**МІНІСТЕРСТВО**

**ОСВІТИ ТА НАУКИ УКРАЇНИ**

**Національний технічний університет України**

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

**Кафедра інформаційних систем та технологій**

**Звіт**

**з лабораторної роботи No 2**

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

**з дисципліни**

**«Алгоритми і структури данних»**

**Варіант No29**

**Перевірив:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Виконав:**

**Студент гр. ІС-12 ФІОТ**

**Хіврич Володимир**

**Київ 2022**

**Лабораторна робота №2**

**Теорія.**

[**https://www.bestprog.net/ru/sitemap\_ru/c-2/**](https://www.bestprog.net/ru/sitemap_ru/c-2/)

**Додатково через ВПН:**

Перегрузка

<https://metanit.com/sharp/tutorial/3.5.php>

<https://metanit.com/sharp/tutorial/3.36.php>

<https://metanit.com/sharp/tutorial/3.37.php>

Ключове слово this

<https://metanit.com/sharp/tutorial/4.10.php>

Наслідування

<https://metanit.com/sharp/tutorial/3.7.php>

Ключове слово base

<https://metanit.com/sharp/tutorial/3.41.php>

<https://metanit.com/sharp/tutorial/3.42.php>

Абстрактні класи

<https://metanit.com/sharp/tutorial/3.8.php>

Віртуальні методи

<https://metanit.com/sharp/tutorial/3.19.php>

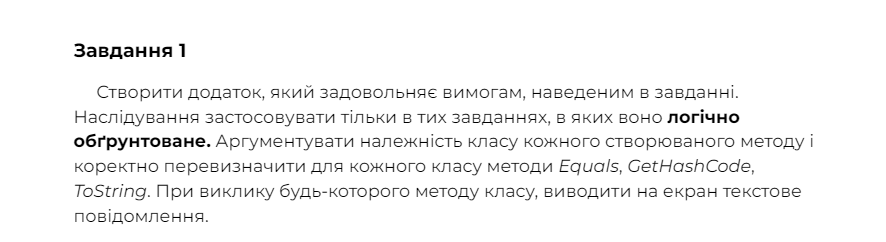
Класи **Object** — вихідний базовий клас. Всі інші класи неявно наслідуються від нього. Одні з віртуальних методів цього класу, методи  *ToString*, *Equals* и *GetHashCode* — віртуальні методи.

<https://metanit.com/sharp/tutorial/3.10.php>

<https://docs.microsoft.com/ru-ru/dotnet/api/system.object?view=netframework-4.8>

При наслідування ваших класів, вам можуть знадобитися власні методи *ToString*, *Equals* и *GetHashCode*. В цій статті розказано як перегрузити ці методи у своїх класах і для чого це робити.

<http://bit.ly/2SIoMpC>

****

**Лістинг програми**

Клас **Window**

using System;

using System.Collections.Generic;

using System.Text;

namespace lab2

{

    class Window

    {

*public* int width { get; }

*public* int height { get; }

*public* Window(int width, int height)

        {

            this.width = width;

            this.height = height;

        }

*public* *override* bool Equals(object obj)

        {

*if* (obj == null) *return* false;

            Window w = obj as Window;

*if* (w as Window == null) *return* false;

*return* this.width == w.width && this.height == w.height;

        }

*public* *override* int GetHashCode() { *return* 31 \* width \* height; }

*public* *override* string ToString()

        {

*return* "{ Width - " + width + "; Height - " + height + " }";

        }

    }

}

**Клас Door**

using System;

using System.Collections.Generic;

using System.Text;

namespace lab2

{

    class Door

    {

*public* int width { get; }

*public* int height { get; }

*public* bool hasKey { get; set; }

*public* Door(int width, int height, bool hasKey)

        {

            this.width = width;

            this.height = height;

            this.hasKey = hasKey;

        }

*public* *override* bool Equals(object obj)

        {

*if* (obj == null) *return* false;

            Door d = obj as Door;

*if* (d as Door == null) *return* false;

*return* this.width == d.width && this.height == d.height && this.hasKey == d.hasKey;

        }

*public* *override* int GetHashCode()

        {

            int key;

*if* (hasKey)

                key = 17;

*else*

                key = 37;

*return* 31 \* width \* height \* key;

        }

*public* *override* string ToString() { *return* "{ Width - " + width + "; Height - " + height + "; Has key - " + hasKey + " }"; }

