**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**По лабораторной работе №2**

**по дисциплине «ООП»**

**Тема**: “**Добавления игрока и элементов для поля”**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Судаков Е.В |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2020

1. **Диаграмма классов**

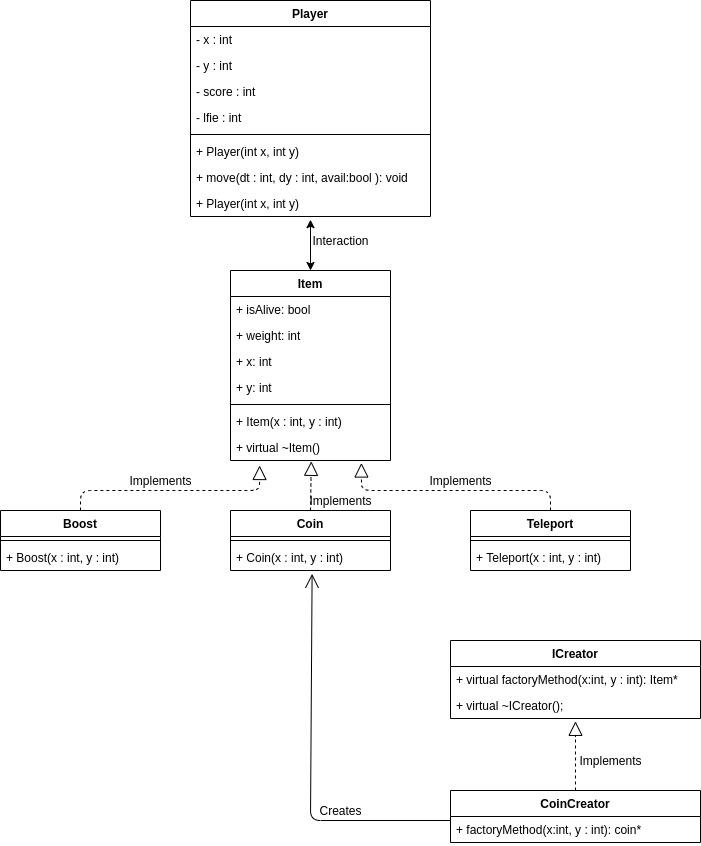


Рисунок 1. Uml диаграмма

1. **Описание архитектурный решений**

Создан класс игрока(Player), которым управляет пользователь(метод move()). Объект класса игрока может перемещаться по полю, а также взаимодействовать с элементами поля. Для элементов поля создан общий интерфейс(Item) и реализованы 3 разных класса элементов(Coin, Boost, Teleport). Для взаимодействия игрока с элементом используется перегруженный оператор

Player& operator+=(const Item &right) {

this->score += right.weight;

return \*this;

}

Для создания элементов поля используется паттерн **Фабричный метод.**

**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**По лабораторной работе №3**

**по дисциплине «ООП»**

**Тема**: “**Добавление логирования”**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Судаков Е.В |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2020

**1. Диаграмма классов**

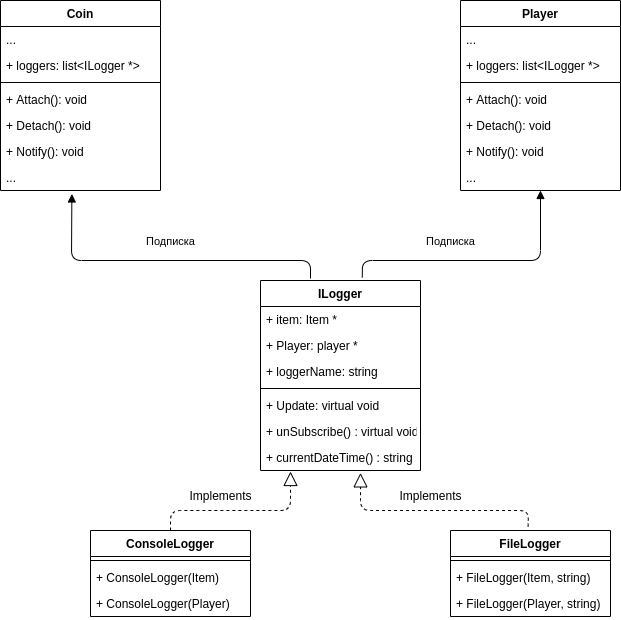


Рисунок 1. Uml диаграмма

**2. Описание архитектурный решений**

Создан интерфейс для классов логирования. От этого интерфейса наследуются классы логирования для консоли и для файла. Взаимодействие между классами логирования и объектами осуществляется с помощью паттерна **Наблюдатель.**

1. **Пример работы**

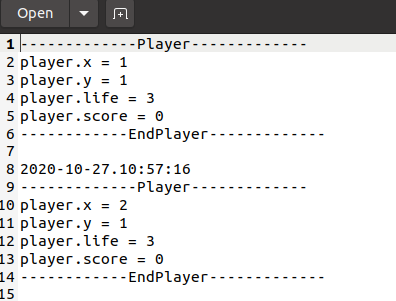
****

Рис.2 Логи

**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**По лабораторной работе №4**

**по дисциплине «ООП»**

**Тема**: “**Добавление класса управления игрой”**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Судаков Е.В |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2020

