Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

Звіт

з лабораторної роботи №4 з дисципліни
“Програмування 2. Структури даних та алгоритми”

“ДОСЛІДЖЕННЯ МЕХАНІЗМУ УСПАДКУВАННЯ КЛАСІВ”

Варіант №13

Виконав студент Флорчук Назарій Петрович

Перевірив викладач Проскура Світлана Леонідівна

Мета роботи:

1. Дослідити механізм успадкування класів.

Завдання:

1. Дослідити механізм однорівневого успадкування класів мовами програмування C++ та C#.
2. Написати програми мовами C++ та C#, що демонструють застосування механізму успадкування класів згідно з варіантом.

Опис базового класу	Опис похідного класу	Завдання
Рядки: значення рядка, конструктор з параметром, метод обчислення довжини рядка.	Цифрові рядки: конструктор з параметром, метод видалення з рядка заданого символу методи отримання даних об’єкту.	Описати класи, об’єкт похідного класу; вивести рядок; обчислити і вивести довжину.

Код програми (C++):

```
#include <iostream>

/**
 * Base string class
 */
class Base
{
    protected:
        std::string _m_string;

    public:
        Base(std::string string)
        {
            this->_m_string = string;
        }

        /**
         * Get string length
         */
        unsigned int Length()
        {
            return this->_m_string.length();
        }

        ~Base()
        {
            this->_m_string.clear();
        }
};

/**
 * Child numeric string class
 */
class Child: public Base
{
    public:
        Child(unsigned int number): Base(std::to_string(number))
        {
            //
        }

        /**
         * Get string
         */
        std::string Get()
        {
            return this->_m_string;
```

```
    }

    /**
     * Remove character from the string
     */
    void Remove(char character)
    {
       for (unsigned int i = 0; i < this->_m_string.length(); i++)
       {
          if (this->_m_string[i] == character)
          {
             this->_m_string.erase(i, 1);

             i--;
          }
       }
    }
};

int main()
{
   Child string(1234567890);

   std::cout << "____ Initial Child object (numeric string):" << std::endl;
   std::cout << string.Get() << std::endl;

   std::cout << "____ Child object (numeric string) length:" << std::endl;
   std::cout << string.Length() << std::endl;

   string.Remove('5');

   std::cout << "____ Child object (numeric string), after removing '5' character:" << std::endl;
   std::cout << string.Get() << std::endl;

   std::cout << "____ Child object (numeric string) length, after removing '5' character:" << std::endl;
   std::cout << string.Length() << std::endl;

   string.Remove('2');

   std::cout << "____ Child object (numeric string), after removing '2' character:" << std::endl;
   std::cout << string.Get() << std::endl;

   std::cout << "____ Child object (numeric string) length, after removing '2' character:" << std::endl;
   std::cout << string.Length() << std::endl;

   exit(0);
}
```

```
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/lab_4$ g++ lab_4.cpp -o lab_4
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/lab_4$ ./lab_4
____ Initial Child object (numeric string):
1234567890
____ Child object (numeric string) length:
10
____ Child object (numeric string), after removing '5' character:
123467890
____ Child object (numeric string) length, after removing '5' character:
9
____ Child object (numeric string), after removing '2' character:
13467890
____ Child object (numeric string) length, after removing '2' character:
8
```

Код програми (C#):

```
/**
 * Base string class
 */
class Base
{
    protected string _m_string;

    public Base(string row)
    {
        this._m_string = row;
    }

    /**
     * Get string length
     */
    public int Length()
    {
        return this._m_string.Length;
    }
};

/**
 * Child numeric string class
 */
class Child: Base
{
    public Child(int number): base(number.ToString())
    {
        //
    }

    /**
     * Get string
     */
    public string Get()
    {
        return this._m_string;
    }

    /**
     * Remove character from the string
     */
    public void Remove(char character)
    {
        for (int i = 0; i < this._m_string.Length; i++)
        {
            if (this._m_string[i] == character)
```

```
            {
                this._m_string = this._m_string.Remove(i, 1);

                i--;
            }
        }
    }
};

class Application
{
    static void Main(string[] args)
    {
        Child row = new Child(1234567890);

        System.Console.WriteLine("____ Initial Child object (numeric string):");
        System.Console.WriteLine(row.Get());

        System.Console.WriteLine("____ Child object (numeric string) length:");
        System.Console.WriteLine(row.Length());

        row.Remove('5');

        System.Console.WriteLine("____ Child object (numeric string), after removing '5' character:");
        System.Console.WriteLine(row.Get());

        System.Console.WriteLine("____ Child object (numeric string) length, after removing '5'
character:");
        System.Console.WriteLine(row.Length());

        row.Remove('2');

        System.Console.WriteLine("____ Child object (numeric string), after removing '2' character:");
        System.Console.WriteLine(row.Get());

        System.Console.WriteLine("____ Child object (numeric string) length, after removing '2'
character:");
        System.Console.WriteLine(row.Length());
    }
}
```

```
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/lab_4$ mcs lab_4.cs
[redacted]@debian:~/Documents/kpi/basics_of_programming_2/lab_4$ mono lab_4.exe
____ Initial Child object (numeric string):
1234567890
____ Child object (numeric string) length:
10
____ Child object (numeric string), after removing '5' character:
123467890
____ Child object (numeric string) length, after removing '5' character:
9
____ Child object (numeric string), after removing '2' character:
13467890
____ Child object (numeric string) length, after removing '2' character:
8
```