    }

}

**Class House**

using System;

using System.Collections.Generic;

using System.Text;

using System.Linq;

namespace lab2

{

    class House

    {

*public* List<Window> windows { get; }

*public* List<Door> doors { get; }

*public* House(List<Window> windows, List<Door> doors)

        {

            this.windows = windows;

            this.doors = doors;

        }

*public* *override* bool Equals(object obj)

        {

*if* (obj == null) *return* false;

            House h = obj as House;

*if* (h as House == null) *return* false;

*return* this.windows.SequenceEqual(h.windows) && this.doors.SequenceEqual(h.doors);

        }

*public* *override* int GetHashCode()

        {

            int result = 31;

*foreach* (Window window *in* windows)

            {

                result \*= window.GetHashCode();

            }

*foreach* (Door door *in* doors)

            {

                result \*= door.GetHashCode();

            }

*return* result;

        }

*public* *override* string ToString()

        {

            StringBuilder result = new StringBuilder("Windows:\n");

            int wCounter = 1;

            int dCounter = 1;

*foreach* (Window window *in* windows)

            {

                result.Append("Window #").Append(wCounter).Append(window.ToString()).Append("\n");

                wCounter++;

            }

            result.Append("\n")

                .Append("Doors:\n");

*foreach* (Door door *in* doors)

            {

                result.Append("Door #").Append(dCounter).Append(door.ToString()).Append("\n");

                dCounter++;

            }

*return* result.ToString();

        }

*public* string closeHouse()

        {

            bool hasKey = false;

*foreach* (Door door *in* doors)

            {

*if* (door.hasKey) hasKey = true;

            }

*if* (hasKey)

*return* "The house has been closed";

*else*

*return* "It is not possible to close the house";

        }

*public* int getAmountOfWindows()

        {

*return* windows.Count();

        }

*public* int getAmountOfDoors()

        {

*return* doors.Count();

        }

    }

}

**Class Program(Головний клас)**

using System;

using System.Collections.Generic;

namespace lab2

{

    class Program

    {

*static* void Main(string[] args)

        {

            List<Window> windows = new List<Window>();

            Window w1 = new Window(120, 60);

            Window w2 = new Window(140, 70);

            Window w3 = new Window(280, 140);

            Window w4 = new Window(140, 70);

            windows.Add(w1);

            windows.Add(w2);

            windows.Add(w3);

            windows.Add(w4);

            Console.WriteLine("First window equal the second one -> " + w1.Equals(w2));

            Console.WriteLine("First window's hashCode -> " + w1.GetHashCode());

            Console.WriteLine("Second window's hashCode -> " + w2.GetHashCode());

            Console.WriteLine("First window:\n" + w1);

            Console.WriteLine("Second window:\n" + w2);

            Console.WriteLine("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");

            Console.WriteLine("Second window equal the fourth one -> " + w2.Equals(w4));

            Console.WriteLine("Second window's hashCode -> " + w2.GetHashCode());

            Console.WriteLine("Fourth window's hashCode -> " + w4.GetHashCode());

            Console.WriteLine("Second window:\n" + w2);

            Console.WriteLine("Fourth window:\n" + w4);

            Console.WriteLine("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");

            List<Door> doors = new List<Door>();

            Door d1 = new Door(80, 190, true);

            Door d2 = new Door(80, 190, false);

            Door d3 = new Door(70, 140, true);

            Door d4 = new Door(80, 190, false);

            doors.Add(d1);

            doors.Add(d2);

            doors.Add(d3);

            doors.Add(d4);

            Console.WriteLine("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");

            Console.WriteLine("First door equal the second one -> " + d1.Equals(d2));

            Console.WriteLine("First door's hashCode -> " + d1.GetHashCode());

            Console.WriteLine("Second door's hashCode -> " + d2.GetHashCode());

            Console.WriteLine("First door:\n" + d1);

            Console.WriteLine("Second door:\n" + d2);

            Console.WriteLine("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");

            Console.WriteLine("Second door equal the fourth one -> " + d2.Equals(d4));

            Console.WriteLine("Second door's hashCode -> " + d2.GetHashCode());

            Console.WriteLine("Fourth door's hashCode -> " + d4.GetHashCode());

            Console.WriteLine("Second door:\n" + d2);

            Console.WriteLine("Fourth door:\n" + d4);

            Console.WriteLine("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");

            House house = new House(windows, doors);

            House houseCopy = new House(windows, doors);

