Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»
Рязанский станкостроительный колледж

Отчёт о практической работе №5
Рефакторинг программного кода
МДК
«02. 02»

Выполнила:
Студент группы ИСП-22
Шерстнёва Ю.С.
Проверил:
Родин Е.Н.

Рязань 2025

Основная часть

Цель работы: Создать новую ветку для реализации интерактивного режима консольного приложения, провести рефакторинг кода для улучшения структуры, слить изменения в основную ветку, получив полностью рабочую версию программы.

Ход выполнения работы:

Задачи:

1. Переключение на основную ветку и создание новой ветки:

Переключитесь на основную ветку: git checkout master

Синхронизируйте локальную master с удаленной: git pull origin master

```
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git checkout master
M       PracticalLesson1/Managers/TodoListManager.cs
M       PracticalLesson1/Program.cs
Switched to branch 'master'
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git pull origin master
From https://github.com/JuliaSherstneva/PracticalLesson1
 * branch            master     -> FETCH_HEAD
Already up to date.
```

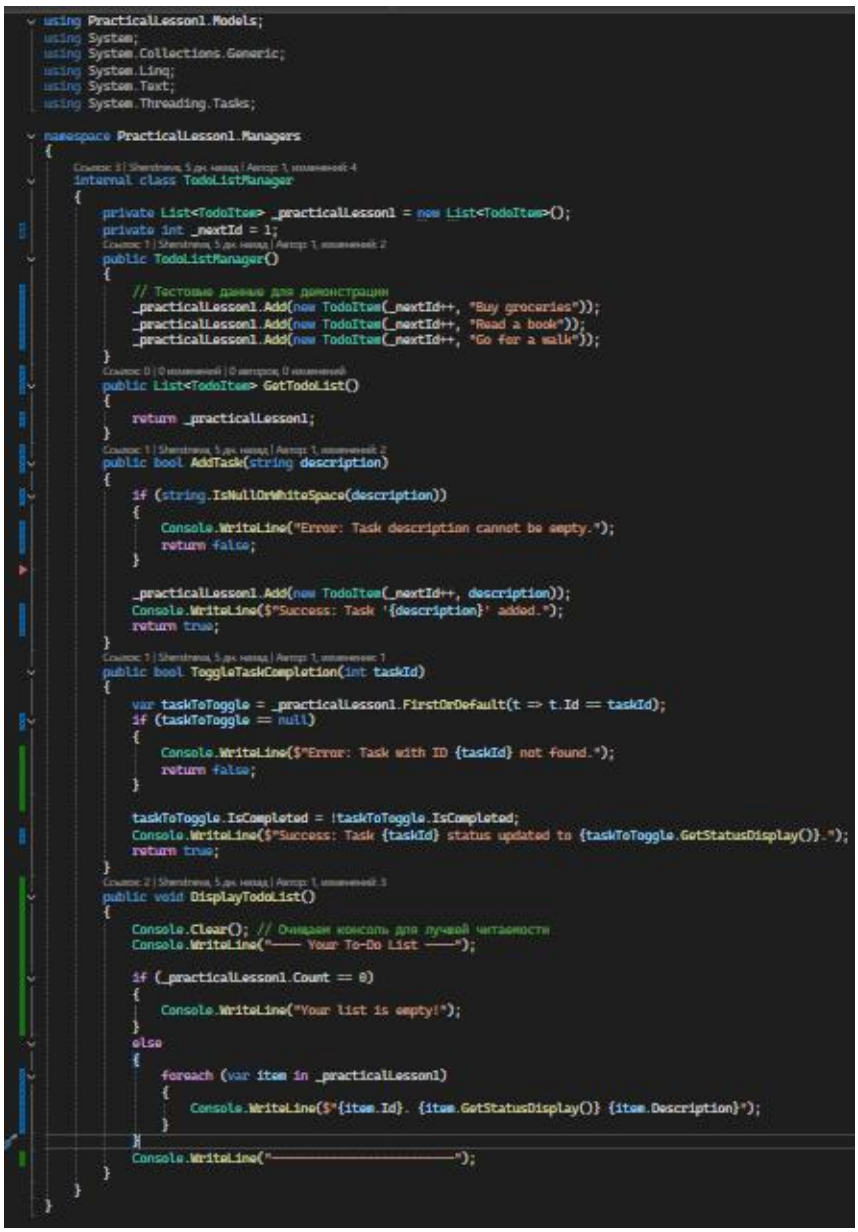
Создайте новую ветку для реализации интерактивного режима:

git checkout -b feature/interactive-mode

```
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git checkout -b feature/interactive-mode
Switched to a new branch 'feature/interactive-mode'
```

