ФЕДЕРАЛЬНОЕ АГЕНСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ Императора Александра I»

Кафедра «Информационные и вычислительные системы»

Дисциплина «Программирование С++»

**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5**

ВАРИАНТ 19

|  |  |
| --- | --- |
| Выполнил студент  Факультет: АИТ  Группа: ИВБ-211 | Шефнер А. |
| Проверил: | Проузин О.В. |

**Санкт-Петербург**

**2023**

Оценочный лист результатов ЛР № 5

Ф.И.О. студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шефнер Альберт\_\_\_\_\_

Группа \_\_\_\_\_\_\_\_\_\_\_\_\_ИВБ-211\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№**  **п/п** | **Материалы необходимые для оценки знаний, умений**  **и навыков** | **Показатель**  **оценивания** | **Критерии**  **Оценивания** | **Шкала оценивания** | **Оценка** |
| 1 | Лабораторная работа№ | Соответствие методике выполнения | Соответствует | 7 |  |
| Не соответствует | 0 |
| Срок выполнения | Выполнена в срок | 2 |  |
| Выполнена с опозданием на 2 недели | 0 |
| оформление | Соответствует требованиям | 1  0 |  |
| Не соответствует |  |
|  | **ИТОГО количество баллов** |  |  | 10 |  |

Доцент кафедры

«Информационные и вычислительные

системы» Проурзин О.В. «\_\_» \_\_\_\_\_\_\_\_\_\_2023 г.

**Цели работы:**

* Использовать наследование классов.

**Задание**

1. Создать базовый класс согласно индивидуальному заданию.
2. Создать два производных от него класса согласно индивидуальному заданию

Требования к базовому классу:

* члены-данные должны быть общими для обоих производных классов и иметь защищённый спецификатор доступа;
* два конструктора (по умолчанию и с параметрами) и деструктор.
* чисто виртуальная функция просмотра состояния;