            Console.WriteLine("First house:\n" + house);

            Console.WriteLine("Second house:\n" + houseCopy);

            Console.WriteLine("First house equals to second one -> " + house.Equals(houseCopy));

            Console.WriteLine("First house's hashCode -> " + house.GetHashCode());

            Console.WriteLine("Second house's hashCode -> " + houseCopy.GetHashCode());

            Console.WriteLine("Amount of Windows in first house -> " + house.getAmountOfWindows()

                + "\nAmount of doors in first house -> " + house.getAmountOfDoors());

            Console.WriteLine("Close the first house -> " + house.closeHouse());

            Console.WriteLine("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");

            List<Door> anotherDoors = new List<Door>();

            Door do1 = new Door(80, 190, false);

            Door do2 = new Door(80, 350, false);

            Door do3 = new Door(70, 140, false);

            Door do4 = new Door(80, 190, false);

            anotherDoors.Add(do1);

            anotherDoors.Add(do2);

            anotherDoors.Add(do3);

            anotherDoors.Add(do4);

            House anotherHouse = new House(windows, anotherDoors);

            Console.WriteLine("First house:\n" + house);

            Console.WriteLine("Another house:\n" + anotherHouse);

            Console.WriteLine("First house equals to another one -> " + house.Equals(anotherHouse));

            Console.WriteLine("First house's hashCode -> " + house.GetHashCode());

            Console.WriteLine("Another house's hashCode -> " + anotherHouse.GetHashCode());

            Console.WriteLine("First house:\n" + house);

            Console.WriteLine("Second house(copy):\n" + houseCopy);

*// exactly this house doesn't have doors with keys, this way it isn't possible to close it*

            Console.WriteLine("Another house:\n" + anotherHouse);

            Console.WriteLine("Close another house -> " + anotherHouse.closeHouse());

