

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 №**1

з дисципліни “ООП ”

тема: “C# .Net. Реалізація основних принципів ООП мовою C#”

|  |  |  |
| --- | --- | --- |
| Виконав  студент II курсу  групи КП-92  Мовчан Максим Олександрович  (*прізвище, ім’я, по батькові*) |  | Перевірив  17 жовтня 2020 р.  Викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи:**

Ознайомитися з основами об’єктно-орієнтованого підходу до створення ПЗ у мові С#, створенням класів, об’єктів, механізмами інкапсуляції, наслідування та поліморфізму.

**Постановка задачі**

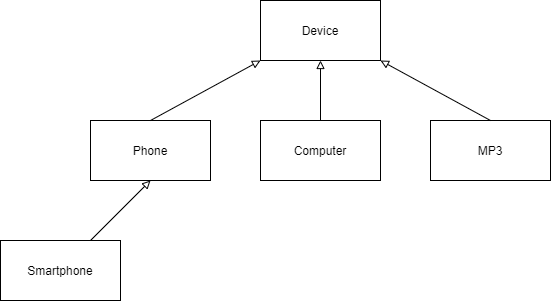
Побудувати ієрархію класів, що відтворюватимуть відношення наслідування між об’єктами реального світу. (Кількість класів >= 5). При цьому забезпечити реалізацію таких механізмів ООП:

1. Наявність у класах статичних даних. Забезпечити коректну ініціалізацію цих даних.
2. Наявність у класах змінних та методів з різними модифікаторами доступу.
3. Наявність у класах полів const та readonly.
4. Наявність у класах властивостей (складніше, ніж просто get;set;).
5. Конструктори. Створити для розроблюваних класів такі конструктори:
   * + перевизначений конструктор за замовчанням;
     + перевизначений конструктор;
     + приватний конструктор;
     + статичний конструктор.

Продемонструвати, яким чином викликаються конструктори базового та дочірнього класів.

1. Використати віртуальні та перевизначені методи.
2. Продемонструвати поліморфізм методів.

**Діаграма залежностей**



**Постановка та виконання завдань**

1.Наявність у класах статичних даних. Забезпечити коректну ініціалізацію цих даних.

|  |  |  |
| --- | --- | --- |
| У класі Device | У класі Phone | У класі Smartphone |
| public static string typeOfElectronics = "This is device"; | new public static string typeOfElectronics = "\nThis is a Phone"; | new public static string typeOfElectronics = "This is a smartphone"; |

2. Наявність у класах змінних та методів з різними модифікаторами доступу.

|  |  |
| --- | --- |
| Приклад змінних у класі Device | Приклад методів класу Computer |
| private int yearOfProducting;  private float price;  protected string ipAdress;  internal string manufacturer;  const bool isDevice = true;  readonly string fieldOfComercial = "Electronics";  protected bool isWork;  public static string typeOfElectronics = "This is device"; | public override int deviceAge()  {  return 2020 - Year;  }  public override void typeOfDevice()  {  Console.WriteLine(typeOfElectronics);  } |

3. Наявність у класах полів const та readonly.

|  |
| --- |
| Приклад змінних у класі Device |
| const bool isDevice = true;  readonly string fieldOfComercial = "Electronics"; |

4. Наявність у класах властивостей .

|  |
| --- |
| ­Приклад властивостей у класі Device |
| public string IP  { get  {  return this.ipAdress;  }  set  {  this.ipAdress = value;  }  }  public float Price  {  get  {  return this.price;  }  set  {  if (value > 0)  {  this.price = value;  }  }  }  public int Year  {  get { return this.yearOfProducting; }  set  {  if (value > 1900)  {  this.yearOfProducting = value;  }  }  } |

5.Конструктори. Створити для розроблюваних класів такі конструктори:

|  |
| --- |
| Перевизначений конструктор за замовчанням; |
| public Device() : this(0, 0, true) { } |

|  |
| --- |
| перевизначений конструктор |
| public Device() : this(0, 0, true) { }  public Device(int year, float cost, bool isWorking)  {  yearOfProducting = year;  price = cost;  isWork = isWorking;  } |

|  |
| --- |
| приватний конструктор |
| private Smartphone()  {  Console.WriteLine("\nConstractor of the Smartphone");  }  private Smartphone SmartphoneInstance { get; set; }  public static Smartphone GetSmartphoneInstance()  {  if (SmartphoneInstance == null)  {  SmartphoneInstance = new Smartphone();  }  return SmartphoneInstance;  }} |

|  |
| --- |
| статичний конструктор |
| static MP3()  {  Console.WriteLine("\n Static constractor for MP3");  } |

|  |  |
| --- | --- |
| конструктори базового та дочірнього класів | Виклик |
| public Device(int year, float cost, bool isWorking)  {  yearOfProducting = year;  price = cost;  isWork = isWorking;  }  public Phone()  {  Console.WriteLine("\nConstractor of the phone");  } | Device dev = new Device(2019, 1200.32f, true);  Phone phone = new Phone(); |

1. Використати віртуальні та перевизначені методи.

|  |  |
| --- | --- |
| У класі Device | У класі Phone |
| public virtual int deviceAge() { return 0; }  public virtual void typeOfDevice()  {  Console.WriteLine(typeOfElectronics);  } | public override int deviceAge()  {  return 2020 - Year;  }  public override void typeOfDevice()  {  Console.WriteLine(typeOfElectronics);  } |

1. Продемонструвати поліморфізм методів.

|  |  |
| --- | --- |
| Клас Device | Клас Phone |
| public virtual int deviceAge() { return 0; } | public override int deviceAge()  {  return 2020 - Year;  } |

**Висновки**

У даній роботі, було проведене ознайомлення з основами об’єктно-орієнтованого підходу до створення ПЗ у мові С#. Через створення класів, на основі базового класу Device, було створено дочірні класи, такі як: Phone, Smartphone, MP3, Computer.

Для створення дочірніх класів був використаний основний з підходів ООП: наслідування. Було виділено предметну галузь, що є одним з механізмів інкапсуляції та також було проведено перевизначення методів, таким чином вони зберігають свій інтерфейс, але змінюють реалізацію, це є складовою поліморфізму.