Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

ПНИПУ

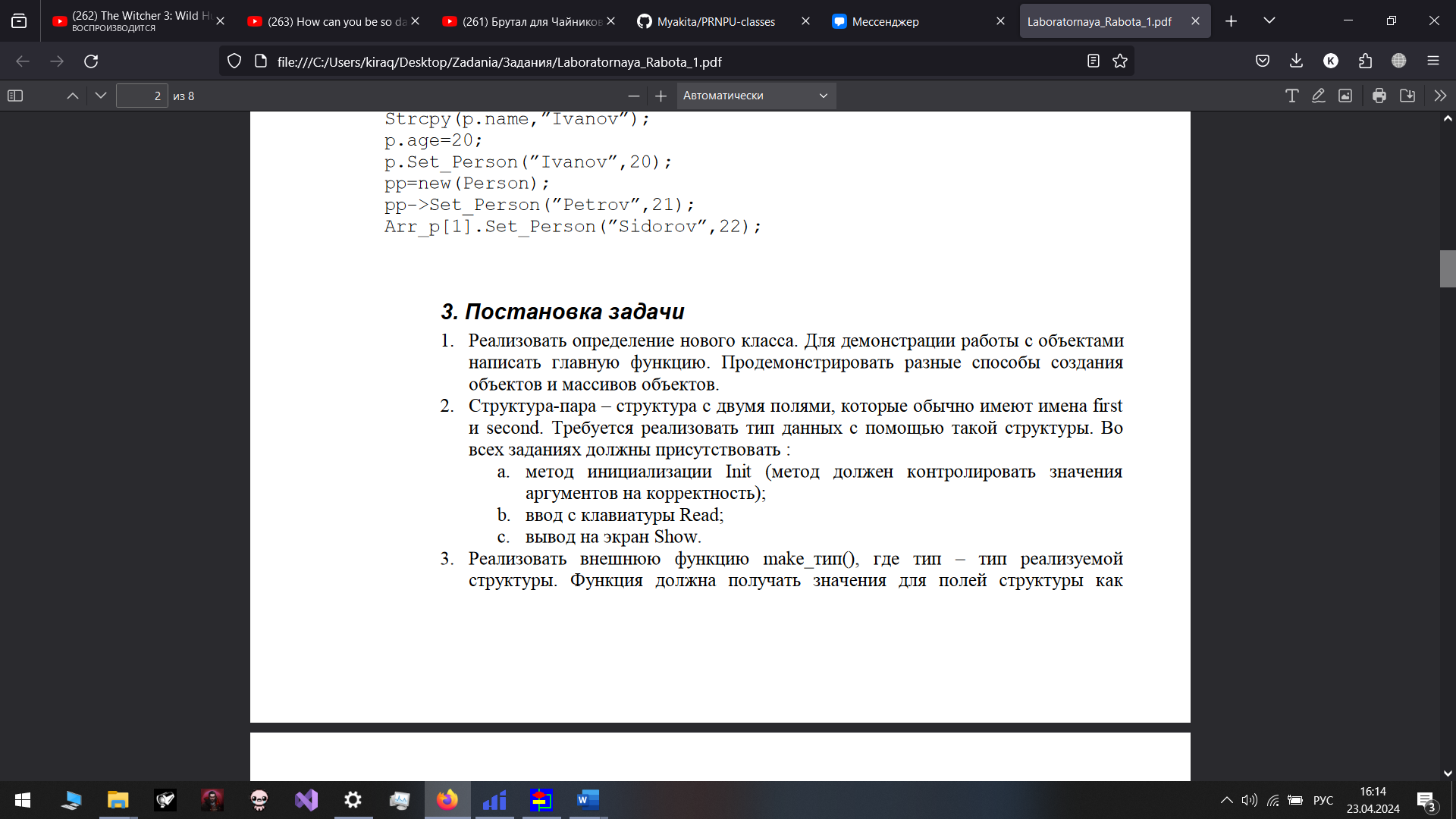
Лабораторная работа  
«Инкапсуляция»