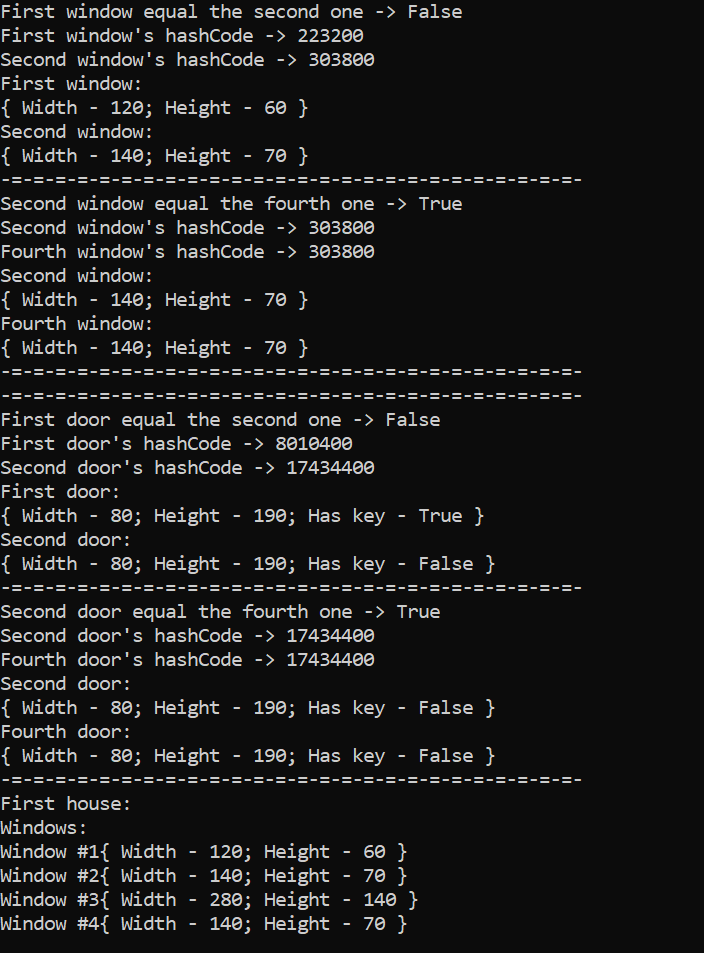
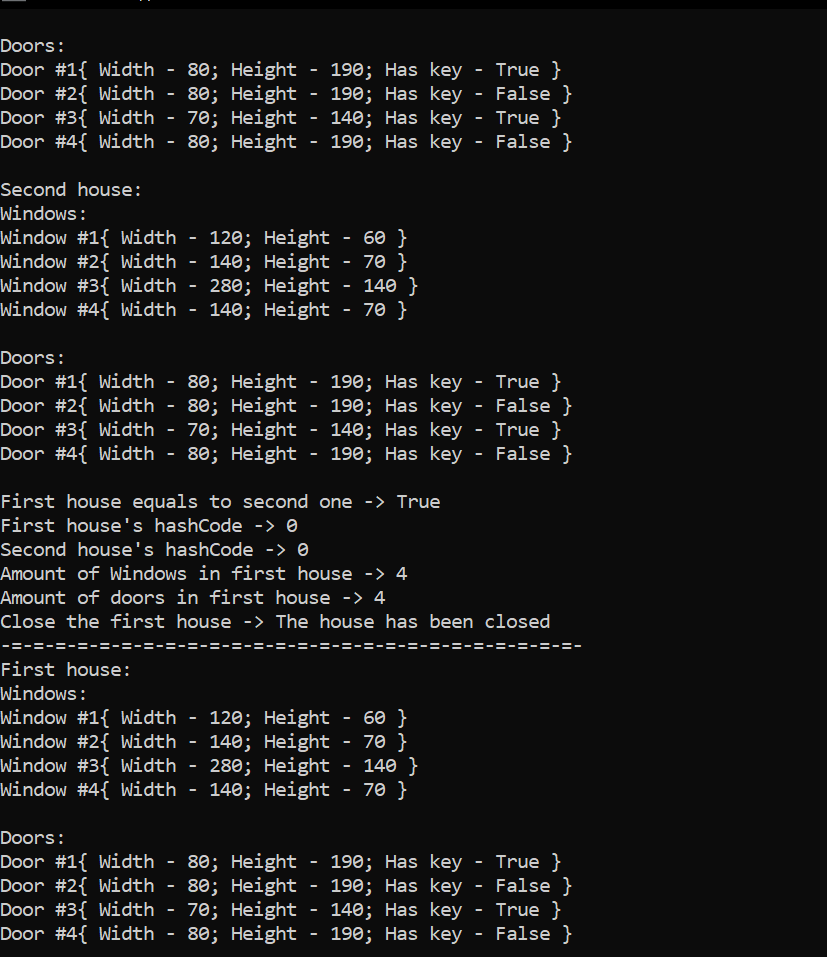
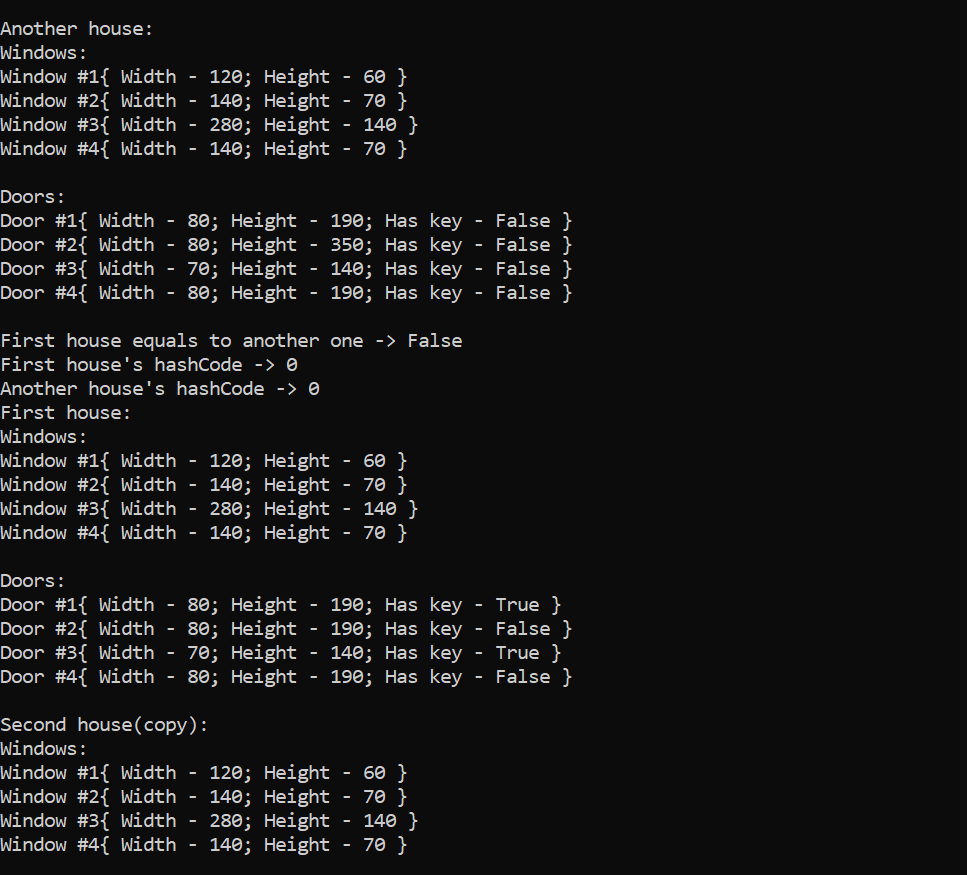