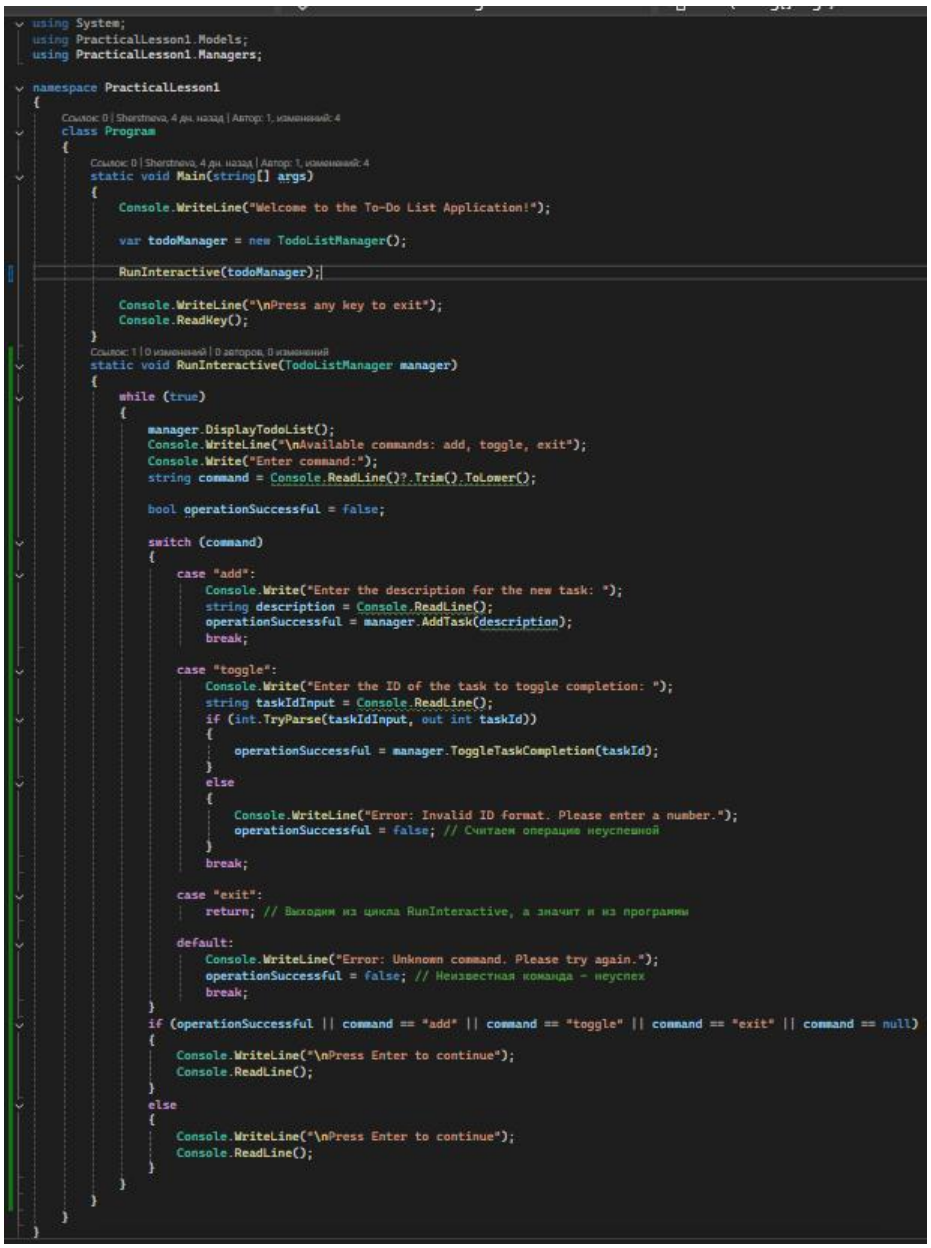
2. Полная реализация интерактивного режима: Models/TodoItem.cs