Висновки / Відповіді на контрольні запитання:

1. У чому полягає сутність механізму успадкування?
   Однією із проблем у програмуванні є повторне використання створеного коду та його модифікація. У об’єктно-орієнтованому програмуванні цю проблему можна розв’язати із використанням успадкування. Успадкування — один із принципів об’єктно-орієнтованого програмування, згідно якого новий клас (клас-нащадок) описується на основі вже класу що існує (класу-предка). При цьому клас-нащадок автоматично успадковує дані (поля) та функціональні можливості (методи) класу-предка.
2. Розкажіть, які бувають типи успадкування.
   Якщо клас (клас-нащадок) успадковується від одного класу (класу-предка), таке успадкування називається *одиночним*. А якщо клас (клас-нащадок) успадковується від декількох класів (класів-предків), таке успадкування називається *множинним.*
3. Поясніть роль специфікатора доступу в успадкуванні.
   Специфікатори доступу дають змогу керувати доступом до полів та методів класу.
   - Якщо поле / метод, оголошено із специфікатором доступу *public*, то поле / метод буде доступним для використання ззовні екземпляра об’єкту, в якому вони оголошені, та класам-нащадкам.
   - Якщо поле / метод, оголошено із специфікатором доступу *protected*, то поле / метод буде доступним для використання в методах класу, в якому вони оголошені, та в методах класів-нащадків.
   - Якщо поле / метод, оголошено із специфікатором доступу *private*, то поле / метод буде доступним для використання в методах класу, в якому вони оголошені.
4. Поясніть сутність одиночного успадкування, наведіть приклад.
   Клас-нащадок успадковується лише від одного класу-предка.
5. Поясніть сутність множинного успадкування, наведіть приклад.
   Клас-нащадок успадковується від декількох класів-предків.
6. Поясніть, чому в C# немає множинного спадкування.
   C# не підтримує множинне успадкування, оскільки розробники мови, вважали, що додавання множинного успадкування додає надто багато складності C#, надаючи надто мало переваг. Але з цієї ситуації можна вийти з використанням interface’ів.
7. Поясніть різницю між прямим базовим класом та непрямим.
   - Прямий базовий клас — від цього класу виконується успадкування безпосередньо при оголошенні похідного класу.
   - Непрямий базовий клас — базовий клас, для одного із базових класів, від який виконується успадкування похідним класом.
8. Поясніть сутність успадкування на основі непрямих розподілених базових класів.
   -
9. Поясніть сутність спадкування на основі непрямих віртуальних базових класів.
   -
10. Поясніть, як керувати викликом конструкторів базового класу у конструкторі похідного класу, наведіть приклад.
    -
11. Як та з якою ціллю приховати член базового класу? Наведіть приклад приховування.
    Якщо необхідно приховати метод, або поле базового класу, від похідного класу, то це можна зробити за допомогою специфікатор доступу *private*, для заданого методу або поля. Якщо ж необхідно регулювати видимість методів, або полів, класу на етапі успадкування, то можна використовувати специфікатори доступу класу при успадкуванні:

- *public* — у цьому випадку, публічні члени базового класу стають публічними членами похідного класу, а захищені члени базового класу стають захищеними членами похідного класу;
- *protected* — у цьому випадку публічні і захищені члени базового класу стають захищеними членами похідного класу;
- *private* — у цьому випадку публічні і захищені члени базового класу стають приватними членами похідного класу.