# МИНОБРНАУКИ РОССИИ

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

# «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

**Кафедра МО ЭВМ**

# ОТЧЕТ

**по лабораторной работе №5**

# по дисциплине «Объектно-ориентированное программирование»

**Тема: Управление, разделение на уровни абстракции.**

Студентка гр. 0382 Кривенцова Л.С.

Преподаватель Жангиров Т.Р.

Санкт-Петербург 2021

# Цель работы.

# Изучить принцип разделения на уровни абстракции, организовать управление игрой.

# Задание.

Необходимо организовать управление игрой (номинально через CLI). При управлении игрой с клавиатуры должна считываться нажатая клавиша, после чего происходит перемещение игрок или его взаимодействия с другими элементами поля.

Требования:

* Реализовать управление игрой. Считывание нажатий клавиш не должно происходить в классе игры, а должно происходить в отдельном наборе классов.
* Клавиши управления не должны жестко определяться в коде. Например, это можно определить в отдельном классе.
* Классы управления игрой не должны напрямую взаимодействовать с элементами игры (поле, клетки, элементы на клетках)
* Игру можно запустить и пройти.

# Выполнение работы.

**Ход решения:**

Для организации управления был создан класс *GameView*.

Выполняет роль прослойки между классом *Game*, который заведует ходом игры, и пользовательским интерфейсом: выводит на экран сообщения, считывает команды.

1. Определяются поля класса с модификатором доступа *private*:

* *bool going*; - логическая переменная, значение которой указывает, должен ли продолжаться ввод команд с клавиатуры

Поля, хранящие символьные переменные – ключи (команды) для направления игрока:

* *char right;*
* *char left;*
* *char up;*
* *char down;*

Поля, хранящие символьные переменные – ключи (команды) для прочих действий:

* *const char \* exit = "x";* - ключ остановки ввода команд (и остановки игры);
* *const char \* menu = "m";* - ключ для печати на экран меню;
* *const char \* instruction = "?";-* ключ для печати на экран инструкции к игре;
* *const char \* change\_keys = "k";* - ключ к изменению клавиш управления.

2. Реализованы методы с модификатором доступа *public*:

*GameView*() – конструктор класса, в котором инициализируются поля (в том числе задаются клавиши управления «по умолчанию» - [w,a,s,d]).

*void* *ChangeControlKeys*() – Метод, принимающий значения полей (ключей для управления игроком) с клавиатуры. Позволяет изменить управление на предложенное игроком (по запросу).

*Coordinates* *GameInput*() – Метод, принимающий ключ команды с клавиатуры. В зависимости от полученного ключа вызывает методы класса, описываемые ниже. Возвращает структуру координат (x и y), на которые нужно сдвинуть игрока (чем занимается класс *Game*, вызывающий этот метод). Если пользователь ввёл «мусор», не соответствующий ни одному из ключей инструкции, возвращаются нулевые координаты и игра завершается (выводя на экран соответствующее сообщение).

*void* *PrintMenu*() – Метод, распечатывающий на экран текст меню игры (с пояснениями ключей и их команд).

*void LeaveGame*() – Метод, печатающий прощание с игроком и устанавливающий флаг продолжения ввода команд(*going*) ложным.

*void GameOver*() – Метод, печатающий на экран текст проигрыша игры.

*void ReadInstruction()* – Метод, выводящий на экран текст с сутью и правилами игры.

*bool Goon()* – геттер для получения значения поля продолжения ввода команд(*going*).

Для завершения ясности игрового процесса был создан класс *ItemView*, который отвечает за вывод сообщений на экран при поднятии предмета.

1. Реализованы методы с модификатором доступа *public*:

*void BoxPrint(bool open, bool key)* – метод, поясняющий игроку, открыл ли он коробку, и что внутри. Принимает на вход логические значения (открытие коробки и содержания в ней ключа от выхода). Метод вызывается в методе *interplay* в классе *Box*.

