

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 №** 1

з дисципліни “Основи програмування”

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

|  |  |  |
| --- | --- | --- |
| Виконав  студент(ка) II курсу  групи КП-01  Северин Юрій Юрійович  (*прізвище, ім’я, по батькові*) | Київ 2021 | Перевірила  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

**Мета роботи**

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

**Постановка завдання**

Побудувати ієрархію класів, що відтворюватимуть відношення наслідування між об’єктами реального світу (кількість класів >= 5). При цьому:

1. Забезпечити наявність у класах полів та методів з різними модифікаторами доступу, пояснити свій вибір **(1 бал)**.
2. Забезпечити наявність у класах властивостей: складніше, ніж просто get;set;, обгрунтувати доцільність створення властивості **(1 бал)**.
3. Створити для розроблюваних класів такі конструктори **(2 бали)**:
   * + конструктор за замовчанням;
     + конструктор з параметрами;
     + приватний конструктор;
     + статичний конструктор.

Продемонструвати, яким чином викликаються конструктори базового та дочірнього класів.

1. Використати віртуальні та перевизначені методи **(1 бал)**.
2. Додати до класів методи, наявність яких дозволятиме управляти знищенням екземплярів цих класів **(2 бали)**:
   1. реалізувати інтерфейс IDisposable;
   2. створити деструктори;
   3. забезпечити уникнення конфліктів між Dispose та деструктором.
3. Забезпечити виклики методів GC таким чином, щоб можна було простежити життєвий цикл об’єктів, що обробляються (зокрема, використати методи Collect, SupressFinalize, ReRegisterForFinalize, GetTotalMemory, GetGeneration, WaitForPendingFinalizers). Створити ситуацію, яка спровокує примусове збирання сміття GC **(2 бали)**.

**Текст коду програми**

|  |
| --- |
| **Program.cs** |
| using System;  using System.Collections.Generic;  using System.Threading;  namespace lab1  {  class Program  {  static void Main(string[] args)  {  Person p1 = new Person("Ada Johns");  SuperHero sh1 = new SuperHero("Ivan Ivanov", new List<string> { "1 superpower", "2 superpower" });  Console.WriteLine(p1.Introduce());  Console.WriteLine(sh1.Introduce());  Console.WriteLine(Person.HowMany() + "\n");  Console.WriteLine("Memory before collect: " + GC.GetTotalMemory(false));  Console.WriteLine("Generation: " + GC.GetGeneration(p1));  GC.Collect(2, GCCollectionMode.Forced);  GC.WaitForPendingFinalizers();  Console.WriteLine("Memory after collect: " + GC.GetTotalMemory(false));  Console.WriteLine("Generation: " + GC.GetGeneration(p1));  Console.WriteLine();  sh1.Dispose();  Console.ReadKey();  }  }  } |

|  |
| --- |
| **Person.cs** |
| using System;  using System.Collections.Generic;  using System.Text;  namespace lab1  {  class Person : IDisposable  {  #region Properties and Fields  internal static long counter;  DateTime bornDate;  private bool disposedValue;  string name;  public virtual string Name  {  get  {  return name;  }  set  {  name = value;  }  }  DateTime BornDate  {  get  {  return bornDate;  }  set  {  if (value > DateTime.Now)  {  throw new ArgumentException("Can`t be born in future");  }  bornDate = value;  }  }  #endregion  #region Constructor  static Person()  {  counter = 0;  }  public Person()  {  counter++;  }  public Person(string name)  {  Name = name;  BornDate = DateTime.Now;  counter++;  }  internal protected Person(string name, DateTime date)  {  Name = name;  BornDate = date;  counter++;  }  #endregion  public string Introduce()  {  return $"Hi I'm {Name}, and I was born on {BornDate.Date}. I'm {this.GetType().Name}";  }  public static string HowMany()  {  return $"There are {counter} of us";  }  #region GC  protected virtual void Dispose(bool disposing)  {  GC.SuppressFinalize(true);  if (!disposedValue)  {  if (disposing)  {  Console.WriteLine("Disposing managed resourses");  }  Console.WriteLine("Disposing unmanaged resourses");  name = null;  disposedValue = true;  }  GC.ReRegisterForFinalize(this);  }  ~Person()  {  Dispose(disposing: false);  Console.WriteLine("Person destructed");  }  public void Dispose()  {  Dispose(disposing: true);  GC.SuppressFinalize(this);  }  #endregion  }  } |

