|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **РОСЖЕЛДОР**  **федеральное государственное бюджетное образовательное учреждение высшего образования**  **«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ПУТЕЙ СООБЩЕНИЯ»**  **(СГУПС)**  Кафедра «Информационные технологии на транспорте»  **Лабораторная работа №4**  по дисциплине «Технология программирования»  Игра «Space Invaders»   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Проверил: | |  | Выполнил: | | | | старший преподаватель | |  | студент гр. | | БИСТ-311 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.В. Спешилов  *Подпись* | |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.Е. Коноваленко  *подпись* | | | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *дата проверки* | |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *дата сдачи на проверку* | | | |  |  | | | |  |  | | |   Краткая рецензия:   |  | | --- | |  | |  | |  | |  | |  |  |  |  |  | | --- | --- | --- | |  |  |  | | *запись о допуске к защите* |  |  | |  |  |  | | *оценка по результатам защиты* |  | *подписи преподавателей и дата защиты* |   Новосибирск  2020 |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc65676329)

[2 Диаграмма классов 4](#_Toc65676330)

[3 Методы 5](#_Toc65676331)

[3.1 Методы Game 5](#_Toc65676332)

[3.2 Методы Player 5](#_Toc65676333)

[3.3 Методы Unit 5](#_Toc65676334)

[3.4 Методы GameFileManager 6](#_Toc65676335)

[3.5 Методы MainWindow 6](#_Toc65676336)

[3.6 Методы UserNamer 6](#_Toc65676337)

[3.7 Методы WpfVisualizer 7](#_Toc65676338)

[4 Метрики кода 8](#_Toc65676339)

[ЗАКЛЮЧЕНИЕ 9](#_Toc65676340)

# **ВВЕДЕНИЕ**

Целью данной лабораторной работы является освоение принципов правильного построение методов. Для этого нужно обновить методы решения. Требуется выполнить следующие задачи:

1) Найти методы,

a. которые имеют более одной ответственности

b. код которых находится на разных уровнях абстракции

c. у которых очень много параметров

d. параметры которых не используются

e. которые имеют случайную, логическую или процедурную связности

f. которые «завидуют» другим классам

2) Исправить найденные методы

3) Определить связность операций для каждого метода

4) Определить цель создания для каждого метода

5) При выделении новых открытых методов расширить тесты

# **2 Диаграмма классов**

Диаграмма классов представлена на рисунке 1.

