

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт  информационных технологий** | **Кафедра информационных систем** |

|  |  |  |
| --- | --- | --- |
| **Основная образовательная программа 09.03.02 «Информационные системы и технологии»**  **Отчет по дисциплине «Объектно-ориентированное программирование на языке высокого уровня»** | | |
| **по лабораторной работе №4** | | |
|  | | |
|  | | |
| **Проверил**  **ассистент** |  | **Михайлуца Н.А.** |
|  |  |  |
| **Выполнил**  **студент группы ИДБ-22-06** |  | **Мустафаева П.М.** |

ОГЛАВЛЕНИЕ

[ЗАДАНИЕ 1 2](#_Toc149855007)

[1.1 ОПИСАНИЕ 2](#_Toc149855008)

[1.2 ПРОГРАММА 3](#_Toc149855009)

[1.3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ 4](#_Toc149855010)

[ЗАДАНИЕ 2 4](#_Toc149855011)

[2.1 ОПИСАНИЕ 4](#_Toc149855012)

[2.2 ПРОГРАММА 5](#_Toc149855013)

[2.3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ 6](#_Toc149855014)

[ЗАКЛЮЧЕНИЕ 8](#_Toc149855015)

# **ЗАДАНИЕ 1**

## **1.1 ОПИСАНИЕ**

Задать два иерархически связанных полиморфных класса: деталь и сборка (один из них базовый). Все конструкторы находятся в зоне protected (чтобы запретить явное создание). В main() следует создать несколько экземпляров указанных типов, помещая их в хранилище (vector), используя шаблонную дружественную функцию.

Дополнить программу всем необходимым для ее правильной работы.

## **1.2 ПРОГРАММА**

#include <iostream>

#include <vector>

using namespace std;

class Detail {

protected:

Detail() {}

public:

virtual ~Detail() {}

template<typename T>

friend void append(vector<Detail\*>& v);

};

class Build : public Detail {

protected:

Build() {};

public:

virtual ~Build() {}

template<typename T>

friend void append(vector<Detail\*>& v);

};

template<typename T>

void append(vector<Detail\*>& v) {

v.push\_back(new T());

}

int main() {

vector<Detail\*> D;

append<Detail>(D);

append<Build>(D);

append<Detail>(D);

append<Build>(D);

for (int i = 0; i < D.size(); i++) {

delete D[i];

}

D.clear();

return 0;

}

## **1.3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

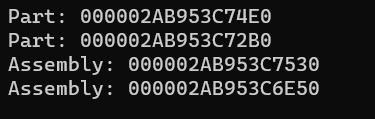


Рис. 1. Результат выполнения программы 1

# **ЗАДАНИЕ 2**

## **2.1 ОПИСАНИЕ**

Задать два иерархически связанных полиморфных класса: Base и Derived.

В главной функции программы последовательно выполняется два действия. В цикле случайно (random) создаются несколько экземпляров любого из заданных типов (например, по четному случайному числу создается Base, а по нечетному - Derived). После создания объектов, их копии передаются в функцию add, которая помещает их в общее хранилище.

Затем в main необходимо показать для каждого из имеющихся в хранилище объектов разницу в поведении их типа, доставая их по очереди из хранилища.

В заключении, освободить все ресурсы.

## **2.2 ПРОГРАММА**

#include <iostream>

#include <vector>

#include <ctime>

#include <random>

using namespace std;

class Base {

public:

int num;

Base() {

num = rand() % 100;

}

Base(const Base& obj) {

num = obj.num;

}

virtual Base\* copy() {

return new Base(\*this);

}

virtual void print()

{

cout << "Base" << endl;

}

virtual ~Base() {}

};

class Derived : public Base {

public:

int num;

Derived() {

num = rand() % 10;

}

Derived(const Derived& obj) {

num = obj.num;

}

virtual Derived\* copy() {

return new Derived(\*this);

}

void print()

{

cout << "Derived" << endl;

}

~Derived() {}

};

void add(const vector<Base\*>& v1, vector<Base\*>& v2)

{

for (int i = 0; i < v1.size(); i++)

v2.push\_back(v1[i]->copy());

}

int main()

{

srand(time(NULL));

vector<Base\*> v1;

vector<Base\*> v2;

for (int i = 0; i < 4; i++)

{

if (rand() % 10 % 2)

v1.push\_back(new Base);

else

v1.push\_back(new Derived);

}

add(v1, v2);

for (int i = 0; i < v1.size(); i++)

{

v1[i]->print();

}

return 0;

}

## **2.3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**



Рис. 2. Результат выполнения программы 2

# **ЗАКЛЮЧЕНИЕ**

В ходе лабораторной работы были реализованы возможности создания нескольких геометрических фигур и их визуализацию с использованием класса Vector2D.