

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Объектно-ориентированное
программирование» Тема: сериализация,
исключения.

Студентка гр. 0382

Здобнова К.Д.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Изучить принципы сериализации игры. Научиться сохранять текущее состояние игры.

Задание.

Сериализация - это сохранение в определенном виде состоянии программы с возможностью последующего его восстановления даже после закрытия программы. В рамках игры, это сохранения и загрузка игры.

Требования:

- Реализовать сохранения всех необходимых состояний игры в файл
- Реализовать загрузку файла сохранения и восстановления состояния игры
- Должны быть возможность сохранить и загрузить игру в любой момент
- При запуске игры должна быть возможность загрузить нужный файл
- Написать набор исключений, который срабатывают если файл с сохранением некорректный
- Исключения должны сохранять транзакционность. Если не удалось сделать загрузку, то программа должна находится в том состоянии, которое было до загрузки. То есть, состояние игры не должно загружаться частично

Потенциальные паттерны проектирования, которые можно использовать:

Снимок (Memento) - получение и восстановления состояния объектов при сохранении и загрузке

Выполнение работы.

Сохранение игры происходит через файл Restoring.txt, в который после каждого хода игрока и врагов записываются данные об объектах, лежащих на игровом поле – их местоположение и характеристики в момент сохранения игры. Если в файле записано «0» - значит, что игра не была загружена, то есть заново создаем поле и все объекты – начало игры. Если же при прочтении первое значение

«1» - это значит, что игра была сохранена.

В методе `void writeInFile(Board* board, Player* player, Boss* boss)` происходит запись состояния игры. При записи после «1» записываются данные поля и объекты на нем:

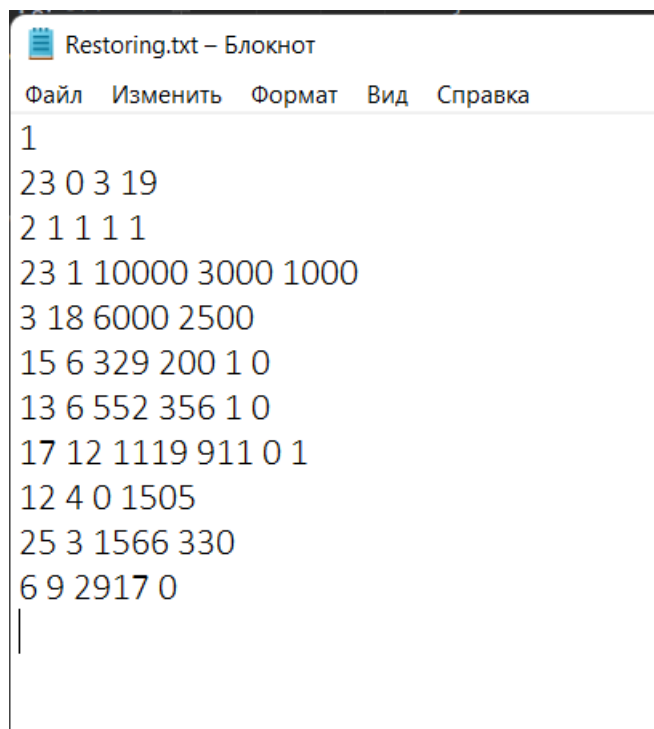
- Координаты клетки входа и выхода;
- Количество врагов каждого типа и количество вещей каждого типа;
- Информация об игроке: местоположение, значения `maxHP`, `HP`, `attack`;
- Информация о боссе: местоположение, значения `HP`, `attack`;
- Информация о врагах: местоположение, значения `HP`, `attack`, их ориентация хода на игровом поле;
- Информация о вещах: местоположение, значения `heal`, `attack`;

Метод `writeInFile()` вызывается в методе `moving()` – после того, как сходят игрок и враги (к-рые ходят автоматически после игрока).

В классе управления игры был расширен метод `startGame()`. Создаются объект класса логгера, объект класса управления игрока, также создается игровое поле, игрок и босс. Далее открывается файл сохранения игры на чтение. Если в нем записано «0», то игра будет находится в состоянии начала – создаются объекты. Также сам текстовый файл будет закрыт для чтения. Если в нем записано «1», то будут считываться состояния объектов, лежащих на поле. Также в зависимости от количества врагов и вещей будут создаваться объекты соответствующих классов. Далее вызывается метод `moving()` – игра продолжается.

В случаях, когда игра завершается – игрок выиграл или проиграл – открывается загрузочный файл, в который записывается «0», то есть игра завершилась, при открытии игры в следующий раз она начнется сначала.

Пример сохранения игры:



```
Restoring.txt – Блокнот
Файл  Изменить  Формат  Вид  Справка
1
23 0 3 19
2 1 1 1 1
23 1 10000 3000 1000
3 18 6000 2500
15 6 329 200 1 0
13 6 552 356 1 0
17 12 1119 911 0 1
12 4 0 1505
25 3 1566 330
6 9 2917 0
|
```

UML-диаграмма предоставлена в отдельном файле.

Тестирование.

```
Player: position(23,1), maxHP:10000, current HP:1500, attack: 500
Sword: position(2,3), can heal:0, get attack: 1000
#####0#####
#           p           #
#           #           #
# !           #           #
#           #           #
#           f           #
#           #           #
#           @           #
#           #           #
#           #           #
#           s           #
#           #           #
#           U           #
# t           #           #
###$#####
Boss: position(2,13) current HP:6000, attack: 2500
```

Logger.txt – Блокнот

Файл Правка Формат Вид Справка

```
Player: position(23,1), maxHP:10000, current HP:1500, attack: 500
Sword: position(2,3), can heal:0, get attack: 1000
Boss: position(2,13) current HP:6000, attack: 2500
Boss: position(2,12) current HP:6000, attack: 2500
Boss: position(3,12) current HP:6000, attack: 2500
Boss: position(4,12) current HP:6000, attack: 2500
Boss: position(4,13) current HP:6000, attack: 2500
Boss: position(3,13) current HP:6000, attack: 2500
Boss: position(2,13) current HP:6000, attack: 2500
Boss: position(2,12) current HP:6000, attack: 2500
Boss: position(3,12) current HP:6000, attack: 2500
Boss: position(4,12) current HP:6000, attack: 2500
Boss: position(4,13) current HP:6000, attack: 2500
```

Итогом работы программы является игровое поле.

Выводы.

Были изучены принципы сериализации игры. В программу добавлен функционал сохранения текущего состояния игры.