**МІНІСТЕРСТВО**

**ОСВІТИ ТА НАУКИ УКРАЇНИ**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Кафедра інформаційних систем та технологій**

**Звіт**

**з лабораторної  роботи № 2**

**«Поліморфізм. Наслідування. Перегрузка методів.»**

**з дисципліни**

**«Програмування – 2. ООП»**

**Варіант №21**

|  |  |  |
| --- | --- | --- |
| **Перевірив:**  **доц. Корнага Ярослав Ігорович** |  | **Виконала:** |
|  |  | **Студентка гр. ІС-12 ФІОТ**  **Мельникова Катерина** |
|  |  | **1 курс,**  **залікова книжка № IC-1222** |

**Київ 2021**

**Тема:** Поліморфізм. Наслідування. Перегрузка методів.

**Завдання 1:**

Створити додаток, який задовольняє вимогам, наведеним в завданні. Наслідування застосовувати тільки в тих завданнях, в яких воно логічно обґрунтоване. Аргументувати належність класу кожного створюваного методу і коректно перевизначити для кожного класу методи Equals, GetHashCode, ToString. При виклику будь-которого методу класу, виводити на екран текстове повідомлення

21. Створити об'єкт класу Квочка, використовуючи класи Птах, Зозуля, Курка. Методи: літати, співати, нести яйця, висиджувати пташенят.

using System;

public class Bird

{

public Bird(string name)

{

Name = name;

}

public string Name { get; set; }

public void Fly()

{

Console.WriteLine("Цей птах вмiє лiтати");

}

public virtual void Sing()

{

Console.WriteLine("Цей птах вмiє спiвати");

}

public void Eggs()

{

Console.WriteLine("Цей птах несе яйця");

}

public virtual void Child()

{

Console.WriteLine("Цей птах висиджує пташенят");

}

public override int GetHashCode()

{

Console.WriteLine("Hash Code: ");

return (Name + this.GetType()).GetHashCode();

}

public override string ToString()

{

Console.WriteLine("Iм'я та тип об'єкту:");

return $"{Name}, {this.GetType()}";

}

public override bool Equals(object obj)

{

Console.WriteLine("\nЧи однаковi об'єкти?");

if (obj is Bird bird) return Name == bird.Name && obj.GetType() == this.GetType();

return false;

}

}

public class Cuckoo : Bird

{

public Cuckoo(string name) : base(name)

{

}

public override void Child()

{

Console.WriteLine("Цей птах не висиджує пташенят");

}

}

public class Chicken : Cuckoo

{

public Chicken(string name) : base(name)

{

}

public override void Sing()

{

Console.WriteLine("Цей птах не вмiє спiвати");

}

}

public class Sitter : Chicken

{

public Sitter(string name) : base(name)

{

}

public override void Child()

{

Console.WriteLine("Цей птах висиджує пташенят");

}

}

**Завдання 2:**

Створити програму, яка задовольняє наступним вимогам:

• Використовувати можливості ООП: класи, наслідування, поліморфізм, інкапсуляція.

• Кожен клас повинен мати змістовну назву та інформативний склад.

• Наслідування має застосовуватися тільки тоді, коли це має сенс.

• Класи повинні бути грамотно розкладені по пакетах.

• Консольне меню повинно бути мінімальним.

• Для зберігання параметрів ініціалізації можна використовувати файли.

21. Продаж смартфонів. Визначити ієрархію телефонів. Відсортувати телефони по моделі, виробнику, розміру дисплею. Знайти телефон, який відповідає вказаним параметрам. Підрахувати загальну кількість девайсів на складі. Реалізувати пошук телефону по діапазону цін.

using System;

using System.Collections.Generic;

public class Smartphone

{

public string Model { get; set; }

public string Produser { get; set; }

public double Size { get; set; }

public int Price { get; set; }

public Smartphone(string model, string produser, double size, int price)

{

Model = model;

Produser = produser;

Size = size;

Price = price;

}

}

public class Shop

{

public List<Smartphone> Items;

public Shop(List<Smartphone> items)

{

Items = items;

}

public List<Smartphone> Sort\_by\_model(string m\_to\_f)

{

List<Smartphone> answer = new List<Smartphone> { };

for (int i = 0; i < Items.Count; i++)

{

if (Items[i].Model == m\_to\_f)

{

answer.Add(Items[i]);

}

}

return answer;

}

public List<Smartphone> Sort\_by\_produser(string p\_to\_f)

{

List<Smartphone> answer = new List<Smartphone> { };

for (int i = 0; i < Items.Count; i++)

{

if (Items[i].Produser == p\_to\_f)

{

answer.Add(Items[i]);

}

}

return answer;