Этот файл остается без изменений со времен второго практического занятия, так как модель задачи не требует доработки. Managers/TodoListManager.cs:



```
using PracticalLesson1.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PracticalLesson1.Managers
{
    internal class TodoListManager
    {
        private List<TodoItem> _practicalLesson1 = new List<TodoItem>();
        private int _nextId = 1;
        public TodoListManager()
        {
            // Тестовые данные для демонстрации
            _practicalLesson1.Add(new TodoItem(_nextId++, "Buy groceries"));
            _practicalLesson1.Add(new TodoItem(_nextId++, "Read a book"));
            _practicalLesson1.Add(new TodoItem(_nextId++, "Go for a walk"));
        }
        public List<TodoItem> GetTodoList()
        {
            return _practicalLesson1;
        }
        public bool AddTask(string description)
        {
            if (string.IsNullOrWhiteSpace(description))
            {
                Console.WriteLine("Error: Task description cannot be empty.");
                return false;
            }

            _practicalLesson1.Add(new TodoItem(_nextId++, description));
            Console.WriteLine($"Success: Task '{description}' added.");
            return true;
        }
        public bool ToggleTaskCompletion(int taskId)
        {
            var taskToToggle = _practicalLesson1.FirstOrDefault(t => t.Id == taskId);
            if (taskToToggle == null)
            {
                Console.WriteLine($"Error: Task with ID {taskId} not found.");
                return false;
            }

            taskToToggle.IsCompleted = !taskToToggle.IsCompleted;
            Console.WriteLine($"Success: Task {taskId} status updated to {taskToToggle.GetStatusDisplay()}.");
            return true;
        }
        public void DisplayTodoList()
        {
            Console.Clear(); // Очищаем консоль для лучшей читаемости
            Console.WriteLine("--- Your To-Do List ---");

            if (_practicalLesson1.Count == 0)
            {
                Console.WriteLine("Your list is empty!");
            }
            else
            {
                foreach (var item in _practicalLesson1)
                {
                    Console.WriteLine($"{item.Id}. {item.GetStatusDisplay()} {item.Description}");
                }
            }
            Console.WriteLine("-----------------------");
        }
    }
}
```

Program.cs:



