Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

ОТЧЕТ

по лабораторной работе

# «МНОЖЕСТВЕННОЕ НАСЛЕДОВАНИЕ, ВИРТУАЛЬНОЕ НАСЛЕДОВАНИЕ»

| Студент гр. 124404 |  | Н.А. Мойсенович |
| --- | --- | --- |
| Проверила |  | К.А. БОРЩЕВИЧ |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Минск 2022

# Цель

# Изучить принципы и получить практические навыки при использовании множественного наследования; рассмотреть случаи, когда необходимо использовать виртуальное наследование.

# Краткие теоретические сведения

Иерархическая структура, в которой производный класс одновременно наследуется от нескольких базовых классов, называется множественным наследованием. В этом случае производный класс, имея собственные компоненты, имеет доступ к protected- и public-компонентам базовых классов. Конструкторы базовых классов при создании объекта производного класса вызываются в том порядке, в котором они указаны в списке при объявлении производного класса.

При применении множественного наследования возможно возникновение нескольких конфликтных ситуаций. Первая конфликтная ситуация возникает из-за конфликта имен методов или полей нескольких базовых классов. Вторая конфликтная ситуация возникает при многократном включении некоторого базового класса. Решение этой проблемы состоит в использовании виртуального наследования (***virtual inheritance***).

Виртуальное наследование базового класса гарантирует, что в любом производном классе будет присутствовать только одна его копия.

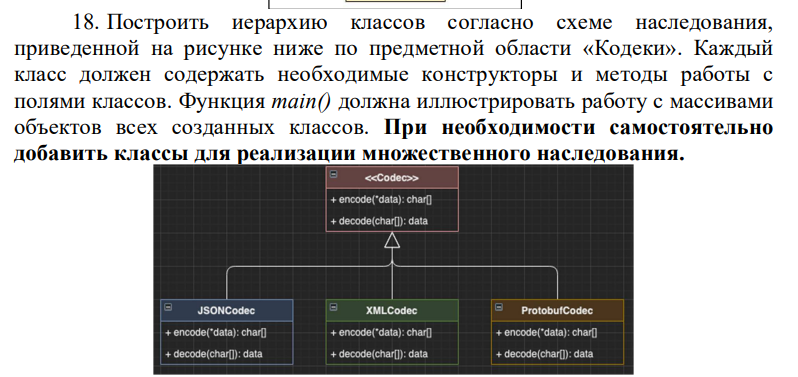
Применение оператора "::" (разрешения контекста, разрешения области видимости) позволяет программе "ручным способом" выбрать версию класса. Но если необходима только одна копия базового класса, то это достигается с помощью виртуального наследования базовых классов.

Если два (или больше) класса выведены из общего базового класса, то можно предотвратить включение нескольких копий базового класса в объекте дочерних классов, выведенного из этих классов, что реализуется путем объявления наследовании базового класса виртуальным. Для этого достаточно предварить имя наследуемого базового класса ключевым словом *virtual*.

Разница между обычным базовым и виртуальным классами становится очевидной только тогда, когда этот базовый класс наследуется более одного раза. Если наследование базового класса объявляется виртуальным, то только один его экземпляр будет включен в объект наследующего класса. В противном случае в этом объекте будет присутствовать несколько его копий.

# Вариант 18

# Общая постановка задачи



**Листинг файла Codec.h**

#pragma once

#include <string>

#include <iostream>

using std::string;

using std::cout;

using std::cin;

using std::endl;

class Codec

{

protected:

int SomeGeneralData;

public:

Codec();

virtual void Encode() = 0;

virtual void Decode() = 0;

int GetSomeGeneralData();

};

class JSONCodec: public virtual Codec

{

public:

JSONCodec();

void Encode() override;

void Decode() override;

};

class XMLCodec : public virtual Codec

{

public:

XMLCodec();

void Encode() override;

void Decode() override;

};

class ProtobuffCodec : public Codec

{

public:

ProtobuffCodec();

void Encode() override;

void Decode() override;

};

class JSONToXMLCodec : public JSONCodec, public XMLCodec

{

public:

JSONToXMLCodec();

void Encode() override;

void Decode() override;

};

**Листинг файлa Codec.cpp**

#include "Codec.h"

/\*Сделаем вид что ф - ии Encode Decode реально декадируют и кодируют данные

в строку и из, а пока просто будет возвращать enum в строку и обратно\*/

Codec::Codec()

{

cout << "Called constructor of Codec" << endl;

SomeGeneralData = 10;

}

JSONCodec::JSONCodec()

{

cout << "Called constructor of JSONCodec" << endl;

}

XMLCodec::XMLCodec()