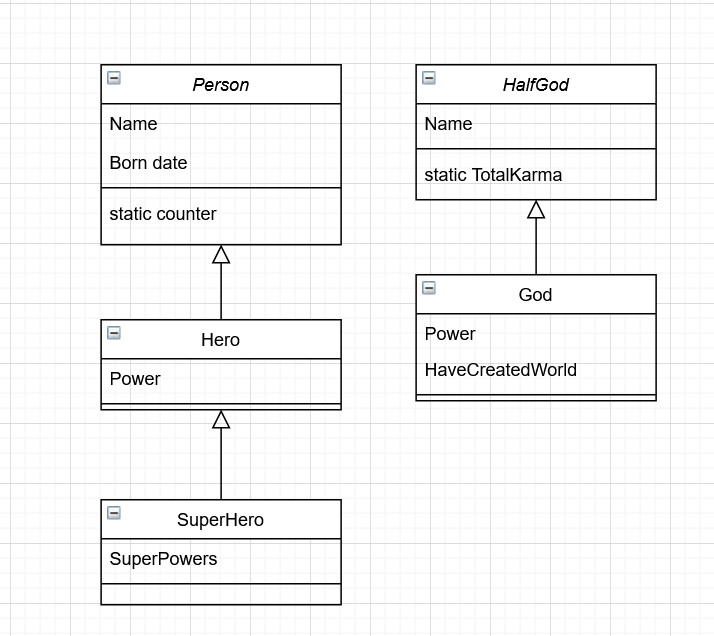
|  |
| --- |
| **Hero.cs** |
| using System;  using System.Collections.Generic;  using System.Text;  namespace lab1  {  class Hero : Person, IDisposable  {  public string Power { get; private set; }  #region Constructors  public Hero()  {  counter++;  }  public Hero(string name, string power) : base(name)  {  Power = power;  }  Hero(string name, DateTime bornDate, string power) : base(name, bornDate)  {  }  #endregion  public virtual string SuperPower()  {  return $"My SuperPower is: {Power}";  }  ~Hero()  {  Dispose(disposing: false);  Console.WriteLine("Hero destructed");  }  }  } |

|  |
| --- |
| **SuperHero.cs** |
| using System;  using System.Collections.Generic;  using System.Text;  namespace lab1  {  class SuperHero : Hero  {  public List<string> SuperPowers { get; private set; }  #region Constructor  public SuperHero()  {  counter++;  }  public SuperHero(string name, List<string> superPowers) : base(name, default)  {  SuperPowers = superPowers;  }  #endregion  public override string SuperPower()  {  string answ = "My superpowers are:";  foreach(var a in SuperPowers)  {  answ += $"- {a}\n";  }  return answ;  }  ~SuperHero()  {  Dispose(disposing: false);  Console.WriteLine("SuperHero destructed");  }  }  } |

|  |
| --- |
| **HalfGod.cs** |
| using System;  using System.Collections.Generic;  using System.Text;  namespace lab1  {  class HalfGod : IDisposable  {  #region Properties and Fields  string name;  private bool disposedValue;  public string Name  {  get  {  return $"My name is {name}";  }  set  {  name = value;  }  }  public static int TotalKarma { get; set; }  #endregion  #region Constructors  public HalfGod(string name)  {  Name = name;  TotalKarma = 0;  }  static HalfGod()  {  TotalKarma = 0;  }  public HalfGod()  {  }  #endregion  public void IncreaseCarma()  {  TotalKarma++;  }  public void DecreaseCarma()  {  TotalKarma--;  }  protected virtual void Dispose(bool disposing)  {  if (!disposedValue)  {  if (disposing)  {  Console.WriteLine("Disposing managed resourses");  }  Console.WriteLine("Disposing unmanaged resourses");  name = null;  disposedValue = true;  }  }  ~HalfGod()  {  Dispose(disposing: false);  Console.WriteLine("HalfGod destructed");  }  public void Dispose()  {  // Do not change this code. Put cleanup code in 'Dispose(bool disposing)' method  Dispose(disposing: true);  GC.SuppressFinalize(this);  }  }  } |

|  |
| --- |
| **God.cs** |
| using System;  namespace lab1  {  sealed class God : HalfGod  {  bool HaveCreatedWorld;  public God()  {  }  public string IsOurGod()  {  if (HaveCreatedWorld)  {  return "I'm God, I've created world";  }  return "I haven't created world(";  }  public God(string name, bool haveCreatedWorld) : base(name)  {  HaveCreatedWorld = haveCreatedWorld;  }  ~God()  {  Dispose(disposing: false);  Console.WriteLine("God destructed");  }  }  } |

**UML діаграма**

****

**Висновки**

Ознайомилися з основами об’єктно-орієнтованого підходу до створення ПЗ у мові С#, створенням класів, об’єктів, механізмами інкапсуляції, наслідування та поліморфізму. Вивчили механізм управління ресурсами, реалізований у .Net.