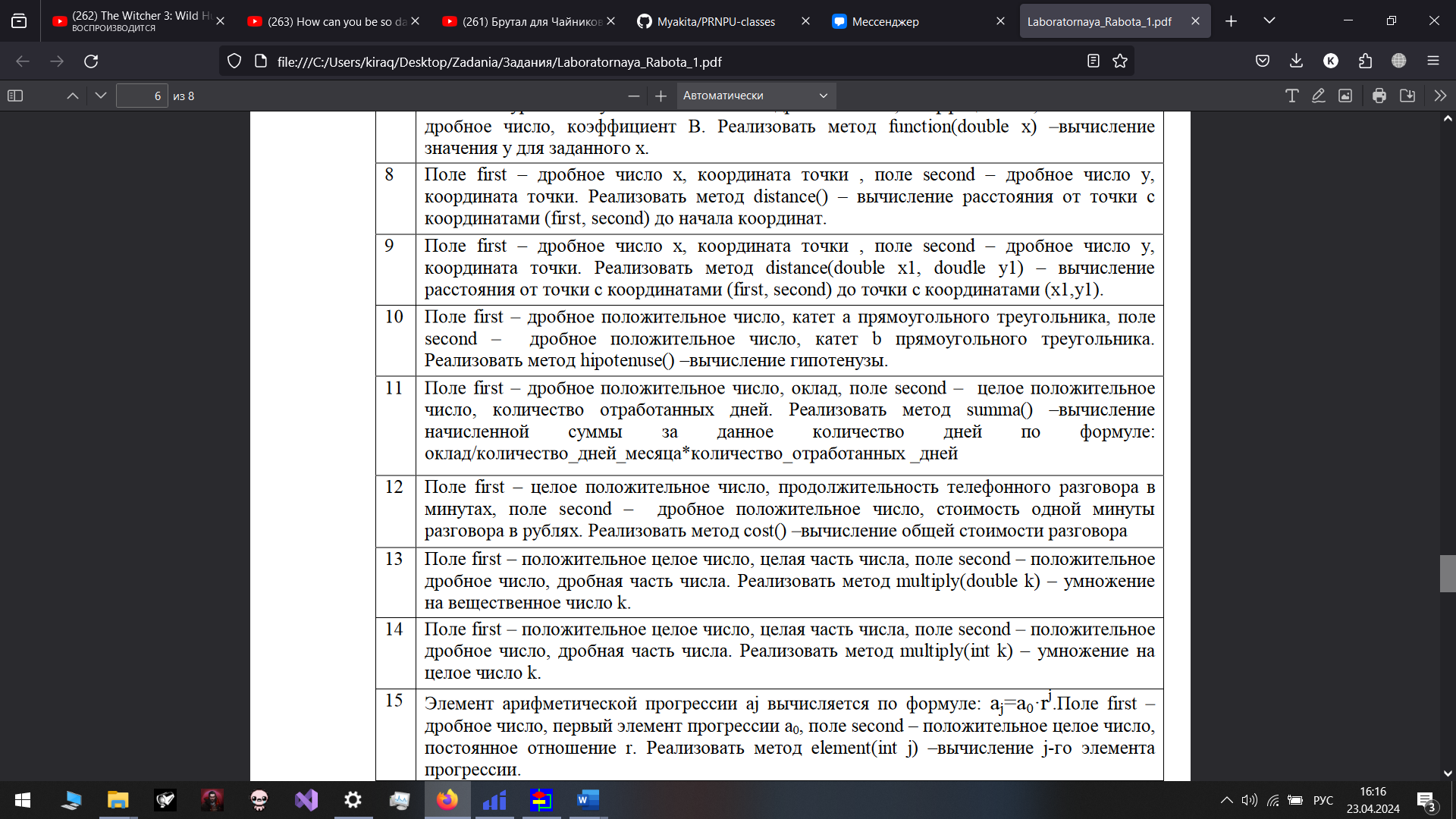
Выполнил:   
студент группы РИС-23-1б   
Мокрушин Никита Дмитриевич

Проверила:   
доцент кафедры ИТАС   
О.А. Полякова

Пермь 2024 г.