*void HealPrint(int health)–* метод, печатающий на экран информацию о том, сколько очков здоровья было начислено игроку при нахождении аптечки. Принимает на вход целое число – количество добавленных очков здоровья. Метод вызывается в методе *interplay* в классе *Heal*.

**Результат работы программы:**

Рис 1. – демонстрация начала игры.

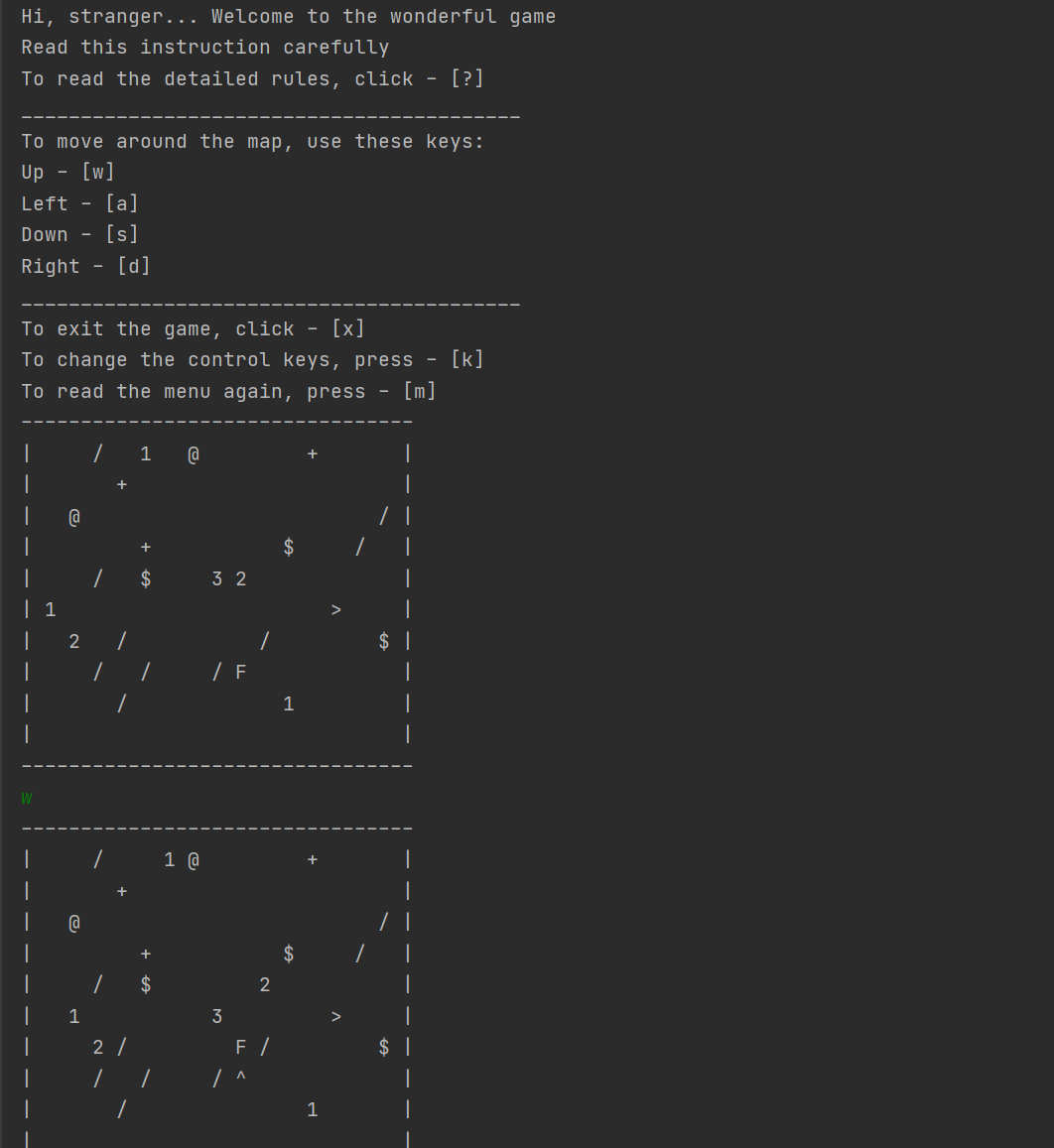


Рис 2. – демонстрация битвы с монстром.

