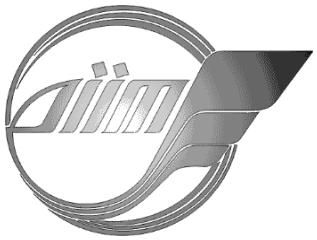
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



Дніпровський національний університет
залізничного транспорту імені академіка В. Лазаряна

Кафедра «Комп'ютерні інформаційні технології»

Лабораторна робота №8

з дисципліни «Об'єктно-орієнтоване програмування»

на тему: «Множинне спадкування»

Виконав:
студент гр.ПЗ1911
Сафонов Д.Є.
Прийняла:
Демидович І. М.

Дніпро, 2020

Тема. Множинне спадкування

1 Постановка задачі згідно загального та індивідуального завдання.

Для будь-якої предметної області продемонстувати різницю між віртуальним та звичайним множинним спадкуванням.

2 Текст програми.

```
#include<iostream>


template<class T>
class Private {
protected:
	T data;
	Private() {}//needed to allow child classes to empty on init
public:
	Private(const T& data) : data(data) {}
};


template<class T>
class PrivateSetable : public virtual Private<T> {
public:
	PrivateSetable() {}
	virtual void set(const T& data) { this->data = data;}
};


template<class T>
class PrivateGetable : public virtual Private<T> {
protected:
	PrivateGetable() {}
public:
```

```
	PrivateGetable(const T& data) : Private<T>(data) {}
	virtual T get() const {return this->data;}
};

template<class T>
class Public : public PrivateSetable<T>, public PrivateGetable<T> {
	bool _engaged;
public:
	Public() : _engaged(false) {}
	Public(const T& data) : PrivateGetable<T>(data), _engaged(true) {}

	T get() const override {
		if(!_engaged)
			throw(std::runtime_error("Getting value from not initialized Public object"));
		return PrivateGetable<T>::get();
	}

	void set(const T& data) override {
		if(!_engaged)
			_engaged = true;
		PrivateSetable<T>::set(data);
	}

	operator bool() {return _engaged;}
};

int main() {
	//Private<int> pint_;//error
```

```
	Private<int> pint(7);//cant get or set this value, kind of useless(imagine it has some inner functioality

	//PrivateGetable<int> pgint_;//error
	PrivateGetable<int> pgint(7);//can get value, but cant set, same as const
	std::cout << "PrivateGetable::get() " << pgint.get() << "\n";

	PrivateSetable<int> psint;//can be empty on init
	psint.set(7);//can change/initialize afterwards

	Public<int> pubint;
	std::cout << "Public::operator bool() on empty " << std::boolalpha << bool(pubint) << "\n";
	//pubint.get();//throws exception
	pubint.set(7);
	std::cout << "Public::operator bool() on initialized " << std::boolalpha << bool(pubint) << "\n";
	std::cout << "Public::get() " << pubint.get() << "\n";
}
```

3 Результати виконання.

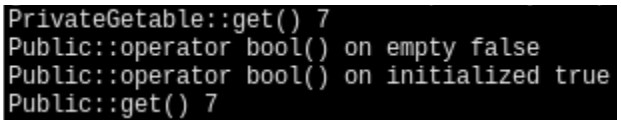


Рисунок 1

4 Висновок.

Множинне спадкування необхідне, коли клас використовує методи декількох класів.

Це можна замінити зіставни класами. Але якщо необхідно успадкувати від класів, які успадковають від одного класу, з’являється проблема — у фінального класу буде два єкземпляри функцій та атрибутів суперкласу. Ця проблема вирішуется віртуальним спадкуванням(якщо успадкувати від класів, які у свою чергу віртуально успадковани від одного класу, то у фінального класу буде лише одна копія базоваго класу).