Disynergy
Страница материала ↗

EXQT
Готовые ответы — на EXQT
Доступ к ответам приобретается отдельно.
Найти ответы ↗

Методические рекомендации

1. Типы данных, операторы, ветвления и циклы

C# — строго типизированный язык программирования, в котором каждый объект и переменная имеет определённый тип данных.

1.1. Основные типы данных

Категория	Примеры	Описание
Целые числа	`int`, `long`, `short`, `byte`	Для хранения целых чисел
Вещественные	`float`, `double`, `decimal`	Для хранения чисел с плавающей точкой
Логические	`bool`	Принимает значения `true` или `false`
Символьные	`char`	Хранит один символ Unicode
Строковые	`string`	Последовательность символов
Объектные	`object`	Базовый тип для всех объектов

```
int age = 20;
double salary = 55000.50;
bool isStudent = true;
string name = "Алексей";
```

1.2. Операторы

- **Арифметические:** +, -, *, /, %
- **Сравнения:** ==, !=, <, >, <=, >=
- **Логические:** &&, ||, !
- **Присваивания:** =, +=, -=, *=, /=
- **Тернарный оператор:** `условие ? значение1 : значение2`

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

1.3. Ветвления

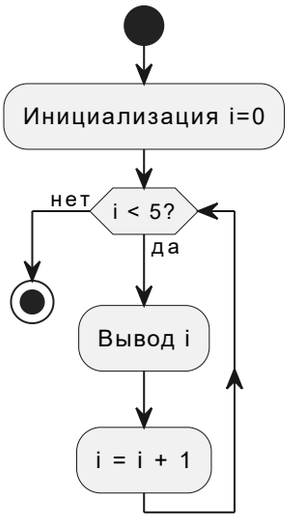
```
int temperature = 10;

if (temperature > 25)
    Console.WriteLine("Жарко");
else if (temperature > 10)
    Console.WriteLine("Тепло");
else
    Console.WriteLine("Холодно");
```

1.4. Циклы

```
// for
for (int i = 0; i < 5; i++)
    Console.WriteLine(i);

// while
int n = 0;
while (n < 3)
{
    Console.WriteLine("Повтор " + n);
    n++;
}

// foreach
string[] names = { "Иван", "Анна", "Пётр" };
foreach (string n in names)
    Console.WriteLine(n);
```

PlantUML — блок-схема цикла

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru



2. Принципы объектно-ориентированного программирования (ООП)

ООП основано на идее моделирования реальных сущностей с помощью классов и объектов.

2.1. Класс и объект

```
class Student
{
    public string Name;
    public int Age;

    public void SayHello()
    {
        Console.WriteLine($"Привет! Меня зовут {Name}, мне {Age} лет.");
    }
}

Student s = new Student();
s.Name = "Анна";
s.Age = 19;
s.SayHello();
```

2.2. Инкапсуляция

Сокрытие данных внутри класса с помощью `private` и открытый доступ через `public` методы.

```
class Account
{
    private double balance;

    public void Deposit(double amount)
    {
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
        balance += amount;
    }

    public double GetBalance() => balance;
}
```

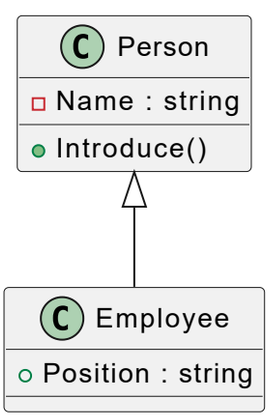
2.3. Наследование

```
class Person
{
    public string Name;
    public void Introduce() => Console.WriteLine($"Я — {Name}");
}

class Employee : Person
{
    public string Position;
}
```

2.4. Полиморфизм

```
class Animal { public virtual void Speak() => Console.WriteLine("Животное говори
class Dog : Animal { public override void Speak() => Console.WriteLine("Гав!");

Animal a = new Dog();
a.Speak(); // Гав!
```

PlantUML — диаграмма классов



3. Работа с коллекциями и обработка исключений

3.1. Коллекции

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
using System.Collections.Generic;

List<string> names = new() { "Иван", "Анна", "Пётр" };
names.Add("Мария");
foreach (var n in names) Console.WriteLine(n);

