15.10.22

# **Конструирование объектов класса**

#include <string>

#include <vector>

#include <iostream>

using namespace std;

class Logger {

public:

Logger(const string& s) : nameOfClass(s) {

cout << nameOfClass << " birth " << endl;

}

~Logger() {

cout << nameOfClass << " Dead like a hero " << endl;

}

string nameOfClass;

};

class MultHeroes {

public:

MultHeroes(const string& s) : L(s + "MultHeroes") {}

string typeOfHero = "MultHeroes";

const Logger L;// = Logger("MultHeroes");

};

class Ogr : public MultHeroes {

public:

Ogr(const string& s) : MultHeroes(s), L(s) {

//typeOfHero = "MultHeroes";

}

Logger L = Logger("Ogr");

};

class FamilyOgr {

public:

FamilyOgr() : Shreck("Shreck"), Fiona("Fiona"), miniShreck("miniShreck") {}

Logger L = Logger("Family");

Ogr Shreck;

Ogr Fiona;

Ogr miniShreck;

};

int main() {

FamilyOgr family;

}

22.10.22

SOLID

S – single responsibility

(принцип единственной ответственности)

O – open-close

(принцип открытости-закрытости)

L – Liskov substitution

(принцип подстановки Барбары Лисков)

I – interface segregation

(принцип разделения интерфейсов)

D – dependency inversion

(принцип инверсии зависимостей)

Singleton – порождающий шаблон проектирования, гарантирующий, что в однопоточном приложении будет единственный экземпляр некоторого класса, и представляющий глобальную точку доступа к этому экземпляру.

Безопасность:

Проблемы с безопасностью singleton:

1. Рефлексия – изменение во время исполнения класса
2. Десериализация – возможность создать объект
3. Поток и небезопасность -
4. Проблемы с тестированием

29.10.22

#Multithreading – мультипоточность