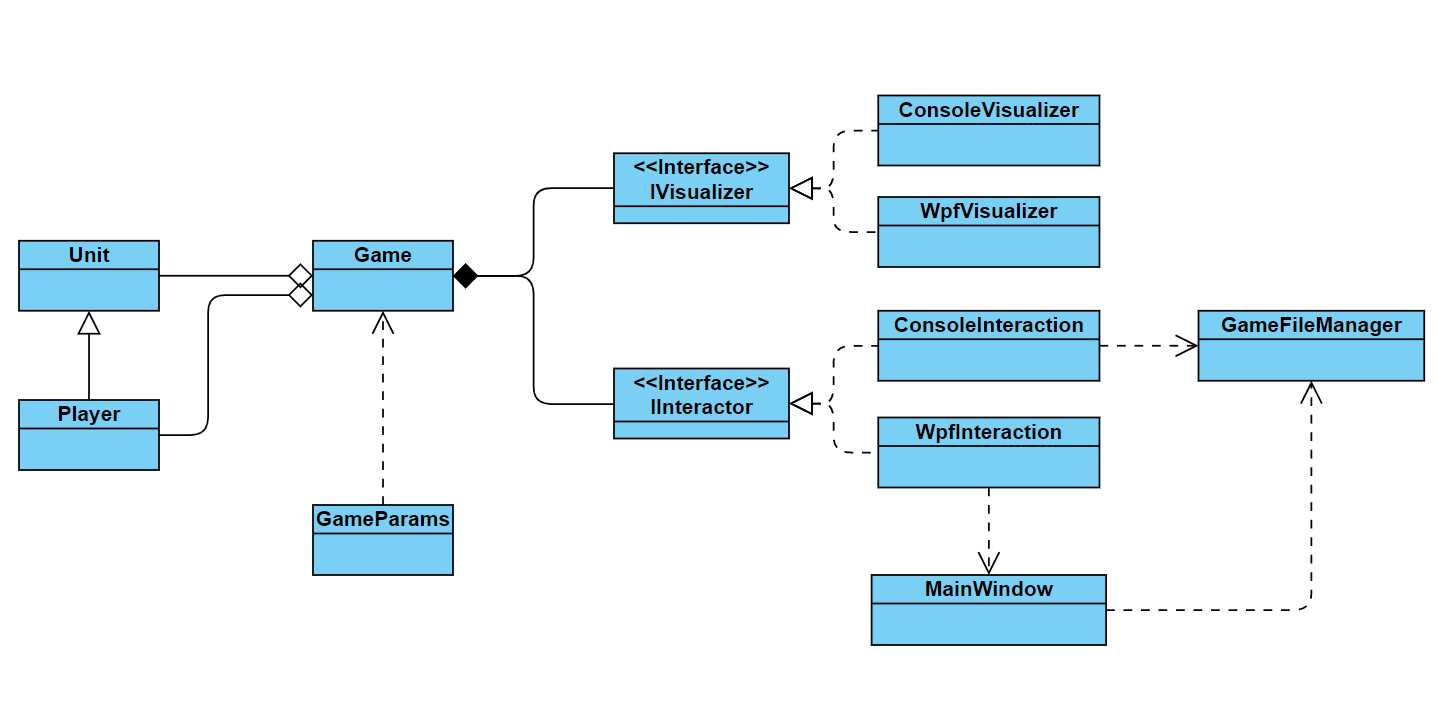
****

Рисунок 1 – Диаграмма классов

# **3 Методы**

## 3.1 Методы Game

1. CallBacker() – Метод включает в себя все что нужно выполнять по таймеру, используется для подписки. Связность – временная. Причина создания – изоляция сложности.
2. Start() – Запускает игру, тем что создает таймер и подписывает на него динамические процессы игры.
3. ProjectilePhx(object param) – Метод перемещает снаряды по полю. Причина создания – изоляция сложности. Связность – последовательная.
4. Comand(string action) – Переводит внешние команды в вызовы методов. Причина создания - упрощение сложных булевых проверок, формирование понятной промежуточной абстракции.
5. EnemyGen() – генерирует противников. Причина создания – снижение сложности. Связность – последовательная.
6. CheckGameOver() – проверяет конец игры. Причина создания – снижение сложности. Связность – последовательная.
7. GameOver(IInteractor interactor) – останавливает игру и записывает счет игрока. Связность – временная. Причина создания – снижение сложность.

## 3.2 Методы Player

Move(int i) – перемещает игрока. Причина создания – создание понятной промежуточной абстракции. Связность – функциональная.

## 3.3 Методы Unit

Collision(int damage) – наносит урон юниту, возвращает булевое значение убил или нет. Причина создания – создание промежуточной абстракции. Связность – временная.

## 3.4 Методы GameFileManager

1. ReadRecordsFromFile(string recordsPath) – восстанавливает словарь рекордов по файлу. Причина создания – снижение сложности. Связность – последовательная.
2. SaveRecord(string recordsPath, (string name, int score) playerInfo) – сохраняет рекорды в файл, добавляя туда новую запись. Причина создания – снижение сложности. Связность – последовательная.
3. WriteRecord(string recordsPath, Dictionary<string, int> temp) – выполняет запись в файл рекордов. Причина создания – снижение сложности, создание промежуточной абстракции. Связность – функциональная.
4. RefreshRecords(Dictionary<string, int> temp, (string name, int score) playerInfo) – обновляет рекорды(добавляет новый или улучшает старый рекорд, после чего сортирует таблицу рекордов). Причина создания – снижение сложности, создание промежуточной абстракции. Связность – последовательная.

## 3.5 Методы MainWindow

1. User\_KeyDown(object sender, KeyEventArgs e) – обрабатывает нажатия игрока. Причина создания – упрощение булевых проверок. Связность – функциональная.

## Методы UserNamer

1. ChooseFile(object sender, RoutedEventArgs e) – позволяет пользователю удобно выбрать файл куда необходимо сохранить рекорды(определяет путь к файлу). Причина создания – снижение сложности. Связность – последовательная.
2. SaveToFile(object sender, RoutedEventArgs e) – сохраняет рекорд по указанному пути. Причина создания – снижение сложности. Связность – временная.

## Методы WpfVisualizer

1. Drawer(object state) – отрисовывает игровое поле на канвасе. Причина создания снижение сложности. Связность – последовательная.
2. EnemyColorPicker(int hp) – определяет цвет противника в зависимости от здоровья. Причина создания – формирование понятной промежуточной абстракции. Связность – функциональная.

# **4 Метрики кода**

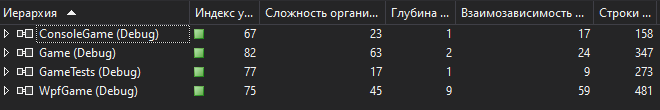


Рисунок 2 – Метрики

# **ЗАКЛЮЧЕНИЕ**

В ходе лабораторной работы были освоены принципы правильного построения методов. Были найдены и исправлены методы, не соответствующие требованиям. Для каждого метода была определена причина создания и связность операций. В результате по сравнению с предыдущей лабораторной работой количество строк кода осталось примерно таким же, вырос индекс удобства поддержки, т.к. из методов были вынесены части из-за чего их стало проще изменить. Классы не были изменены, поэтому глубина наследования и взаимозависимость классов остались прежними.