Dictionary<string, int> grades = new()
{
    ["Математика"] = 5,
    ["Информатика"] = 4
};
Console.WriteLine(grades["Информатика"]);
```

3.2. Обработка исключений

```
try
{
    int a = 10, b = 0;
    Console.WriteLine(a / b);
}
catch (DivideByZeroException ex)
{
    Console.WriteLine("Ошибка деления: " + ex.Message);
}
finally
{
    Console.WriteLine("Программа завершена.");
}
```

4. Организация структуры программного проекта

4.1. Структура проекта

```
MyApp/
├── Program.cs
├── Models/
│   └── Student.cs
├── Modules/
│   └── Report.cs
```

4.2. Пространства имён (namespace)

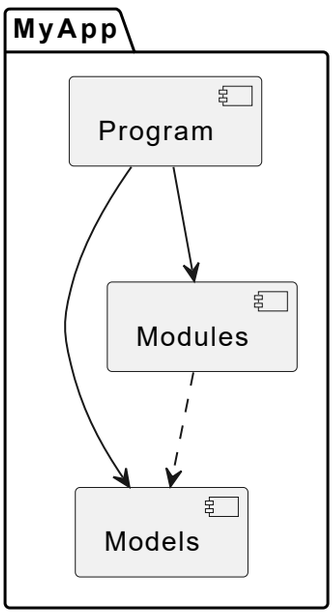
Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
namespace MyApp.Models
{
    public class Student
    {
        public string Name { get; set; }
    }
}
```

4.3. Модульность

Разделение проекта на независимые логические блоки: классы, модули, пространства имён.

PlantUML — диаграмма компонентов



5. Основы работы с файлами и форматами данных

5.1. Работа с файлами

```
File.WriteAllText("data.txt", "Привет, C#!");
string text = File.ReadAllText("data.txt");
Console.WriteLine(text);
```

5.2. Работа с JSON

```
using System.Text.Json;

var person = new { Name = "Анна", Age = 20 };
string json = JsonSerializer.Serialize(person);
File.WriteAllText("person.json", json);
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

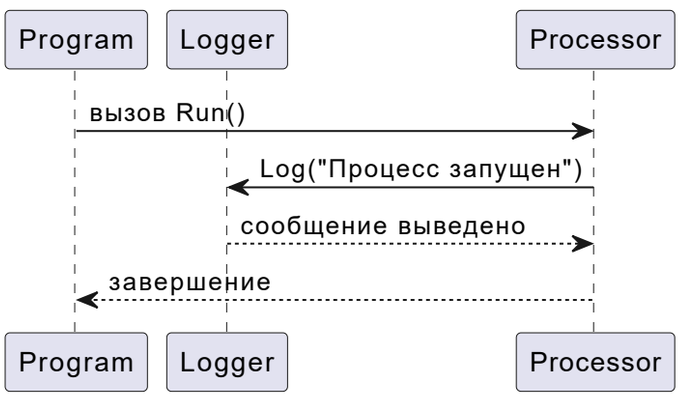
```
var read = JsonSerializer.Deserialize<Dictionary<string, object>>(File.ReadAllTe
Console.WriteLine(read["Name"]);
```

6. Принципы интеграции и взаимодействия модулей

Интеграция — это процесс соединения независимых частей программы в единую систему.

```
class Logger
{
    public void Log(string message) => Console.WriteLine("[LOG] " + message);
}

class Processor
{
    private Logger logger = new();

    public void Run()
    {
        logger.Log("Процесс запущен");
    }
}
```

PlantUML — взаимодействие модулей



7. Тестирование и отладка программ

7.1. Пошаговое выполнение

В Visual Studio можно использовать точки останова (breakpoints) и выполнять код по шагам (`F10`/`F11`).

7.2. Логические проверки

```
Debug.Assert(value > 0, "Значение должно быть положительным");
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

7.3. Пример простого теста

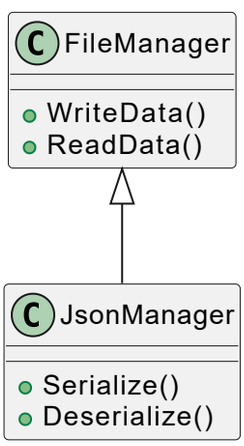
```
void TestSum()
{
    int a = 2, b = 3;
    int result = a + b;
    if (result == 5)
        Console.WriteLine("Тест пройден");
    else
        Console.WriteLine("Ошибка!");
}
```

8. UML-диаграммы и архитектурное моделирование

UML (Unified Modeling Language) используется для описания структуры и поведения программных систем.