**1. Диаграмма классов**

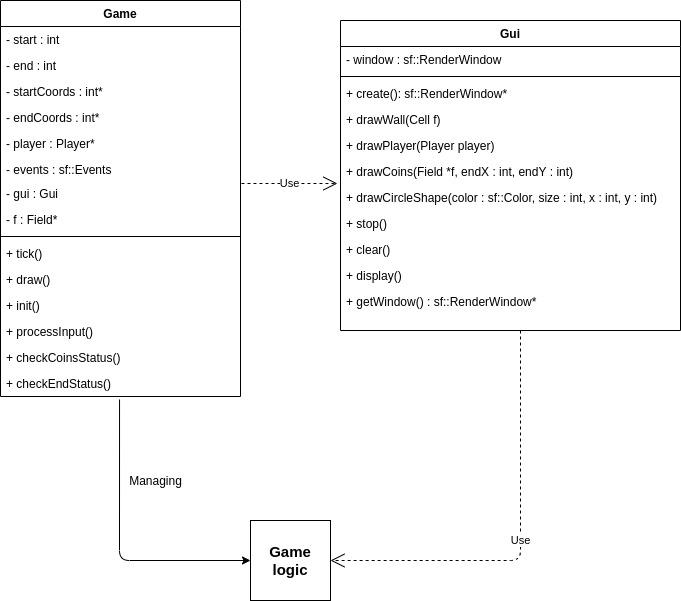


Рисунок 1. Uml диаграмма

**2. Описание архитектурный решений**

Создан класс игры(Game), через который пользователь взаимодействует с игрой. Управление игроком, начало новой игры, завершение игры.

Класс используется паттерн “**Фасад**”, благодаря чему запуск игры происходит внешне очень наглядно:

int main() {

Game game;

game.init();

return 0;

}

Однако внутри происходят довольно много вещей: создается поле, инициализируются его элементы. Создается объект игрока, который взаимодействует с полем и подчиняется командам пользователя.

После того, как игра инициализировалась, запускается классический game-loop :

while (gui.getWindow()->isOpen()) {

tick();

draw();

}

Где в методе tick() происходит обработка пользовательского ввода и обновление состояния логики. Метод draw() вызывает соответствующие методы из класса графического интерфейса пользователя (Gui).

Класс Gui специально спроектирован так, что его можно с легкостью заменить на другую библиотеку (сейчас sfml), не внося существенный изменений в класс управления игрой, и тем более в логику(она не знает ни про Gui, ни про саму игру).

1. **Пример работы**

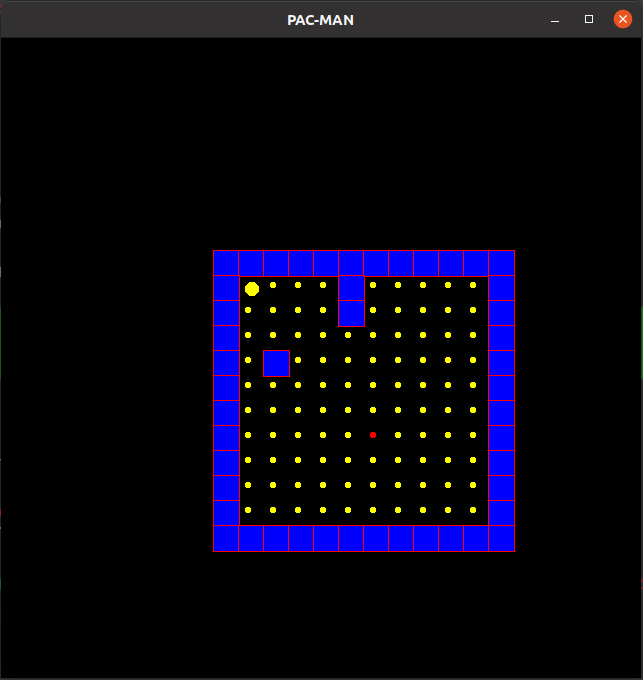
****

Рис.2 Игра после запуска

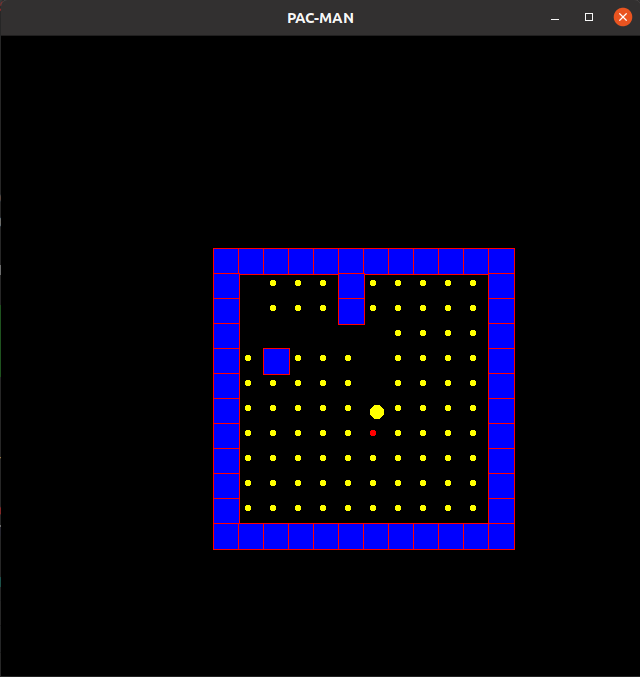
****

Рис.3 Игрок собирает монетки