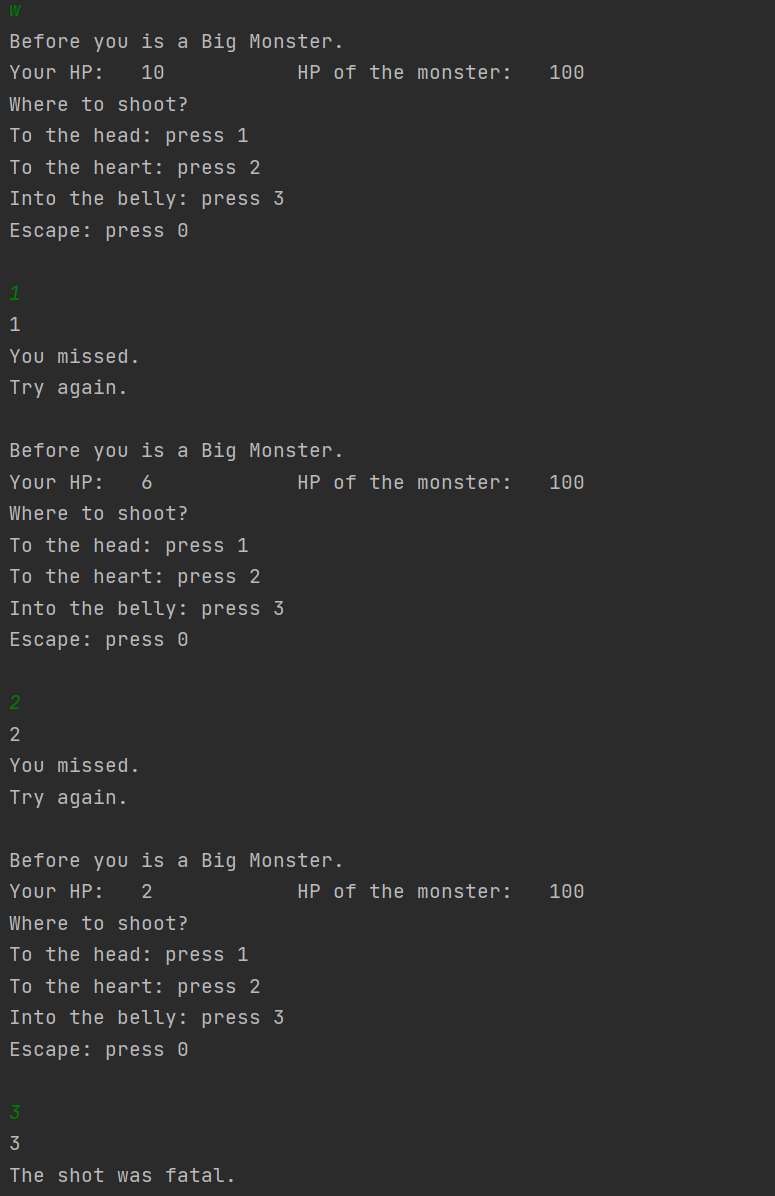


Рис 3, рис 4. – демонстрация находки вещи.

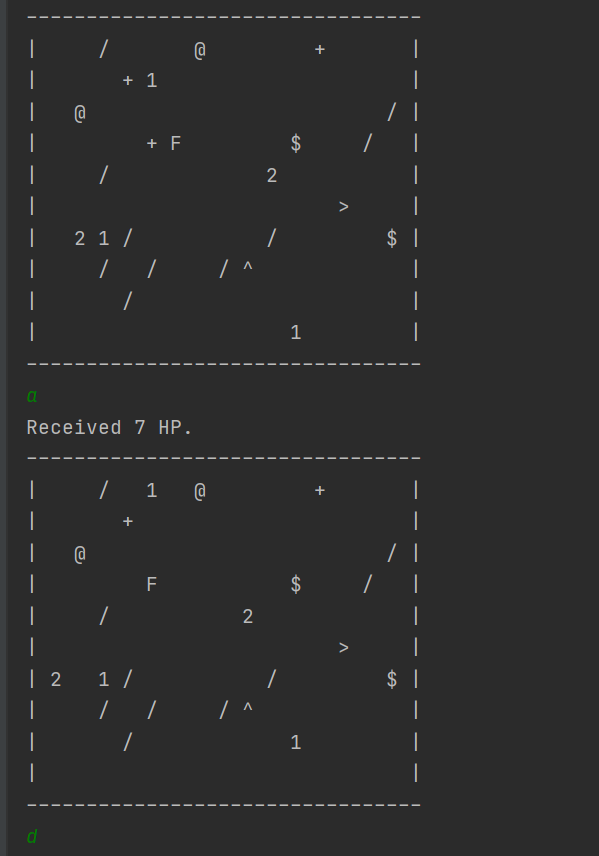
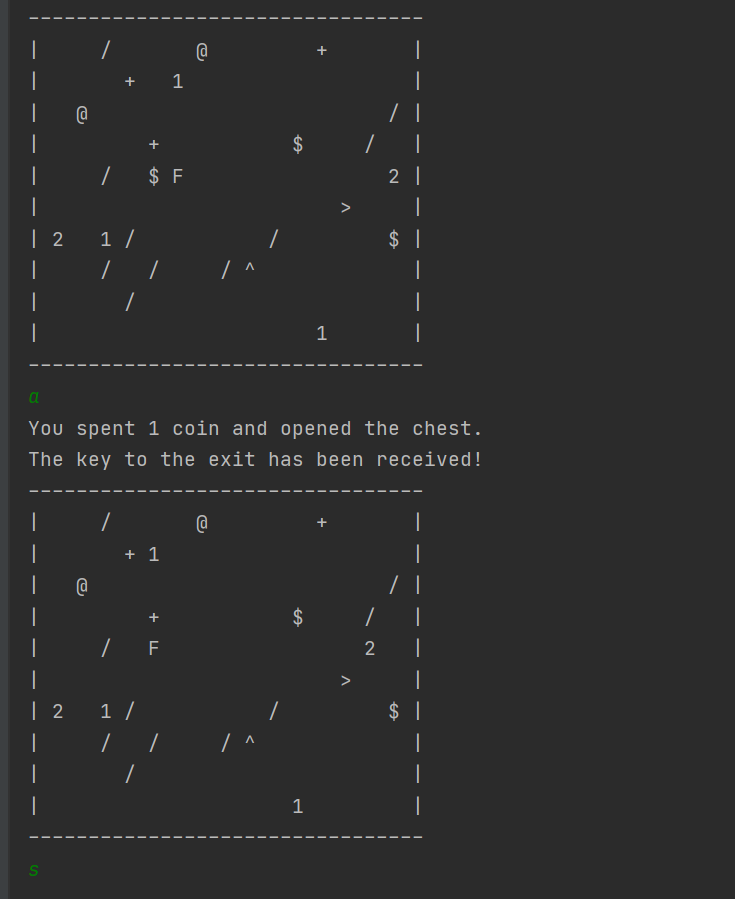
****

Рис 5. – демонстрация взаимодействия с телепортом.

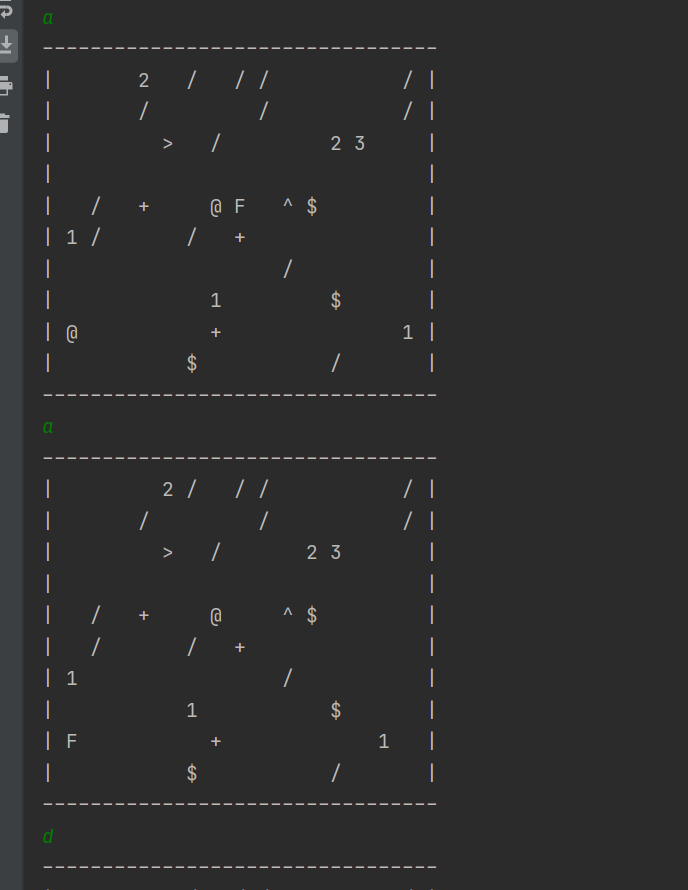


Рис 6. – демонстрация выигрыша игры.

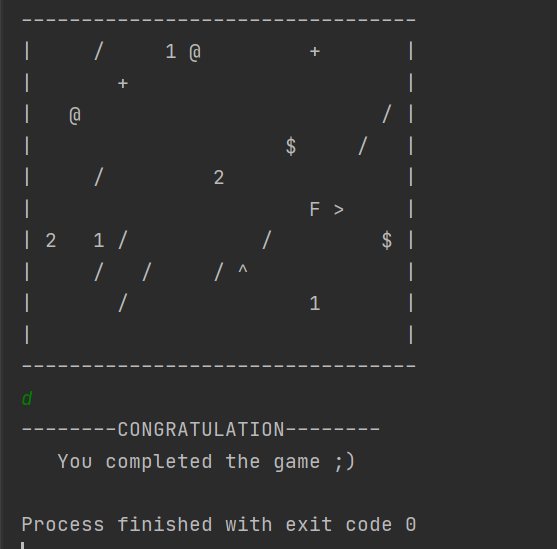
****

Рис 7. – демонстрация проигрыша игры.

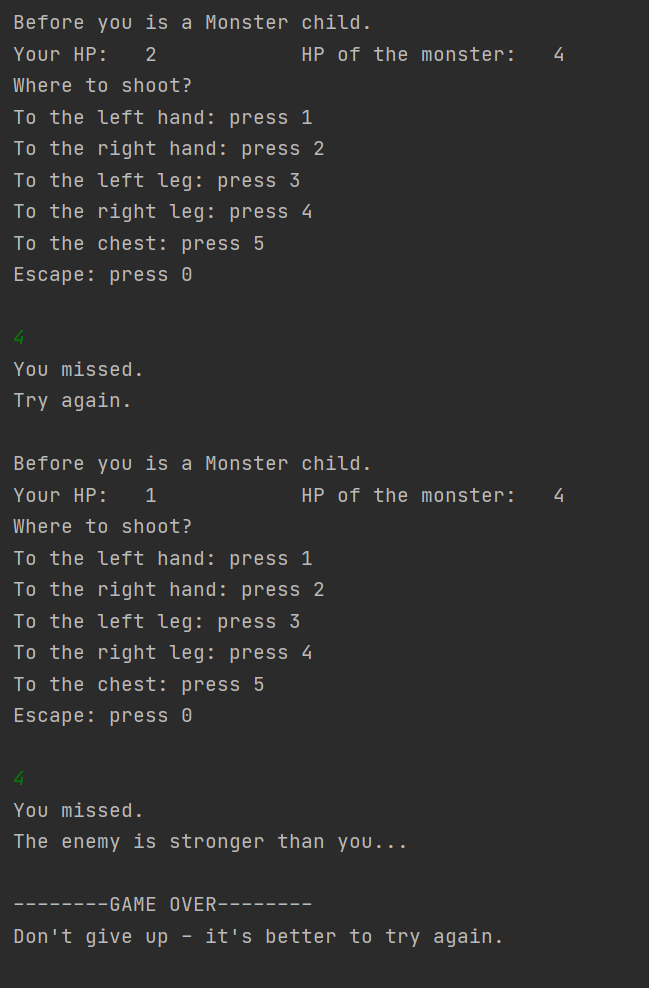
****

Рис 8. – демонстрация смены управления игры.

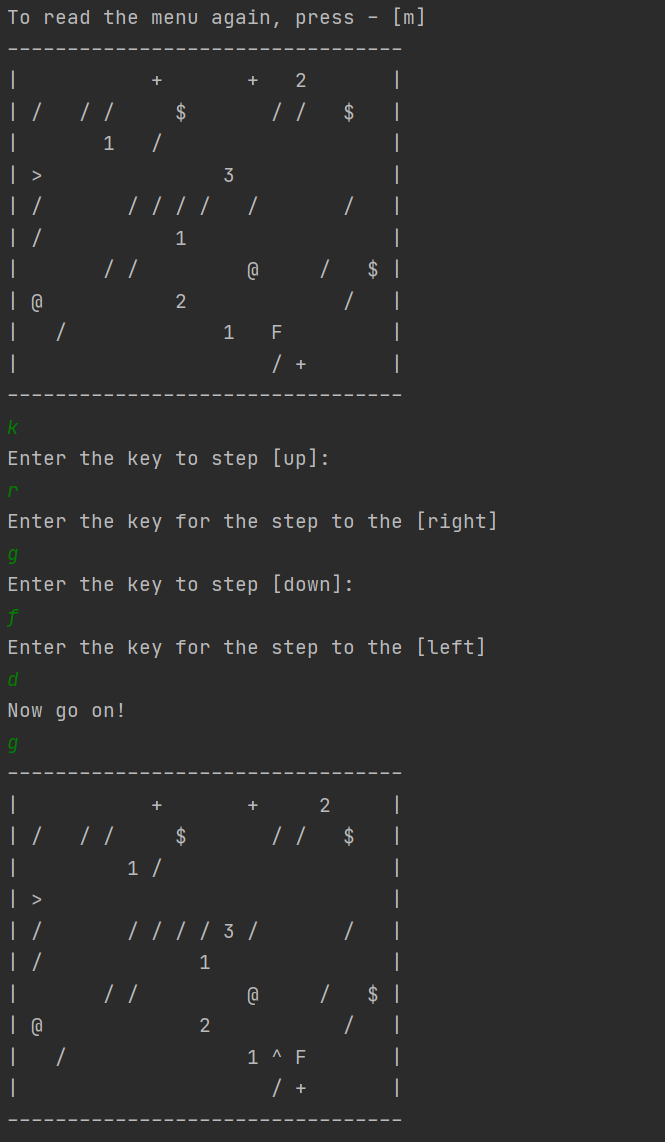
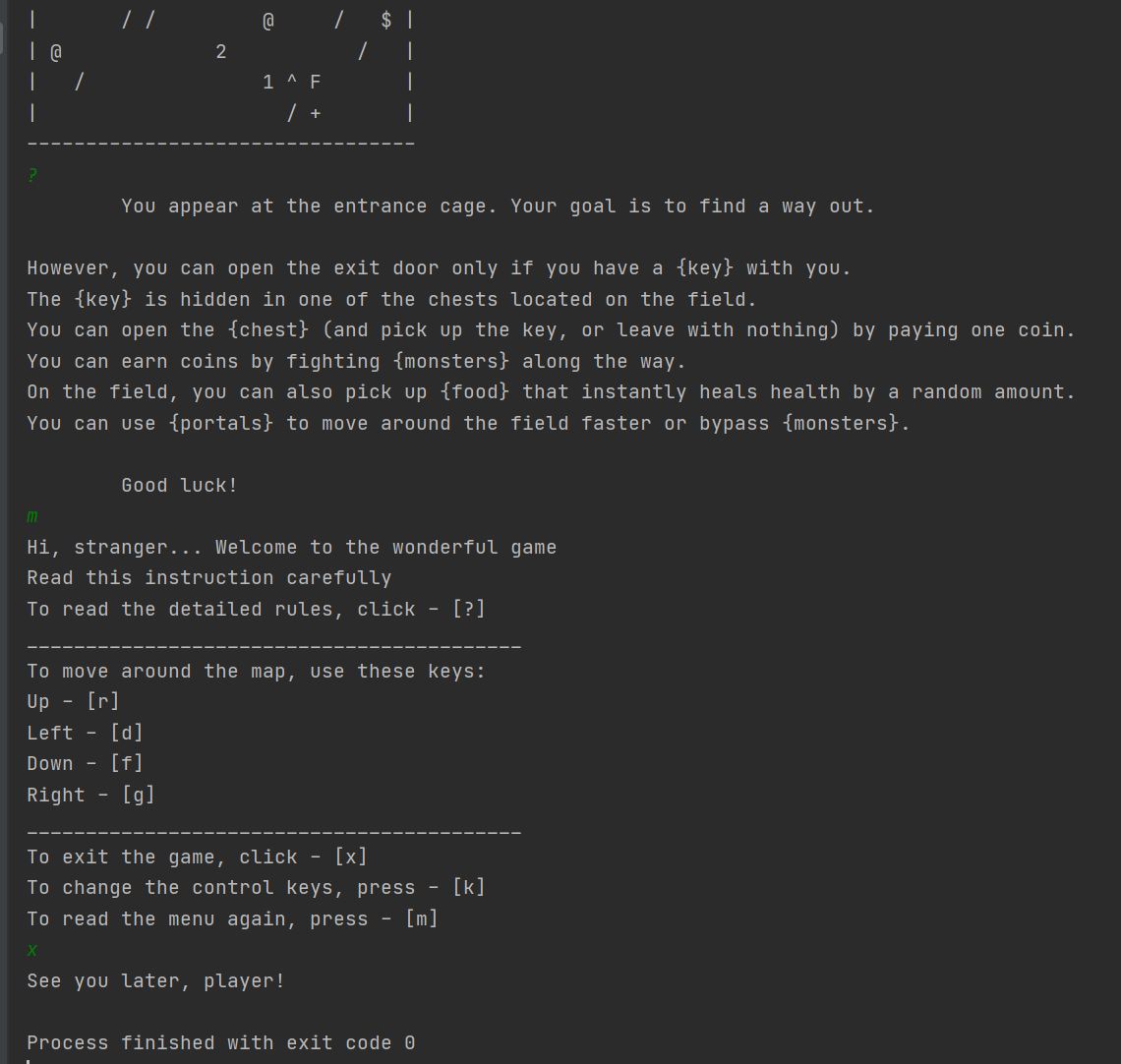
****

Рис 9. – демонстрация команд игры.

****

**UML-диаграмма межклассовых отношений:**

Находится в PDF-файле. Формат документов Word не позволяет вставить

копию, сохраняя читаемость и разборчивость диаграммы

# Выводы.

Был изучен принцип разделения на уровни абстракции, организовано управление игрой через отдельный класс.