Диаграмма	Назначение
Классов	структура классов и их связи
Компонентов	составные части системы
Последовательности	порядок вызовов и взаимодействий
Деятельности	логика выполнения действий

PlantUML — пример диаграммы классов приложения



Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Методические рекомендации

2. Выполнение задания — интеграция модулей, архитектура и тестирование

Описание задания

1. Разработать многомодульное программное приложение на языке C#.
2. Спроектировать архитектуру программной системы с разделением функций по модулям.
3. Организовать взаимодействие между модулями приложения с использованием методов, классов и интерфейсов.
4. Реализовать обмен данными между компонентами, включая передачу объектов и работу с файлами.
5. Создать не менее трёх логических модулей, таких как:
    - модуль ввода данных;
    - модуль хранения информации (в формате JSON);
    - модуль уведомлений или отчётности.
6. Обеспечить корректное взаимодействие модулей и их согласованную работу.
7. Провести отладку и тестирование интегрированной системы для выявления ошибок и проверки функционала.

Подготовить документацию и отчёт по выполненному проекту, включающий описание структуры программы, UML-диаграммы и результаты тестирования.

0. Предварительные требования

- Установлен **.NET SDK 6+** (`dotnet --version`).
- Редактор: **Visual Studio 2022** / **VS Code**.
- Базовые знания: C#, ООП, JSON, UML.
- Git (рекомендовано): `git init`, первый коммит после шага 1.

1. Создать проект и базовую структуру

1.1. Инициализация проекта

```
dotnet new console -n IntegrationApp
cd IntegrationApp
mkdir Models Modules Docs
```

1.2. Структура папок (целевой вид)

```
IntegrationApp/
├── Program.cs
├── IntegrationApp.csproj
├── Models/
│   └── Domain.cs
├── Modules/
│   ├── InputModule.cs
```