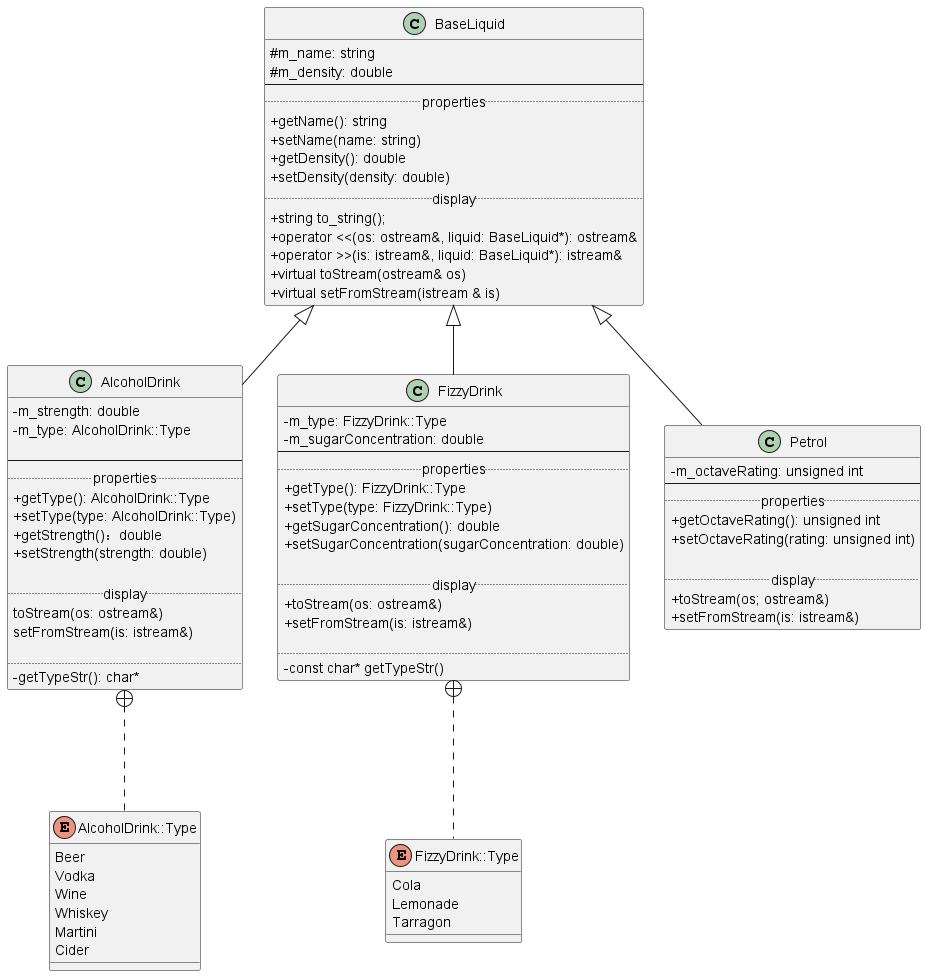
**Вариант 9**

**Создать иерархическую систему классов:**

* базовым классом должен быть абстрактный класс «Жидкость» (СLiquid);
* производными классами должны стать классы «Спирт» (CAlcohol) и «Бензин» (CPetrol).
  + При этом:
* в базовом классе СLiquid должны быть заданы поля для названия и плотности.
* в производном классе CAlcohol должно быть задано поля для хранения крепости (основная характеристика жидкости);
* в производном классе CPetrol должно быть задано поля для хранения октанового числа (основная характеристика жидкости);
* все классы иерархии должны иметь следующие методы: методы получения значения и задания основной характеристики жидкости (два последних метода должны быть чисто виртуальными в базовом классе), а также метод для выведения на экран основных характеристик (данный метод должен быть чисто виртуальным).
* **Используемые средства**

В качестве интегрированной среды разработки использовалась JetBrains CLion.

Для работы в консоли с потоками ввода-вывода использовалась стандартная библиотека <iostream>.

**UML-диаграмма программы:**

**Код программы**

**main.cpp**

#include <iostream>

#include <vector>

#import "liquids/BaseLiquid.h"

#import "liquids/Petrol.h"

#import "liquids/FizzyDrink.h"

#import "liquids/AlcoholDrink.h"

int main() {

std::vector<BaseLiquid\*> liquids({

new Petrol("Kazmunaygaz", 750, 95),

new FizzyDrink("CoolCola", 1030, FizzyDrink::Cola, 10.6),

new FizzyDrink("Buratino", 1030, FizzyDrink::Lemonade, 5.8),

new AlcoholDrink("Eboshi", 1002, AlcoholDrink::Beer, 4.6),

new AlcoholDrink("Vozduh", 970, AlcoholDrink::Vodka, 45),

});

for(auto liquid : liquids) {

std::cout << liquid << "\n";

}

return 0;

}

**BaseLiquid.h**

#ifndef CPP\_LAB\_5\_BASELIQUID\_H

#define CPP\_LAB\_5\_BASELIQUID\_H

#include <string>

#include <utility>

class BaseLiquid {

public:

BaseLiquid() = default;

BaseLiquid(std::string name, double density) : m\_name(std::move(name)), m\_density(density) {};

std::string getName() { return m\_name; }

void setName(std::string name) { m\_name = std::move(name); }

double getDensity() { return m\_density; }

void setDensity(double density) { m\_density = density; }

std::string to\_string();

friend std::ostream& operator <<(std::ostream& os, BaseLiquid\* liquid) { liquid->toStream(os); return os; }

friend std::istream& operator >>(std::istream& is, BaseLiquid\* liquid) { liquid->setFromStream(is); return is; };

virtual void toStream(std::ostream& os) = 0;

virtual void setFromStream(std::istream & is) = 0;

protected:

std::string m\_name;

double m\_density = 0;

};

#endif //CPP\_LAB\_5\_BASELIQUID\_H

**BaseLiquid.cpp**

#include "BaseLiquid.h"

#include <sstream>

std::string BaseLiquid::to\_string() {

std::stringstream ss;

ss << this;

return ss.str();

}

**AlcoholDrink.h**

#include "BaseLiquid.h"

class AlcoholDrink : public BaseLiquid {

public:

enum Type {

Beer, Vodka, Wine, Whiskey, Martini, Cider

};

AlcoholDrink() = default;

AlcoholDrink(std::string name, double density, Type type, double strength);

void toStream(std::ostream& os) override;

void setFromStream(std::istream & is) override;