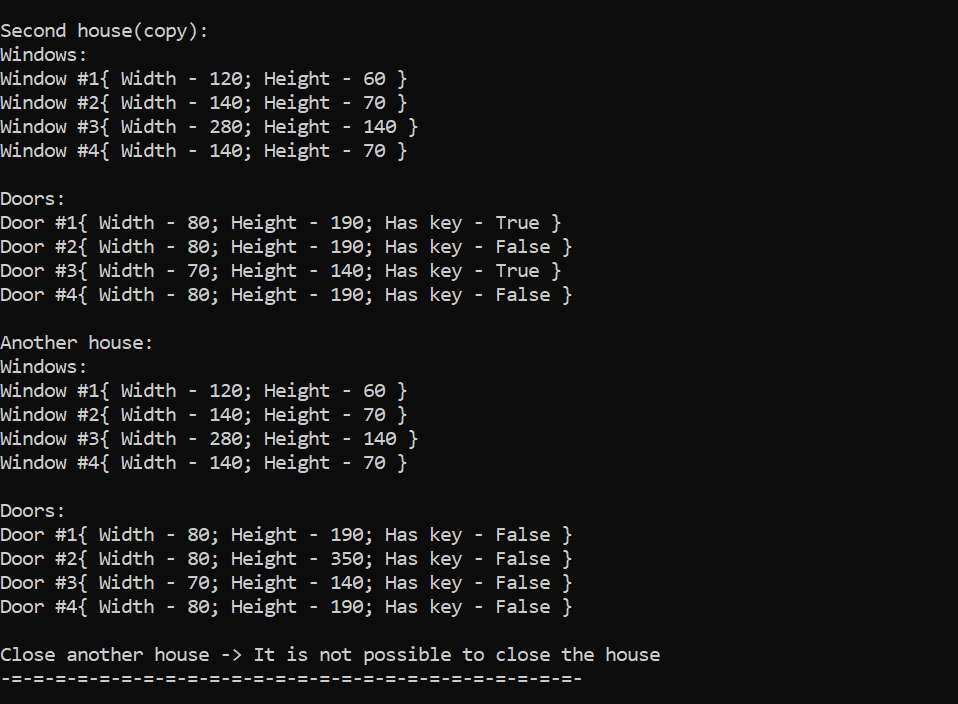
            Console.WriteLine("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-");