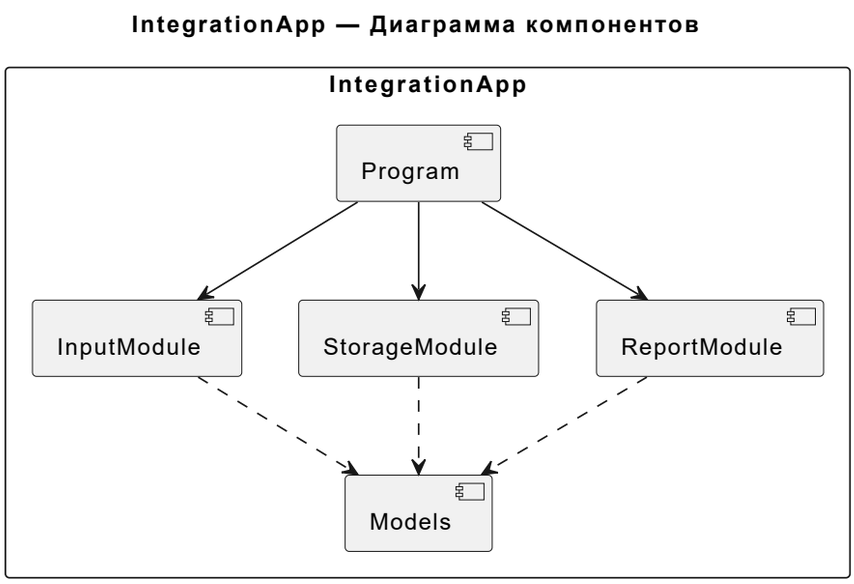
Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
│   ├── StorageModule.cs
│   └── ReportModule.cs
└── Docs/
    ├── components.puml
    └── classes.puml
```

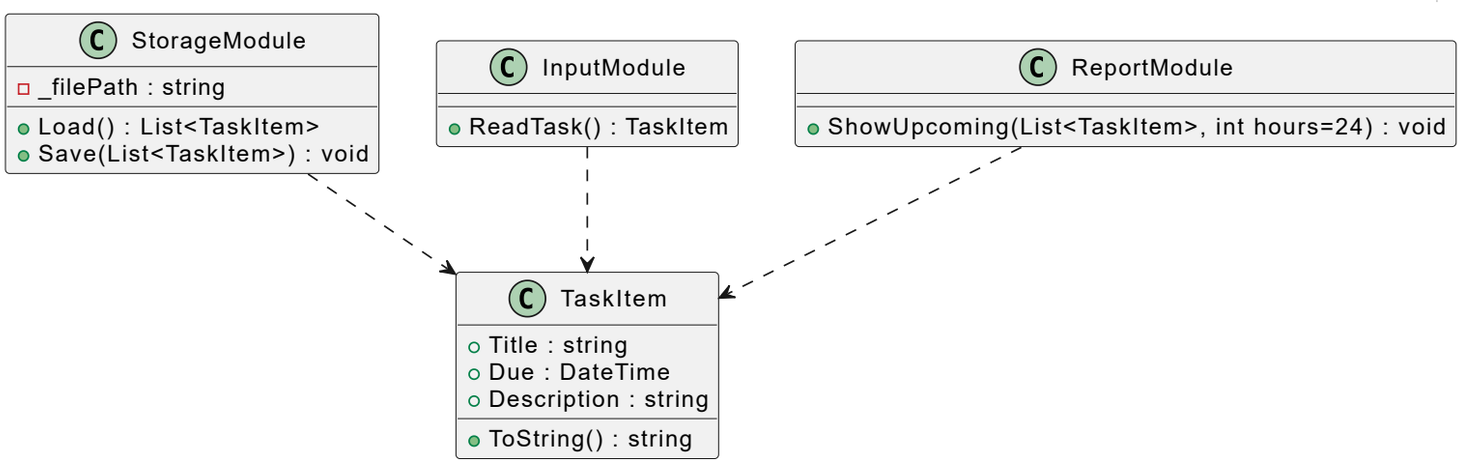
Контрольная точка: проект компилируется `dotnet build`.

2. Спроектировать архитектуру (UML → код)

2.1. Диаграмма компонентов (PlantUML)



2.2. Диаграмма классов (PlantUML)



Контрольная точка: схемы согласованы с ТЗ: есть минимум три логических модуля (ввод, хранение JSON, отчёт/уведомления).

3. Описать доменную модель (Models)

`Models/Domain.cs`

```
namespace IntegrationApp.Models;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
public class TaskItem
{
    public string Title { get; set; } = string.Empty;
    public DateTime Due { get; set; }
    public string Description { get; set; } = string.Empty;

    public override string ToString() => $"{Title} — {Due:g} — {Description}";
}
```

Контрольная точка: класс компилируется, отражает сущность предметной области.

4. Реализовать модуль хранения JSON (StorageModule)

`Modules/StorageModule.cs`

```
using System.Text.Json;
using IntegrationApp.Models;

namespace IntegrationApp.Modules;

public class StorageModule
{
    private readonly string _filePath;
    private readonly JsonSerializerOptions _opts = new() { WriteIndented = true };

    public StorageModule(string filePath = "tasks.json")
        => _filePath = filePath;

    public List<TaskItem> Load()
    {
        if (!File.Exists(_filePath)) return new List<TaskItem>();
        var json = File.ReadAllText(_filePath);
        return JsonSerializer.Deserialize<List<TaskItem>>(json) ?? new List<TaskItem>();
    }

    public void Save(List<TaskItem> tasks)
    {
        var json = JsonSerializer.Serialize(tasks, _opts);
        File.WriteAllText(_filePath, json);
    }
}
```

Контрольная точка: можно вручную создать `tasks.json`, проверить чтение/запись.

5. Реализовать модуль ввода с валидацией (InputModule)

`Modules/InputModule.cs`

```
using IntegrationApp.Models;

namespace IntegrationApp.Modules;
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

```
public static class InputModule
{
    public static TaskItem ReadTask()
    {
        Console.Write("Название: ");
        var title = (Console.ReadLine() ?? "").Trim();

        DateTime due;
        while (true)
        {
            Console.Write("Срок (yyyy-MM-dd HH:mm): ");
            if (DateTime.TryParse(Console.ReadLine(), out due)) break;
            Console.WriteLine("Некорректная дата/время. Повторите.");
        }

        Console.Write("Описание: ");
        var desc = (Console.ReadLine() ?? "").Trim();

        return new TaskItem { Title = title, Due = due, Description = desc };
    }
}
```

Контрольная точка: ввести задачу, получить корректный объект `TaskItem`.