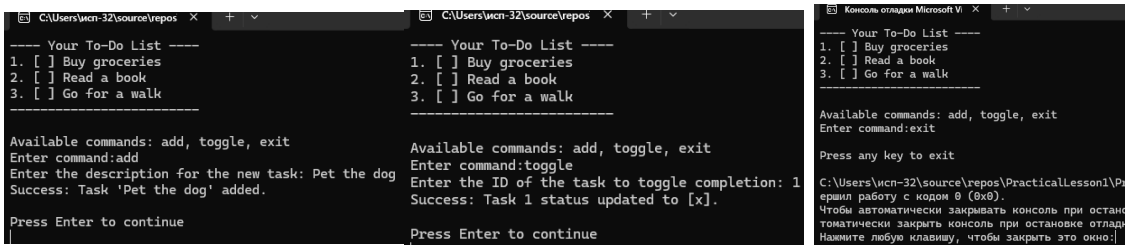
```
using System;
using PracticalLesson1.Models;
using PracticalLesson1.Managers;

namespace PracticalLesson1
{
    Ссылок: 0 | Sherstneva, 4 дн. назад | Автор: 1, изменений: 4
    class Program
    {
        Ссылок: 0 | Sherstneva, 4 дн. назад | Автор: 1, изменений: 4
        static void Main(string[] args)
        {
            Console.WriteLine("Welcome to the To-Do List Application!");

            var todoManager = new TodoListManager();

            RunInteractive(todoManager);

            Console.WriteLine("\nPress any key to exit");
            Console.ReadKey();
        }
        Ссылок: 1 | 0 изменений | 0 авторов, 0 изменений
        static void RunInteractive(TodoListManager manager)
        {
            while (true)
            {
                manager.DisplayTodoList();
                Console.WriteLine("\nAvailable commands: add, toggle, exit");
                Console.Write("Enter command:");
                string command = Console.ReadLine()?.Trim().ToLower();

                bool operationSuccessful = false;

                switch (command)
                {
                    case "add":
                        Console.Write("Enter the description for the new task: ");
                        string description = Console.ReadLine();
                        operationSuccessful = manager.AddTask(description);
                        break;

                    case "toggle":
                        Console.Write("Enter the ID of the task to toggle completion: ");
                        string taskIdInput = Console.ReadLine();
                        if (int.TryParse(taskIdInput, out int taskId))
                        {
                            operationSuccessful = manager.ToggleTaskCompletion(taskId);
                        }
                        else
                        {
                            Console.WriteLine("Error: Invalid ID format. Please enter a number.");
                            operationSuccessful = false; // Считаем операция неуспешной
                        }
                        break;

                    case "exit":
                        return; // Выходим из цикла RunInteractive, а значит и из программы

                    default:
                        Console.WriteLine("Error: Unknown command. Please try again.");
                        operationSuccessful = false; // Неизвестная команда - неуспех
                        break;
                }
                if (operationSuccessful || command == "add" || command == "toggle" || command == "exit" || command == null)
                {
                    Console.WriteLine("\nPress Enter to continue");
                    Console.ReadLine();
                }
                else
                {
                    Console.WriteLine("\nPress Enter to continue");
                    Console.ReadLine();
                }
            }
        }
    }
}
```

3. Тестирование интерактивного режима:

Запустите приложение (dotnet run или через Visual Studio).

Попробуйте все команды: add (с корректным и некорректным описанием), toggle (с корректным Id, несуществующим Id, нечисловым вводом), exit.

Убедитесь, что все работает как ожидалось.



```
C:\Users\исп-32\source\repos

---- Your To-Do List ----
1. [ ] Buy groceries
2. [ ] Read a book
3. [ ] Go for a walk
--------------------------

Available commands: add, toggle, exit
Enter command:add
Enter the description for the new task: Pet the dog
Success: Task 'Pet the dog' added.

Press Enter to continue
```

```
C:\Users\исп-32\source\repos

---- Your To-Do List ----
1. [ ] Buy groceries
2. [ ] Read a book
3. [ ] Go for a walk
--------------------------

Available commands: add, toggle, exit
Enter command:toggle
Enter the ID of the task to toggle completion: 1
Success: Task 1 status updated to [x].

Press Enter to continue
```

```
Консоль отладки Microsoft Vi

---- Your To-Do List ----
1. [ ] Buy groceries
2. [ ] Read a book
3. [ ] Go for a walk
--------------------------

Available commands: add, toggle, exit
Enter command:exit

Press any key to exit

C:\Users\исп-32\source\repos\PracticalLesson1\Pr
ершил работу с кодом 0 (0x0).
Чтобы автоматически закрывать консоль при остан
томатически закрыть консоль при остановке отлад
Нажмите любую клавишу, чтобы закрыть это окно:
```

4. Фиксация изменений:

Зафиксируйте все завершенные изменения: cd PracticalLesson1

git add .

git commit -m "Feat: Implement interactive mode with add, toggle, exit commands"

```
PS C:\Users\исп-32\source\repos\PracticalLesson1> cd PracticalLesson1
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git add .
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git commit -m "Feat: Implement interactive mode with add, toggle, exit commands"
[master b5ad245] Feat: Implement interactive mode with add, toggle, exit commands
 2 files changed, 103 insertions(+), 34 deletions(-)
```

5. Слияние ветки в master:

Переключитесь на основную ветку: git checkout master

Синхронизируйте master с удаленной: git pull origin master

```
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git checkout master
Already on 'master'
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git pull origin master
From https://github.com/JuliaSherstneva/PracticalLesson1
 * branch            master     -> FETCH_HEAD
Already up to date.
```