        }

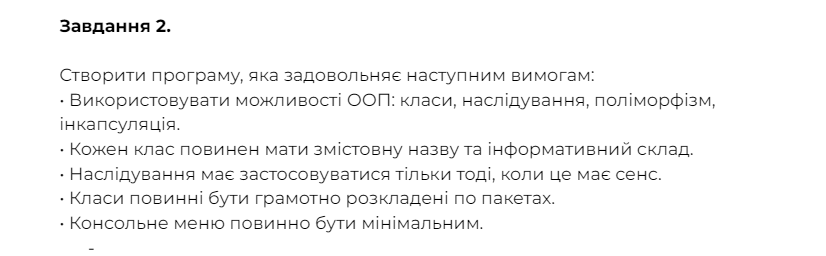
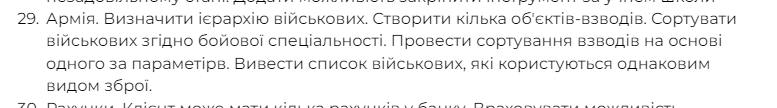
    }

}

**Результат виконання програми:**

**** ****  

Завдання №2

**** ****

**Лістинг програми**

Тут будуть тільки абстрактні класи, щоб не забивати документ

**Abstract class Soldier**

using lab2.\_1.weapons;

using System;

using System.Collections.Generic;

using System.Text;

namespace lab2.\_1.soldiers

{

*public* *abstract* class Soldier

    {

*protected* string name, rank, regiment, speciality;

*protected* Weapon weapon;

*public* Soldier(string name, string regiment, string speciality, Weapon weapon)

        {

            this.name = name;

            this.regiment = regiment;

            this.speciality = speciality;

            this.weapon = weapon;

        }

*public* void greatColleague(Soldier soldier) { Console.WriteLine("I wish you well, comrade " + soldier.getRank()); }

*public* *abstract* void shoot();

*// ToString(), Equals(), HashCode()*

*public* *override* bool Equals(object obj)

        {

*if* (obj == null)

*return* false;

            Soldier s = obj as Soldier;

*if* (s == null) *return* false;

*return* getName() == s.getName() && getRank() == s.getRank() && getRegimant() == s.getRegimant()

                && getSpeciality() == s.getSpeciality() && getWeapon().Equals(s.getWeapon());

        }

*public* *override* int GetHashCode()

        {

*return* 31 \* getWeapon().getId();

        }

*public* *override* string ToString()

        {

*return* getRank() + ": { " + getName() + ", " + getRegimant() + ", " + getSpeciality() + ", " + getWeapon() + " }";

        }

*// getters and setters*

*public* string getName() { *return* this.name; }

*public* string getRank() { *return* this.rank; }

*public* string getRegimant() { *return* this.regiment; }

*public* string getSpeciality() { *return* this.speciality; }

*public* Weapon getWeapon() { *return* this.weapon; }

*public* void setRank(string rank)

        {

*if* (rank.Length == 0 || rank == null)

            {

                Console.WriteLine("length of the rank cannot be zero.");

*return*;

            }

*else*

            {

                this.rank = rank;

            }

        }

*public* void setRegiment(string regiment)

        {

*if* (regiment.Length == 0 || regiment == null)

            {

                Console.WriteLine("length of the regiment cannot be zero.");

*return*;

            }

*else*

            {

                this.regiment = regiment;

            }

        }

*public* void setSpeciality(string speciality)

        {

*if* (speciality.Length == 0 || speciality == null)

            {

                Console.WriteLine("length of the speciality cannot be zero.");

*return*;

            }

*else*

            {

                this.speciality = speciality;

            }

        }

*public* void setWeapon(Weapon weapon)

        {

*if* (weapon == null)

            {

                Console.WriteLine("Weapon cannot be null");

*return*;

            }

*else*

            {

                this.weapon = weapon;

            }

        }

    }

}

**Abstract class Weapon**

using System;

using System.Collections.Generic;

using System.Text;

namespace lab2.\_1.weapons

{

*public* *abstract* class Weapon

    {

*private* *static* int generalIdNumber = 1;

*protected* int id;

*protected* string type;

*public* Weapon() { this.id = generalIdNumber++; }

*public* void makeShot()

        {

            Console.WriteLine("Weapon \"" + type + "\" is shooting.");

        }

*// getters and setters*

*public* int getId() { *return* id; }

*public* string getTypeOfWeapon()

        {

*return* type;

        }

*public* *override* bool Equals(object obj)

        {

*if* (obj == null) *return* false;

            Weapon w = obj as Weapon;

*if* (w == null) *return* false;

*return* type == w.type && id == w.id;

        }

*public* *override* int GetHashCode()

        {

*return* id;

        }

*public* *override* string ToString()

        {

*return* "Weapon: { " + type + ", id number -> " + id + " }";

        }

    }

}

**Main class**

using lab2.\_1.soldiers;

using lab2.\_1.weapons;

using System;

using System.Collections.Generic;

namespace lab2.\_1

{

    class Program

    {

*static* void Main(string[] args)

        {

            List<Soldier> squad1 = new List<Soldier>();

            squad1.Add(new Ranker("Степан Степанов", "полк #1", "Разведчик", new Pistol()));

            squad1.Add(new Major("Иван Иванов", "полк #2", "Танкист", new MachineGun()));

            squad1.Add(new Lieutenant("Андрей Андреев", "полк #1", "Зенитчик", new Pistol()));

            squad1.Add(new General("Виктор Баранов", "полк #3", "Десантник", new MachineGun()));

            squad1.Add(new Ensign("Виталий Козлов", "полк #6", "Артиллерист", new Pistol()));

            squad1.Add(new Ensign("Сергей Матвиенко", "полк #6", "Танкист", new Pistol()));

            squad1.Add(new Ensign("Владимир Федорченко", "полк #6", "Танкист", new Pistol()));