6. Реализовать модуль отчётов/уведомлений (ReportModule)

`Modules/ReportModule.cs`

```
using IntegrationApp.Models;

namespace IntegrationApp.Modules;

public static class ReportModule
{
    public static void ShowUpcoming(List<TaskItem> tasks, int hours = 24)
    {
        var now = DateTime.Now;
        var upcoming = tasks
            .Where(t => t.Due >= now && t.Due <= now.AddHours(hours))
            .OrderBy(t => t.Due)
            .ToList();

        if (!upcoming.Any())
        {
            Console.WriteLine($"Нет задач в ближайшие {hours} часов.");
            return;
        }

        Console.WriteLine($"Ближайшие задачи (<= {hours} ч):");
        foreach (var t in upcoming) Console.WriteLine("- " + t);
    }
}
```

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

Контрольная точка: при наличии задач в интервале виден корректный список.

7. Собрать взаимодействие в Program.cs (интеграция модулей)

`Program.cs`

```
using IntegrationApp.Models;
using IntegrationApp.Modules;

Console.OutputEncoding = System.Text.Encoding.UTF8;

var storage = new StorageModule();
var tasks = storage.Load();

bool exit = false;
while (!exit)
{
    Console.WriteLine("\n=== IntegrationApp ===");
    Console.WriteLine("1. Добавить задачу");
    Console.WriteLine("2. Показать ближайшие задачи (24 ч)");
    Console.WriteLine("3. Сохранить и выйти");
    Console.Write("Выбор: ");
    var choice = Console.ReadLine();

    try
    {
        switch (choice)
        {
            case "1":
                var t = InputModule.ReadTask();
                tasks.Add(t);
                Console.WriteLine("Задача добавлена.");
                break;
            case "2":
                ReportModule.ShowUpcoming(tasks, 24);
                break;
            case "3":
                storage.Save(tasks);
                Console.WriteLine("Сохранено. Выход...");
                exit = true;
                break;
            default:
                Console.WriteLine("Неизвестная команда.");
                break;
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Ошибка: " + ex.Message);
    }
}
```

Контрольная точка: приложение запускается `dotnet run`, добавляет задачи, сохраняет `tasks.json`.

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

8. Добавить обработку ошибок и логические проверки

- В `StorageModule.Save/Load` уже есть базовая защита.
- При вводе дат — повторный запрос.
- Добавьте дополнительные проверки: пустое название, дата в прошлом и т. п.

```
if (string.IsNullOrWhiteSpace(title)) throw new ArgumentException("Название не может быть пустым
if (due < DateTime.Now) Console.WriteLine("Внимание: срок в прошлом.");
```

Контрольная точка: негативные сценарии не приводят к аварийному завершению программы.

9. Простые интеграционные тесты (без фреймворка)

Создайте вспомогательный метод для автопроверки связок «ввод → хранение → отчёт»:

```
static void IntegrationSmokeTest()
{
    var storage = new StorageModule("test.json");
    var list = new List<TaskItem>
    {
        new() { Title = "Тест", Due = DateTime.Now.AddHours(1), Description = "Smoke" }
    };
    storage.Save(list);
    var read = storage.Load();
    ReportModule.ShowUpcoming(read, 2);
    File.Delete("test.json");
}
// Запуск один раз вручную:
// IntegrationSmokeTest();
```

Контрольная точка: тест подтверждает корректную связку модулей. При желании используйте MSTest/xUnit.