}

public List<Smartphone> Sort\_by\_size(double s\_to\_f)

{

List<Smartphone> answer = new List<Smartphone> { };

for (int i = 0; i < Items.Count; i++)

{

if (Items[i].Size == s\_to\_f)

{

answer.Add(Items[i]);

}

}

return answer;

}

public List<Smartphone> Sort\_by\_price(int minp\_to\_f = 0, int maxp\_to\_f = 100000000)

{

List<Smartphone> answer = new List<Smartphone> { };

for (int i = 0; i < Items.Count; i++)

{

if (Items[i].Price >= minp\_to\_f && Items[i].Price <= maxp\_to\_f)

{

answer.Add(Items[i]);

}

}

return answer;

}

public List<Smartphone> Find\_by\_parameters(string m\_to\_f = null, string p\_to\_f = null , double s\_to\_f = 0)

{

List<Smartphone> answer = new List<Smartphone> { };

for (int i = 0; i < Items.Count; i++)

{

if ((Items[i].Model == m\_to\_f || m\_to\_f == null) && (Items[i].Produser == p\_to\_f || p\_to\_f == null) && (Items[i].Size == s\_to\_f || s\_to\_f == 0))

{

answer.Add(Items[i]);

}

}

return answer;

}

public int Number\_of\_phones()

{

return Items.Count;

}

public void Print\_phones(List<Smartphone> list, string title = "")

{

Console.WriteLine(title);

for (int i = 0; i < list.Count; i++)

{

Console.WriteLine("Модель: " + list[i].Model + " Виробник: " + list[i].Produser + " Розмiр дисплею: " + list[i].Size + " Цiна: " + list[i].Price);

}

}

}

Program.cs

using System;

using System.Collections.Generic;

namespace Lab\_2

{

class Program

{

static void Main(string[] args)

{

void Task1()

{

Console.WriteLine("Завдання 1: ");

Bird S = new Sitter("Ряба");

S.Fly();

S.Sing();

S.Eggs();

S.Child();

Console.WriteLine(S.GetHashCode());

Console.WriteLine(S.ToString());

Bird s = new Sitter("Ряба");

Console.WriteLine(S.Equals(s));

Console.WriteLine(s.GetHashCode());

Console.WriteLine(s.ToString());

Bird s2 = new Sitter("Ряба2");

Console.WriteLine(S.Equals(s2));

Console.WriteLine(s2.GetHashCode());

Console.WriteLine(s2.ToString());

Bird k = new Cuckoo("Ряба");

Console.WriteLine(S.Equals(k));

Console.WriteLine(k.GetHashCode());

Console.WriteLine(k.ToString());

int n = 5;

Console.WriteLine(S.Equals(n));

Console.WriteLine(n.GetHashCode());

Console.WriteLine(n.ToString());

}

void Task2()

{

List<Smartphone> Phones\_list = new List<Smartphone>

{

new Smartphone("Gallaxy A51", "Samsung", 6.5, 10000),

new Smartphone("iPhone 13", "Apple", 6.1, 29999),

new Smartphone("Redmi Note 11", "Xiaomi", 6.43, 8499),

new Smartphone("iPhone 11", "Apple", 6.1, 19999),

new Smartphone("Gellaxy S22", "Samsung", 6.1, 13999),

new Smartphone("Redmi 9A", "Xiaomi", 6.53, 4149)

};

Shop S = new Shop (Phones\_list);

Console.WriteLine("Кiлькiсть телефонiв на складi: " + S.Number\_of\_phones());

S.Print\_phones(S.Sort\_by\_model("iPhone 13"), "Телефони, вiдсортованi по моделi: ");

S.Print\_phones(S.Sort\_by\_produser("Samsung"), "Телефони, вiдсортованi по виробнику: ");

S.Print\_phones(S.Sort\_by\_size(6.1), "Телефони, вiдсортованi по розмiру дiагоналi: ");

S.Print\_phones(S.Sort\_by\_price(5000, 10000), "Телефони у вказаному цiновому дiапазонi: ");

S.Print\_phones(S.Find\_by\_parameters("Gallaxy A51", "Samsung", 6.5), "Телефон, що вiдповiдає вказаним параметрам: ");

S.Print\_phones(S.Find\_by\_parameters(p\_to\_f: "Apple", s\_to\_f : 6.1), "Телефон, що вiдповiдає вказаним параметрам2: ");

}

Console.WriteLine("Оберiть завдання 1 або 2:");

var answer = Console.ReadLine();

if (answer == "1") Task1();

if (answer == "2") Task2();

}

}

}

Висновок:

Під час виконання цієї лабораторної роботи я навчилась працювати з такими парадигмами ООП як наслідування та поліморфізм.