МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №2

**«НАСЛЕДОВАНИЕ И ВИРТУАЛЬНЫЕ ФУНКЦИИ»**

Выполнил:

студент 2 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверил:

Козик И. Д.

labrab2.cpp

#include <iostream>

#include "SteamShip.h"

#include "Sailboat.h"

#include "Corvette.h"

using namespace std;

// Variant #15

// корабль, пароход, парусник, корвет.

//

// \_\_\_\_ Ship \_\_\_\_

// / | \

// SteamShip Sailboat Corvette

//

// 1. Определить иерархию классов(в соответствии с вариантом).

// 2. Определить в классе статическую компоненту - указатель на начало связанного

// списка объектов и статическую функцию для просмотра списка.

// 3. Реализовать классы.

// 4. Написать демонстрационную программу, в которой создаются объекты различных

// классов и помещаются в список, после чего список просматривается.

// 5. Сделать соответствующие методы не виртуальными и посмотреть, что будет. (Попытка вызвать базовый, а его нет, компилятор кричит ХАНА)

// 6. Реализовать вариант, когда объект добавляется в список при создании,

// т.е.в конструкторе(смотри пункт 6 следующего раздела).

int main()

{

SteamShip\* st\_ship = new SteamShip();

st\_ship->setShipName("StmS 1");

// st\_ship->addToShipList();

// (new Sailboat())->addToShipList();

// (new Corvette())->addToShipList();

new Sailboat();

new Corvette();

// Ship::logShipList();

Sailboat::logShipList();

}

Ship.h

#pragma once

#include <list>

#include <string>

// (\*) Abstract class

class Ship

{

protected:

static std::list<Ship\*> ship\_list;

std::string ship\_name = "A boat";

public:

// Просмотр (логирование в консоль) списка

static void logShipList();

// Добавить в список корабль

void addToShipList();

// Этот метод будет менятся для потомков

// Чистый виртуальный метод

virtual std::string getClass() = 0;

// Эти методы не будут меняться

std::string getShipName();

void setShipName(std::string ship\_name);

// Конструкторы

Ship();

// Деструктор

virtual ~Ship();

};

Ship.cpp

#include "Ship.h"

#include <iostream>

#include <string>

#include <list>

std::list<Ship\*> Ship::ship\_list;

void Ship::logShipList() {

for (Ship\* ship : Ship::ship\_list) {

std::cout << ship->getClass() << ": " << ship->getShipName() << std::endl;

}

}

void Ship::addToShipList() {

Ship::ship\_list.push\_back(this);

}

std::string Ship::getShipName() {

return this->ship\_name;

}

void Ship::setShipName(std::string ship\_name) {

this->ship\_name = ship\_name;

}

Ship::Ship() {

Ship::ship\_list.push\_back(this);

}

Ship::~Ship() {

delete& (this->ship\_name);

}

Corvette.h

#pragma once

#include "Ship.h"

#include <string>

class Corvette : public Ship

{

std::string getClass();

};

Corvette.cpp

#include "Corvette.h"

#include <string>

std::string Corvette::getClass() {

return "Corvette";

}

Sailboat.h

#pragma once

#include "Ship.h"

#include <string>

class Sailboat : public Ship

{

std::string getClass();

};

Sailboat.cpp

#include "Corvette.h"

#include <string>

std::string Corvette::getClass() {

return "Corvette";

}

Steamship.h

#pragma once

#include "Ship.h"

#include <string>

class Steamship: public Ship

{

std::string getClass();

};

Steamship.cpp

#include "SteamShip.h"

#include <string>

std::string SteamShip::getClass() {

return "SteamShip";

}

Результат: