**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: Полиморфизм

|  |  |  |
| --- | --- | --- |
| Студент гр. 2381 |  | Двиков Д.В. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

**Цель работы.**

Изучить основы парадигмы объектно-ориентированного программирования. Реализовать Уровни абстракции,  Создать класс игры,  cоздать класс (или набор классов) считывания ввода пользователя

**Задание**

а) Создать класс игры. Класс игры отвечает за запуск игры (в начале и во время текущей сессии), выход из игры, выбор уровня, а также инициализирующего перемещение игрока. Также класс должен проводить проверку проигрыша или выигрыша игрока и давать возможность начать новую игру или завершить работу программы.

б) Создать класс (или набор классов) считывания ввода пользователя. Данный класс(ы) должен сопоставить ввод пользователя с командой, которую необходимо выполнить. И передать эту информацию в класс игры. Клавиши управления (то на какую клавишу назначено определенное действия) должны считываться из файла. Класс(ы) должен быть разработан так, чтобы:

* потенциально можно было масштабировать управление с минимальным количеством изменений в коде
* можно было заменить схему управления, например, заменить ввод команду через терминал на чтение команд из файла или из сети с другого устройства

Примечания:

* После считывания клавиши, считанный символ должен сразу обрабатываться, и далее работа должна проводить с сущностью, которая представляет команду.
* Для представления команды можно разработать системы классов или использовать перечисление enum.
* Хорошей практикой является создание “прослойки” между считыванием/обработкой команды и классом игры, которая сопоставляет команду и вызываемым методом игры. Существуют альтернативные решения без явной “прослойки”
* При считывания управления необходимо делать проверку, что на все команды назначена клавиша, что на одну клавишу не назначено две команды, что на одну команду не назначено две клавиши.
* При работе с файлом используйте идиому RAII.

**Выполнение работы**

Определяется класс InputHandler, который отвечает за обработку ввода команд из файла.

1. stringToCommand: Это приватный метод класса InputHandler, который преобразует строковое представление команды в перечисление Command. Он сопоставляет строковые команды (например, "QUIT", "MOVE\_UP", и т.д.) с соответствующими значениями перечисления Command.
2. InputHandler(const std::string& filename): Конструктор класса InputHandler открывает файл с именем filename и читает его содержимое. Он парсит файл построчно, ожидая формат "ключ значение", где ключ - это строка, а значение - это также строка, представляющая команду. Он использует метод stringToCommand, чтобы преобразовать строковую команду из файла в перечисление Command и сохраняет соответствие ключа и команды в commandMap.
3. ~InputHandler(): Деструктор закрывает файл, если он был открыт.
4. handleInput(const std::string& input): Этот метод принимает строку input в качестве аргумента и пытается найти соответствующую команду в commandMap. Если команда найдена, он возвращает соответствующее значение перечисления Command, иначе возвращает значение Command::NOTHING

Определяется класс Game, который отвечает за запуск игры

Конструктор класса Game, который устанавливает начальное состояние игры.

void Game::startGame(): Метод, который начинает игру.

Создается игровое поле Gamefield, игрок Player, контроллер игрока Playercontroller и представление PrintField.

Запрашивается у пользователя выбор уровня игры и в соответствии с выбором строится игровое поле.

Начинается основной игровой цикл, где происходит обработка команд, вводимых игроком через InputHandler из файла "command.txt". Проверяется условие победы и поражения, и игра завершается соответственно.

void Game::handleCommand(Player& player, Playercontroller& cnt, Command command): Метод для обработки команд игрока. В зависимости от команды он вызывает соответствующие методы у контроллера игрока (cnt).

bool Game::checkWin(Player &player, Playercontroller &cnt, Gamefield &field): Метод для проверки условий победы в игре.

bool Game::checkLoss(Player& player): Метод для проверки условий поражения в игре.

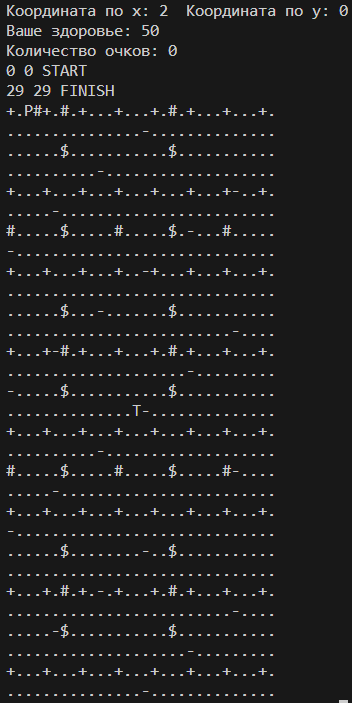
void Game::restart(): Метод для перезапуска игры, который сбрасывает состояние игры и вызывает startGame().

void Game::quit(): Метод для завершения игры.

