МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе №5**

**«**Наследование. Полиморфизм.**»**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБс-222», «АВТФ» *доцент кафедры ЗИ*

*Азява Д.А. Архипова А. Б.*

«\_\_\_» \_\_\_\_\_\_ 2023 г.«\_\_\_» \_\_\_\_\_\_ 2023 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2023

Задание к работе: Написать программу с использованием принципов ООП и паттернов программирования.

**Вариант 14 (вес 1)**

Паттерн!

Разработать комплекс по выставлению и учету оценок. Преподаватель может давать доклады, выставлять задания, регистрировать студентов, создавать предметы, ставить оценки студентам по предмету за доклады и лабы, добавлять других преподавателей. Студенты могут регистрироваться на доклады, производить самозапись что они что-то сдали (просто напоминалка преподавателю), после чего преподаватель "утверждает" сдачу, и она выставляется в журнал. или отвергает ее. Студент может спрыгнуть с доклада, тогда он освобождается для записи. Разработать также приложение для выгрузки итогов по предметам.

Для выполнения поставленной задачи используем паттерн «Строитель», где будут отдельные “Строители” для создания доклада и задания.

Таблица 1 - Листинг программы

|  |  |
| --- | --- |
| Файл | Код |
| Task-14.cpp | #include <iostream>  #include <vector>  #include <string>  using namespace std;  // Класс Доклад  class Presentation {  public:  string title;  string speaker;  Presentation(const string& title, const string& speaker) : title(title), speaker(speaker) {}  };  // Класс Задание  class Assignment {  public:  string name;  string description;  Assignment(const string& name, const string& description) : name(name), description(description) {}  };  // Класс Студент  class Student {  public:  string name;  vector<Presentation> presentations;  vector<Assignment> assignments;  Student(const string& name) : name(name) {}  void addPresentation(const Presentation& presentation)  {  presentations.push\_back(presentation);  }  void addAssignment(const Assignment& assignment)  {  assignments.push\_back(assignment);  }  };  // Класс Предмет  class Subject {  public:  string name;  vector<Student> students;  Subject(const string& name) : name(name) {}  void addStudent(const Student& student)  {  students.push\_back(student);  }  };  // Builder для создания Доклада  class PresentationBuilder {  private:  string title;  string speaker;  public:  PresentationBuilder& setTitle(const string& title)  {  this->title = title;  return \*this;  }  PresentationBuilder& setSpeaker(const string& speaker)  {  this->speaker = speaker;  return \*this;  }  Presentation build() {  return Presentation(title, speaker);  }  };  // Builder для создания Задания  class AssignmentBuilder {  private:  string name;  string description;  public:  AssignmentBuilder& setName(const string& name)  {  this->name = name;  return \*this;  }  AssignmentBuilder& setDescription(const string& description)  {  this->description = description;  return \*this;  }  Assignment build()  {  return Assignment(name, description);  }  };  int main() {  // Создание урока  Subject subject("Subject 1");  // Создание студента  Student student("Student 1");  // Создание доклада с использованием Builder  Presentation presentation = PresentationBuilder()  .setTitle("Raport 1")  .setSpeaker("Student 1")  .build();  // Создание задания с использованием Builder  Assignment assignment = AssignmentBuilder().setName("Task 1").setDescription("Description of task 1").build();  // Добавление доклада и задания студенту  student.addPresentation(presentation);  student.addAssignment(assignment);  // Добавление студента в предмет  subject.addStudent(student);  // Вывод информации о студенте  for (const auto& s : subject.students)  {  cout << "Student: " << s.name << endl;  cout << "Raports:" << endl;  for (const auto& p : s.presentations)  {  cout << "- " << p.title << " (Speaker: " << p.speaker << ")" << endl;  }  cout << "Tasks:" << endl;  for (const auto& a : s.assignments)  {  cout << "- " << a.name << " (Description: " << a.description << ")" << endl;  }  }  return 0;  } |