");  }  void IRinging.notification()  {  Console.WriteLine($"Notification from Ringing");  }  public void WriteMessageTo(string abonent, string message)  {  Console.WriteLine($"You have written {abonent}: {message}");  }  public void ReadMessageFrom(string abonent) {  Console.WriteLine($"You have read message from {abonent}");  }  public bool isCallStarted { get; set; }  public void Call(string number) {  Console.WriteLine($"You are calling to {number}");  }  }  static void Main(string[] args)  {  Device dev = new Device();  Phone phone = new Phone();  dev.workingMode += phone.reaction;  dev.Mode();    try  {  phone.Message = "SOOOOOMETHING with really long text for sms texting. For example, this text that more then 30 symbols sooooo";  }  catch (OverLengthError e)  {  Console.WriteLine(e.err);  }  Console.ReadKey();  } |
| Механізм створення та обробки власних помилок. |
| class ExceptionArgs  {  public readonly string message;  public readonly int maxLength;  public ExceptionArgs(string str, int limit)  {  maxLength = limit;  message = str;  }  }  class OverLengthError : ApplicationException  {  public readonly string err;  public OverLengthError(ExceptionArgs ex)  {  err = "There are extra symbols in the message. Make it shorter!";  }  }  class Phone : Device, ISMS, IRinging, IDownload  {  protected string message;  public string Message  {  set  {  if (value.Length > 0 && value.Length <= 30) message = value;  else throw new OverLengthError(new ExceptionArgs(value, 30));  }  get  {  return message;  }  }  public void DownloadProgram(string name)  {  Console.WriteLine("Program " + name + " is downloaded successfuly");  }  public void WriteMessageTo(string abonent, string message)  {  Console.WriteLine($"You have written {abonent}: {message}");  }  public void ReadMessageFrom(string abonent) {  Console.WriteLine($"You have read message from {abonent}");  }  public bool isCallStarted { get; set; }  public void Call(string number) {  Console.WriteLine($"You are calling to {number}");  }  } |
| Приклади використання |
| Phone phone = new Phone();  try  {phone.Message = "SOOOOOMETHING with really long text for sms texting. For example, this text that more then 30 symbols sooooo";}  catch (OverLengthError e)  {  Console.WriteLine(e.err);  } |
|  |

**Висновки**

У даній лабораторній роботі було створено інтерфейси, які реалізовані у класах також наявна реалізація одразу декількох інтерфейсів у одному класі.

Створено абстрактний клас з подальшим його наслідуванням та реалізацією методів. Реалізовано механізм «делегат – подія – обробник події».

Також створено новий клас виключної ситуації, новий клас аргументів для передачі їх до обробника виключної ситуації, забезпечино ініціювання створеної виключної ситуації, та реалізовано обробник даної помилки.