Вывод:

В ходе выполнения лабораторной работы были изучены основы парадигмы объектно-ориентированного программирования, создан интерфейс игрового события и его реализации. Кроме этого были доработаны функции взаимодействия с игроком, создан класс, генерирующий игровое поле и заполняющий его определенными событиями.

**Результаты работы программы:**



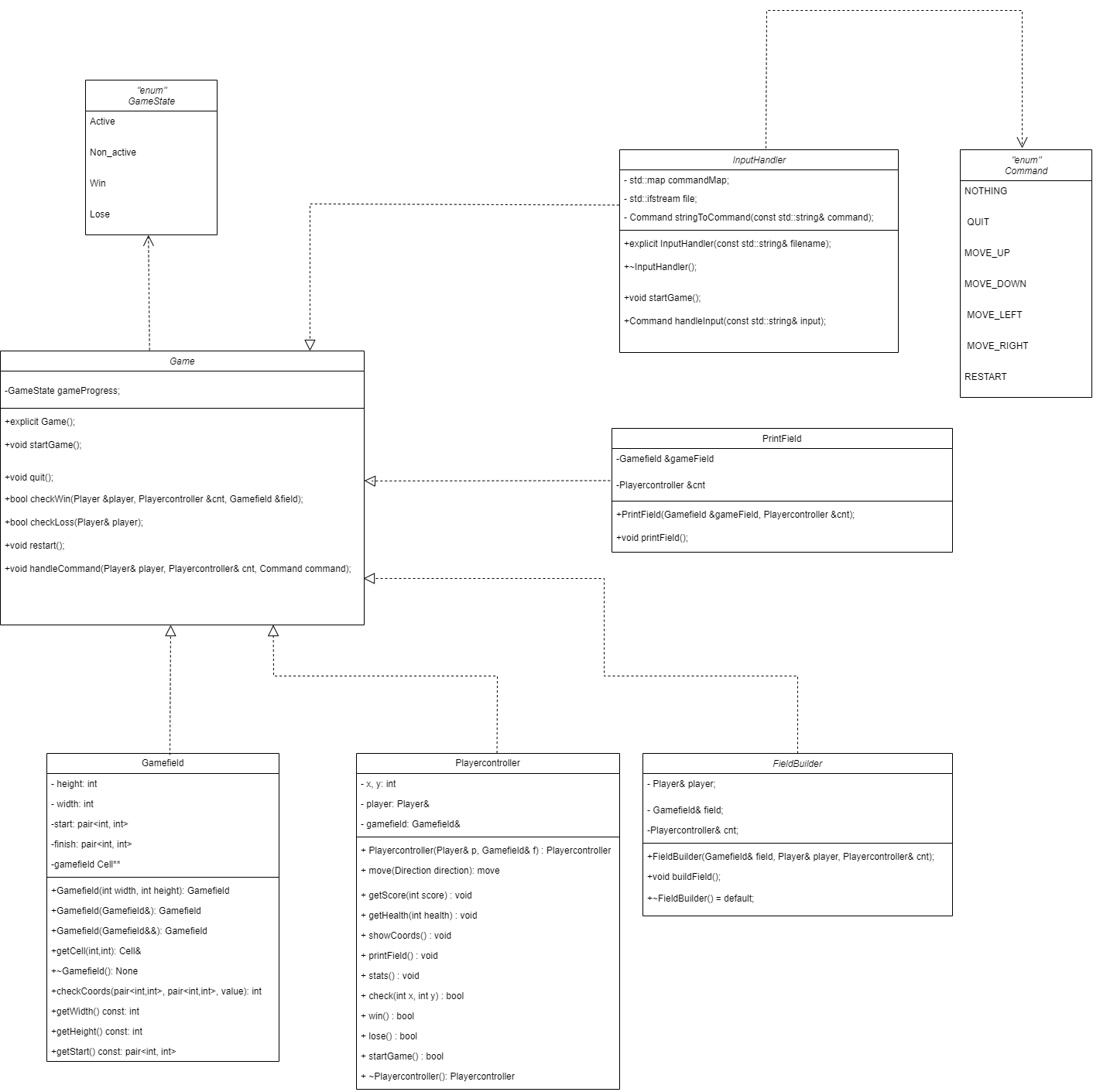
**Приложение В**  


Рисунок 1 – Uml -диаграмма