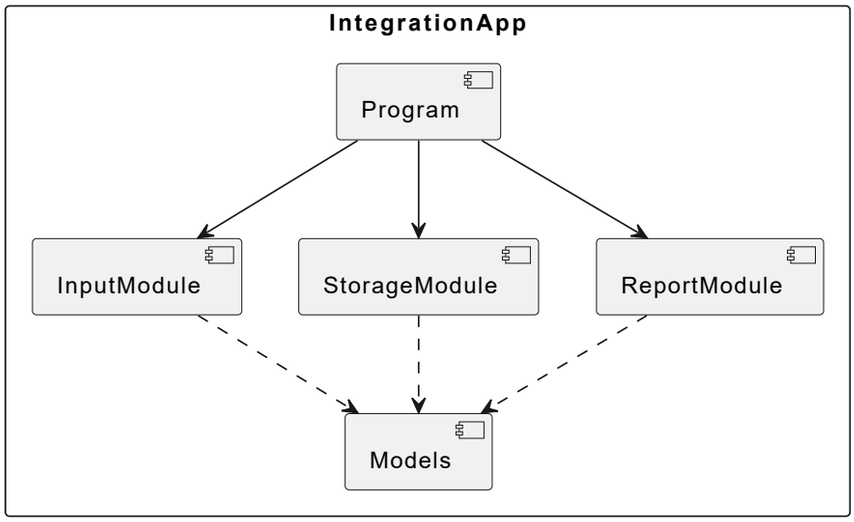
Документация и отчёт

Подготовьте документацию: описание структуры программы, UML-диаграммы (из раздела 2) и результаты тестирования со скриншотами.

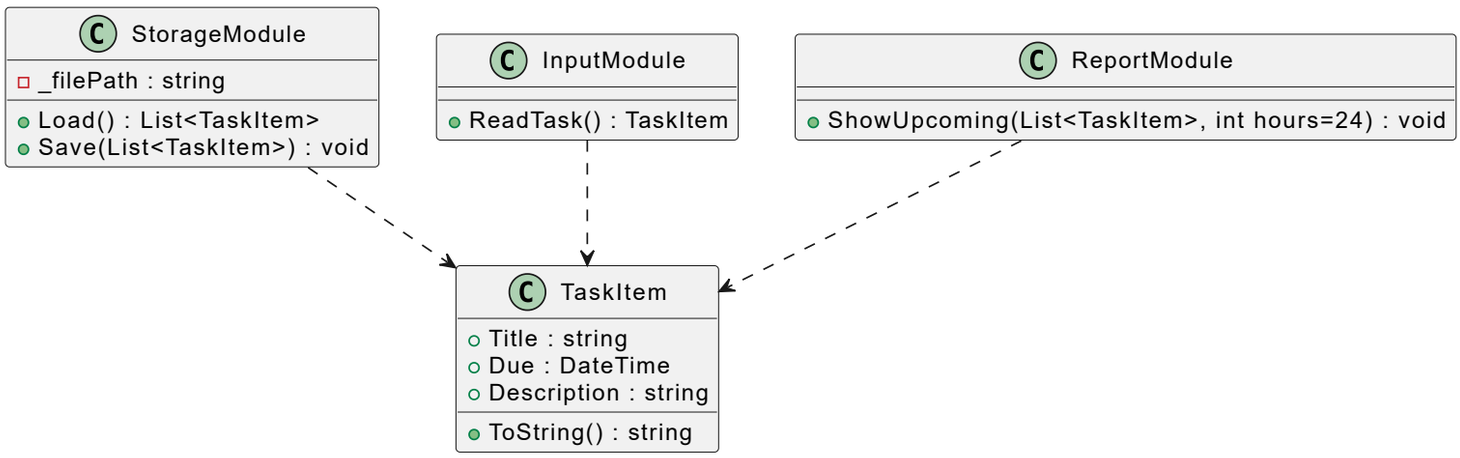
PlantUML — Диаграмма компонентов

Помогу с учебным заданием без предоплаты. Оплата после выполнения
Нажмите на мессенджер, чтобы написать
Telegram ↗ WhatsApp ↗ MAX ↗
+7 (924) 305-23-08 · Пишите в мессенджер
disynergy.ru

IntegrationApp — Диаграмма компонентов



PlantUML — Диаграмма классов



Пример готового приложения: GitHub — PP.PM.02-testProject - https://github.com/OlgaKraven/PP.PM.02-testProject.git

Помогу с учебным заданием без предоплаты. Оплата после выполнения

Нажмите на мессенджер, чтобы написать

Telegram ↗ WhatsApp ↗ MAX ↗

+7 (924) 305-23-08 · Пишите в мессенджер

disynergy.ru