            List<Soldier> squad2 = new List<Soldier>();

            squad2.Add(new Sergeant("Степан Бандера", "полк #1", "Артиллерист", new Pistol()));

            squad2.Add(new Major("Владимир Кличко", "полк #2", "Десантник", new MachineGun()));

            squad2.Add(new Lieutenant("Виталий Кличко", "полк #1", "Разведчик", new MachineGun()));

            squad2.Add(new General("Роман Нагорный", "полк #3", "Разведчик", new MachineGun()));

            squad2.Add(new Ensign("Александр Македонский", "полк #6", "Десантник", new Pistol()));

            squad2.Add(new Ensign("Иван Будько", "полк #6", "Зенитчик", new Pistol()));

            squad2.Add(new Ensign("Михаил Литвиносов", "полк #6", "Зенитчик", new Pistol()));

*// getting soldiers from first squad with the gun -> Pistol*

            printSoldiersWithTheSameGun(squad1, new Pistol());

            Console.WriteLine("-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");

*// getting soldiers from second squad with the gun -> MachineGun*

            printSoldiersWithTheSameGun(squad2, new MachineGun());

            squad1 = getSortedSquadBySpeciality(squad1);

*foreach* (Soldier soldier *in* squad1)

            {

                Console.WriteLine(soldier);

            }

        }

*public* *static* void printSoldiersWithTheSameGun(List<Soldier> squad, Weapon weapon)

        {

            Console.WriteLine("Soldiers from squad with the gun -> " + weapon.getTypeOfWeapon());

*foreach* (Soldier soldier *in* squad)

*if* (soldier.getWeapon().getTypeOfWeapon().Equals(weapon.getTypeOfWeapon())) Console.WriteLine(soldier.getName());

        }

*// Сортировка происходит по военной специальности. Например: Вначале будет Артеллирист, а потом уже Десантник*

*public* *static* List<Soldier> getSortedSquadBySpeciality(List<Soldier> squad)

        {

            List<Soldier> sortedSquad = new List<Soldier>();

            SortedSet<string> specialities = new SortedSet<string>();

*foreach* (Soldier soldier *in* squad)

            {

                specialities.Add(soldier.getSpeciality());

            }

*foreach* (string specility *in* specialities)

            {

*foreach* (Soldier soldier *in* squad)

                {

*if* (soldier.getSpeciality().Equals(specility))

                    {

                        sortedSquad.Add(soldier);

                    }

                }

            }

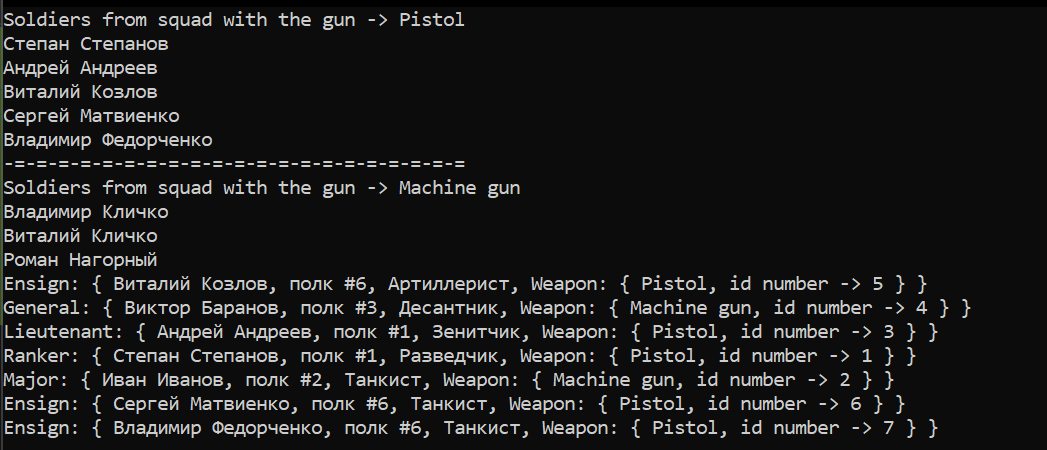
*return* sortedSquad;

        }

    }

}

**Результат виконання програми**

****