Type getType() { return m\_type; }

void setType(Type type) { m\_type = type; }

double getStrength() { return m\_strength; }

void setStrength(double strength) { m\_strength = strength; }

private:

double m\_strength = 0;

Type m\_type = Beer;

const char\* getTypeStr();

};

**AlcoholDrink.cpp**

#include "AlcoholDrink.h"

#include <utility>

#include <iostream>

AlcoholDrink::AlcoholDrink(std::string name, double density, Type type, double strength) : BaseLiquid(std::move(name), density) {

m\_type = type;

m\_strength = strength;

}

void AlcoholDrink::toStream(std::ostream &os) {

os << getTypeStr() << " " << m\_name << ". Strength: " << m\_strength;

}

void AlcoholDrink::setFromStream(std::istream &is) {

int enumNumber;

is >> m\_name >> m\_density >> enumNumber >> m\_strength;

}

const char \*AlcoholDrink::getTypeStr() {

switch (this->m\_type) {

case Beer: return "Beer";

case Vodka: return "Vodka";

case Wine: return "Wine";

case Whiskey: return "Whiskey";

case Martini: return "Martini";

case Cider: return "Cider";

default: return "Unknown alcohol drink";

}

}

**FizzyDrink.h**

#include "BaseLiquid.h"

class FizzyDrink : public BaseLiquid{

public:

enum Type {

Cola, Lemonade, Tarragon

};

FizzyDrink() = default;

FizzyDrink(std::string name, double density, Type type, double sugarConcentration);

void toStream(std::ostream& os) override;

void setFromStream(std::istream & is) override;

Type getType() { return m\_type; }

void setType(Type type) { m\_type = type; }

double getSugarConcentration() { return m\_sugarConcentration; }

void setSugarConcentration(double sugarConcentration) { m\_sugarConcentration = sugarConcentration; }

private:

Type m\_type = Cola;

double m\_sugarConcentration = 0;

const char\* getTypeStr();

};

**FizzyDrink.cpp**

#include "FizzyDrink.h"

#include <utility>

#include <iostream>

FizzyDrink::FizzyDrink(std::string name, double density, FizzyDrink::Type type, double sugarConcentration) : BaseLiquid(std::move(name), density) {

m\_type = type;

m\_sugarConcentration = sugarConcentration;

}

void FizzyDrink::toStream(std::ostream &os) {

os << getTypeStr() << " " << m\_name << ". Sugar concentration: " << m\_sugarConcentration;

}

void FizzyDrink::setFromStream(std::istream &is) {

int enumNumber;

is >> m\_name >> m\_density >> enumNumber >> m\_sugarConcentration;

}

const char \*FizzyDrink::getTypeStr() {

switch (this->m\_type) {

case Cola: return "Cola";

case Lemonade: return "Lemonade";

case Tarragon: return "Tarragon";

default: return "Unknown fizzy drink";

}

}

**Petrol.h**

#ifndef CPP\_LAB\_5\_PETROL\_H

#define CPP\_LAB\_5\_PETROL\_H

#include "BaseLiquid.h"

class Petrol : public BaseLiquid {

public:

Petrol() = default;

Petrol(std::string name, double density, unsigned int octaveRating);

unsigned int getOctaveRating() {return m\_octaveRating; }

void setOctaveRating(unsigned int rating) { m\_octaveRating = rating; }

void toStream(std::ostream& os) override;

void setFromStream(std::istream & is) override;

private:

unsigned int m\_octaveRating = 0;

};

#endif //CPP\_LAB\_5\_PETROL\_H

**Petrol.cpp**

#include "Petrol.h"

#include <utility>

#include <iostream>

Petrol::Petrol(std::string name, double density, unsigned int octaveRating) : BaseLiquid(std::move(name), density) {

m\_octaveRating = octaveRating;

}

void Petrol::toStream(std::ostream &os) {

os << "Patrol " << m\_name << ". Octave rating: " << m\_octaveRating;

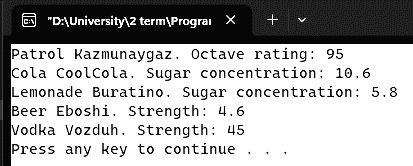
}

void Petrol::setFromStream(std::istream &is) {

is >> m\_name >> m\_density >> m\_octaveRating;

}

**Тестовые примеры**

****