Министерство образования и науки Российской Федерации  
Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский  
политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.1**

Дисциплина: «Информатика»

Тема: Объектно-ориентированное программирование. Инкапсуляция

Вариант 13

Выполнил:

Студент группы РИС-20-1б

Кузнецов Михаил Сергеевиич

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

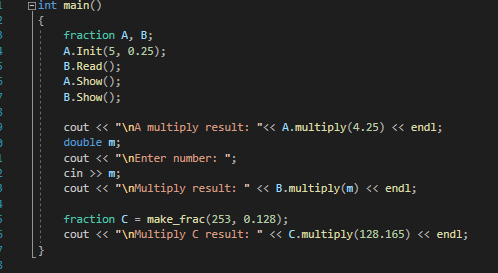
**Постановка задачи**

1. Реализовать определение нового класса. Продемонстрировать разные способы создания объектов и массивов объектов.

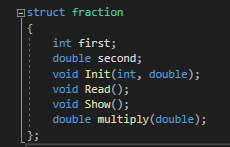
2. Структура-пара – структура с двумя полями, которые обычно имеют имена first и second.

**Анализ задачи**

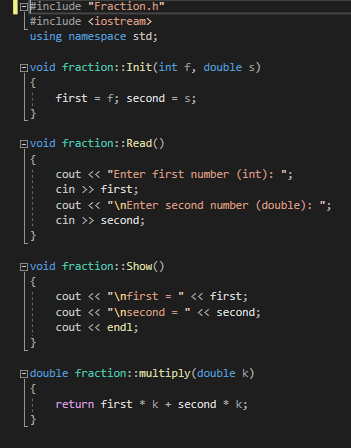
1. Для решения задачи необходимо…
   1. Написать функцию int main, в которой будет производиться считка введенного времени в часах и в минутах с клавиатуры, а затем передача этих чисел в функцию;



* 1. В заголовочном файле описать структуру и методы;



* 1. В файле fraction.cpp определить методы.



1. В программе были использованы следующие типы данных:
   1. Тип данных int для хранения первого числа;



* 1. Тип данных double для хранения второго числа;



**Код**

#include <iostream>

#include "Fraction.h"

using namespace std;

fraction make\_frac(int f, double s)

{

fraction t;

t.Init(f, s);

return t;

}

int main()

{

fraction A, B;

A.Init(5, 0.25);

B.Read();

A.Show();

B.Show();

cout << "\nA multiply result: "<< A.multiply(4.25) << endl;

double m;

cout << "\nEnter number: ";

cin >> m;

cout << "\nMultiply result: " << B.multiply(m) << endl;

fraction C = make\_frac(253, 0.128);

cout << "\nMultiply C result: " << C.multiply(128.165) << endl;

}

#include "Fraction.h"

#include <iostream>

using namespace std;

void fraction::Init(int f, double s)

{

first = f; second = s;

}

void fraction::Read()

{

cout << "Enter first number (int): ";

cin >> first;

cout << "\nEnter second number (double): ";

cin >> second;

}

void fraction::Show()

{

cout << "\nfirst = " << first;

cout << "\nsecond = " << second;

cout << endl;

}

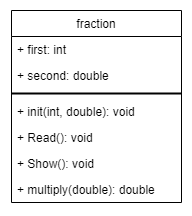
double fraction::multiply(double k)

{

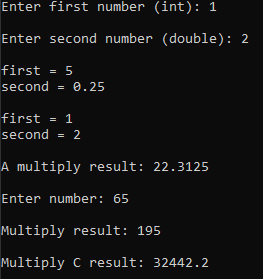
return first \* k + second \* k;

}

**UML диаграмма**



**Скриншоты тестов**

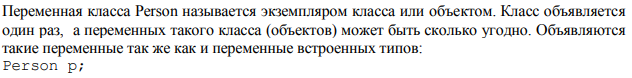
****

**Контрольные вопросы**



1. 



2.



3. Полями/атрибутами называются данные класса.



4. Функции класса называются методами.



5. Спецификаторы доступа управляют видимостью элементов класса.



6. Спецификатор public используется, чтобы задать общедоступную видимость.



7. Спецификатор private используется, чтобы задать видимость только внутри класса для полей или методов под этим спецификатором.



8. Для спецификатора class по умолчанию используется private.



9. Для спецификатора struct по умолчанию используется public.



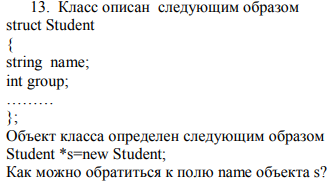
10. Интерфейс класса описывается после спецификатора public, чтобы к методам класса можно было обратится из других функций.



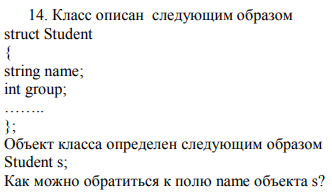
11. Значения атрибутов экземпляра класса можно изменить с помощью методов класса: модификаторов.



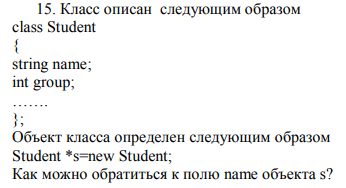
12. Значения атрибутов экземпляра класса можно получить с помощью методов класса: селекторов.



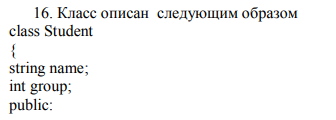
13. Через «->» можно обратиться к полю name, так как оно имеет открытый доступ public.

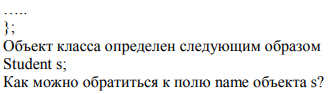


14. Через «.» можно обратиться к полю name, так как оно имеет открытый доступ public.

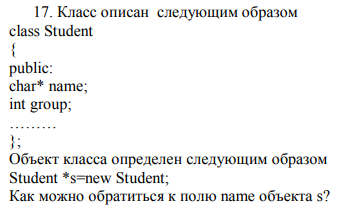


15. Необходимо прописать модификатор для класса Student.





16. Необходимо прописать модификатор для класса Student.



17. 16. Необходимо прописать селектор и модификатор для класса Student.