Постановка задачи:





Анализ задачи:

* Создаем класс Sum – пара чисел с разными типами данных
* Создаём 3 конструктора – по умолчанию, с параметрами, без параметров. При работе каждого конструктора выводим уникальную надпись.
* Разрабатываем деструктор и при его использовании выводим надпись.
* Перегружаем операции: +, -, =, ++ и – (префиксные и постфиксные).
* Разрабатываем операцию по добавлению константы к числу.

Код:

#pragma once

class Sum{

public:

void summa(double salary, int days\_of\_work);

void show();

private:

double salary;

int days\_of\_work;

#include "Sum.h"

#pragma once

#include <iostream>

using namespace std;

void Sum::summa(double first, int second)

{

salary = first;

days\_of\_work = second;

}

void Sum::show()

{

cout << salary\*days\_of\_work/31 << endl;

}

#include <iostream>

#include "Sum.h"

using namespace std;

int main()

{

setlocale(0, "");

system("chcp 1251");

system("cls");

double salary;

int days\_of\_work;

Sum sum;

cout << "Введите оклад работника: ";

cin >> salary;

cout << "Введите кол-во отработанных дней: ";

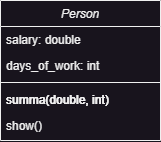
cin >> days\_of\_work;

sum.summa(salary, days\_of\_work);

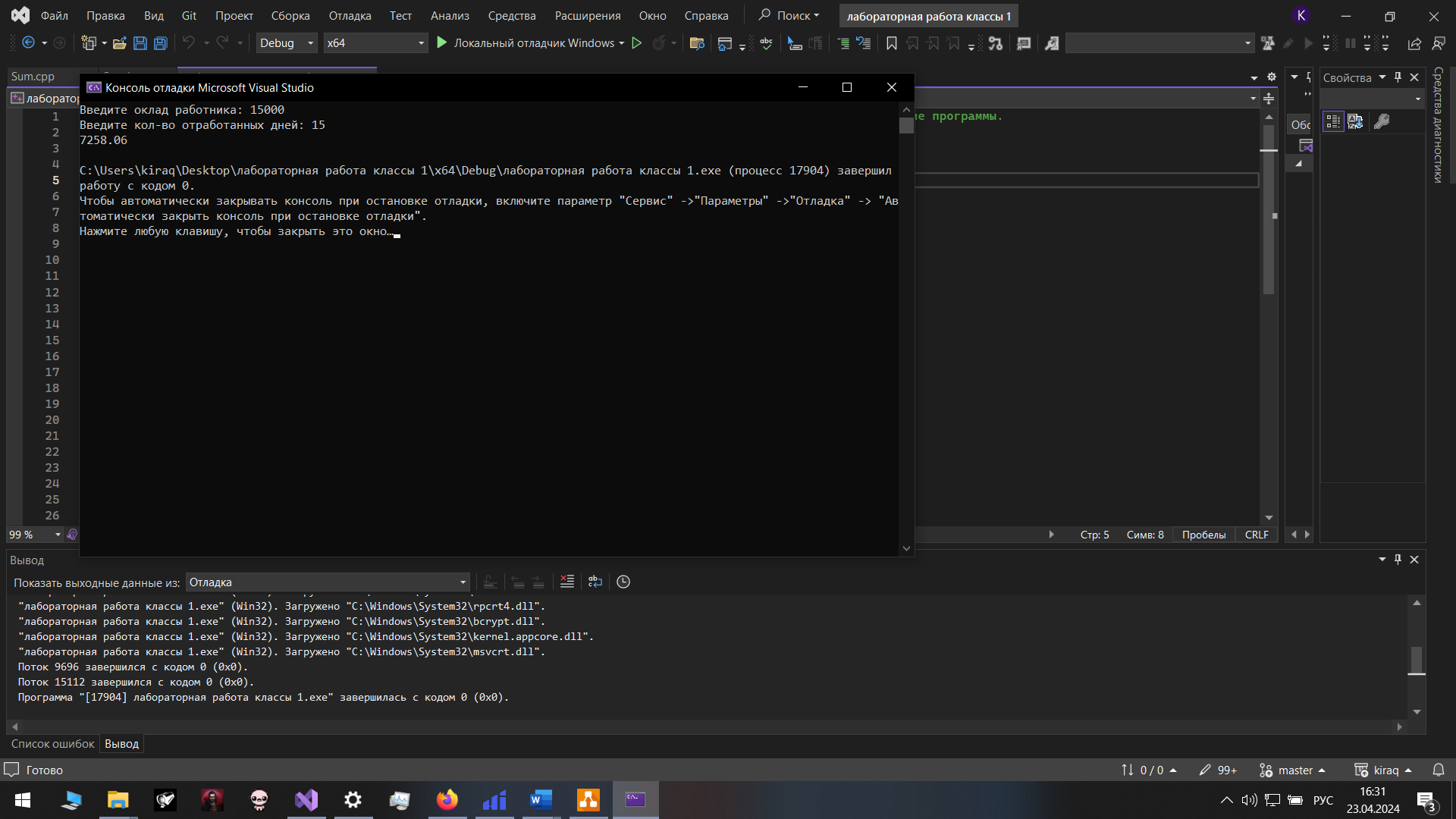
sum.show();