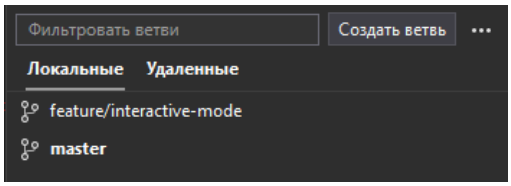
Слейте вашу ветку feature/interactive-mode в master: git merge feature/interactive-mode

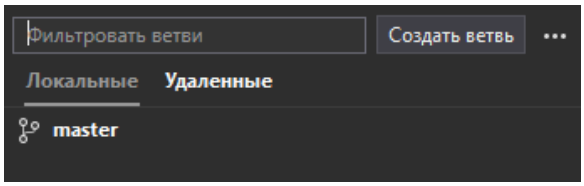
```
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git merge feature/interactive-mode
Already up to date.
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git push origin master
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 2.00 KiB | 1023.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/JuliaSherstneva/PracticalLesson1.git
   1546afe..b5ad245  master -> master
```

6. Удаление ветки (опционально):

Удалите локальную ветку: git branch -d feature/interactive-mode

```
PS C:\Users\исп-32\source\repos\PracticalLesson1\PracticalLesson1> git branch -d feature/interactive-mode
Deleted branch feature/interactive-mode (was 1546afe).
```





Результат: Реализован интерактивный режим консольного приложения, включая добавление, отметку выполнения и выход. Код рефакторирован для лучшей структуры и удобства использования. Все

изменения успешно слиты в основную ветку main, и конечная рабочая версия программы загружена на GitHub.

Структура проекта:

```
TodoListApp/
├── bin/
├── obj/
├── Managers/
│   └── TodoListManager.cs
├── Models/
│   └── TodoItem.cs
├── Program.cs
└── TodoListApp.csproj
```

Этот набор файлов представляет собой полностью рабочую версию вашего консольного приложения для управления списком дел с интерактивным режимом, готовым к дальнейшему развитию.

Заключение

Таким образом, в ходе выполнения работы была создана новая ветка для реализации интерактивного режима консольного приложения, проведён рефакторинг кода для улучшения структуры, слиты изменения в основную ветку, получена полностью рабочая версия программы.

```
using PracticalLesson1.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace PracticalLesson1.Managers
{
    internal class TodoListManager
    {
        private List<TodoItem> _practicalLesson1 = new List<TodoItem>();
        private int _nextId = 1;

        public TodoListManager()
        {
            _practicalLesson1.Add(new TodoItem(1, "Buy groceries"));
            _practicalLesson1.Add(new TodoItem(2, "Read a book"));
            _practicalLesson1.Add(new TodoItem(3, "Go for a walk"));
        }
        public List<TodoItem> GetTodoList()
        {
            return _practicalLesson1;
        }
        public bool AddTask(string description)
        {
            if (!string.IsNullOrWhiteSpace(description))
            {
                _practicalLesson1.Add(new TodoItem(_nextId++, description));
                Console.WriteLine("Task added successfully");
            }
            else
            {
                Console.WriteLine("Task description cannot be empty");
            }
        }
        public bool ToggleTaskCompletion(int taskId)
        {
            var taskToToggle = _practicalLesson1.FirstOrDefault(t => t.Id ==
taskId);
            if (taskToToggle != null)
            {
                taskToToggle.IsCompleted = !taskToToggle.IsCompleted;
                Console.WriteLine($"Задание {taskId} выполнено статус обновлён");
                return true;
            }
            else
            {
                Console.WriteLine($"Задание с ID {taskId} не найден");
                return false;
            }
        }
```

```
        public void DisplayTodoList()
        {
            Console.WriteLine("\n--- Your To-Do List ---");
            if (_practicalLesson1.Count == 0)
            {
                Console.WriteLine("Your list is empty!");
            }
            else
            {
                foreach (var item in _practicalLesson1)
                {
                    Console.WriteLine($"{item.Id}. {item.GetStatusDisplay()}
{item.Description}");
                }
            }
            Console.WriteLine("-------------------------");
        }
    }
}
```