{

cout << "Called constructor of XMLCodec" << endl;

}

ProtobuffCodec::ProtobuffCodec()

{

cout << "Called constructor of ProtobuffCodec" << endl;

}

JSONToXMLCodec::JSONToXMLCodec()

{

cout << "Called constructor of JSONToXMLCodec" << endl;

}

int Codec::GetSomeGeneralData()

{

return SomeGeneralData;

}

void JSONCodec::Encode()

{

cout << "Called Encode JSONCodec" << endl;

}

void JSONCodec::Decode()

{

cout << "Called Decode JSONCodec" << endl;

}

void XMLCodec::Encode()

{

cout << "Called Encode XMLCodec" << endl;

}

void XMLCodec::Decode()

{

cout << "Called Decode XMLCodec" << endl;

}

void ProtobuffCodec::Encode()

{

cout << "Called Encode ProtobuffCodec" << endl;

}

void ProtobuffCodec::Decode()

{

cout << "Called Decode ProtobuffCodec" << endl;

}

void JSONToXMLCodec::Encode()

{

cout << "Called Encode JSONToXMLCodec" << endl;

}

void JSONToXMLCodec::Decode()

{

cout << "Called Decode JSONToXMLCodec" << endl;

}

**Листинг файла Source.cpp**

#include "Codec.h"

void ShowWorkMenu(Codec\*\* codecs, int size)

{

for (int i = 0; i < size; i++)

{

int answer;

cout << "What you wanna do with your codecs consistensly" << endl;

cout << "1.Encode" << endl;

cout << "2.Decode" << endl;

cout << "3.Show some general data" << endl;

cout << "Or press any other button to exit" << endl;

cin >> answer;

switch (answer)

{

default:

exit(0);

case 1:

codecs[i]->Encode();

break;

case 2:

codecs[i]->Decode();

break;

case 3:

cout << codecs[i]->GetSomeGeneralData() << endl;

break;

}

}

}

void ShowMenu(int size)

{

Codec\*\* codecs = new Codec\*[size];

for (int i = 0; i < size; i++)

{

int answer;

cout << "What you wanna create and work with?" << endl;

cout << "1.JSONCodec" << endl;

cout << "2.XMLCodec" << endl;

cout << "3.ProtobuffCodec" << endl;

cout << "4.JSONToXMLCodec" << endl;

cout << "Or input any other button to exit programm" << endl;

cin >> answer;

switch (answer)

{

default:

delete[] codecs;

exit(0);

break;

case 1:

{

JSONCodec jsonCodec;

codecs[i] = &jsonCodec;

}

break;

case 2:

{

XMLCodec xmlCodec;

codecs[i] = &xmlCodec;

}

break;

case 3:

{

ProtobuffCodec protobuffCodec;

codecs[i] = &protobuffCodec;

}

break;

case 4:

{

JSONToXMLCodec jsonToXMLCodec;

codecs[i] = &jsonToXMLCodec;

}

break;

}

}

ShowWorkMenu(codecs, size);

}

void main()

{

bool correctInput = false;

int size;

while (true)

{

if (correctInput == false)

{

cout << "Enter a size of codec array" << endl;

cin >> size;

if (cin)

correctInput = true;

cin.clear();

cin.ignore(1, '\n');

}

else

{

ShowMenu(size);

}

}

}

# UML-диаграмма

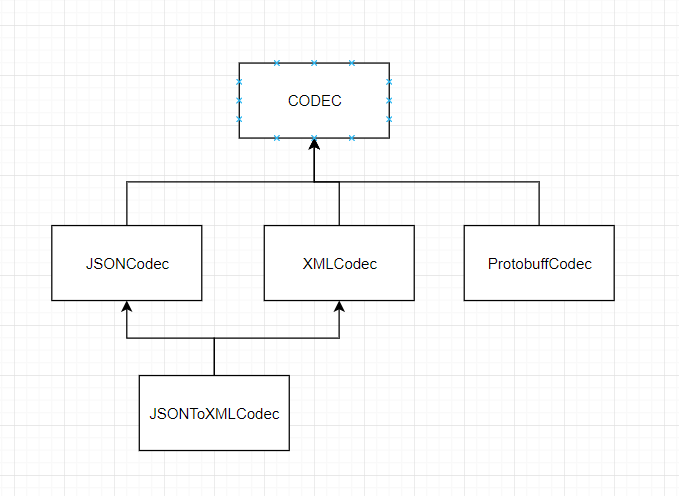


Рисунок 2 – Диаграмма классов UML

# Вывод

Были изучены принципы и получены практические навыки при использовании множественного наследования; рассмотрены случаи, когда необходимо использовать виртуальное наследование.