}

UML Диаграммы:



Вывод:



программа работает корректно и выдаёт требуемый результат.  
Ответы на вопросы:

1. Класс - это шаблон или формальное описание для создания объектов. Он определяет состояние (поля) и поведение (методы) объектов.
2. Объект (или экземпляр) класса - это конкретный экземпляр класса, созданный на основе его определения. Он содержит реальные данные, соответствующие полям класса, и может вызывать методы класса для выполнения определенных действий.
3. Поля класса - это переменные, которые определяют состояние объектов данного класса. Они содержат данные, которые характеризуют объекты.
4. Функции класса, также называемые методами, определяют поведение объектов данного класса. Они могут выполнять операции над данными класса или предоставлять доступ к ним.
5. Спецификаторы доступа определяют уровень доступа к членам класса (полям и методам) извне класса.
6. Спецификатор public используется для обозначения членов класса, к которым можно обращаться извне класса. Они являются общедоступными.
7. Спецификатор private используется для обозначения членов класса, к которым нельзя обращаться извне класса. Они доступны только внутри класса.
8. Если описание класса начинается со спецификатора class, то по умолчанию используется спецификатор доступа private.
9. Если описание класса начинается со спецификатора struct, то по умолчанию используется спецификатор доступа public.
10. При описании интерфейса класса рекомендуется использовать спецификатор доступа public, чтобы предоставить доступ к методам класса извне для использования.
11. Значения атрибутов экземпляра класса можно изменить, вызывая методы класса, предназначенные для этого, либо напрямую доступая к полям класса, если они имеют доступный уровень доступа (например, если они объявлены как public).
12. Значения атрибутов экземпляра класса можно получить, обратившись к полям класса, если они имеют доступный уровень доступа, или вызвав методы, которые возвращают эти значения.
13. Для обращения к полю name объекта s, определенного как указатель на объект класса Student, можно использовать оператор разыменования (->), например: s->name.
14. Для обращения к полю name объекта s, определенного как объект класса Student, можно использовать оператор доступа к члену (.), например: s.name.
15. Для обращения к полю name объекта s, определенного как указатель на объект класса Student, также можно использовать оператор разыменования (->), например: s->name.
16. Если поле name объявлено как public, как в вашем примере, то можно обратиться к нему напрямую через оператор доступа к члену (.), например: s.name.
17. Если поле name объявлено как указатель на char, как в вашем примере, и объект s определен как указатель на объект класса Student, то можно обратиться к нему так же, как и в случае с объектом, который является указателем: s->name.