**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**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0303 |  | Калмак Д.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2021

**Цель работы.**

Реализовать сохранение 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.**

Класс Fieldn содержит два поля: x – ширина поля, y – высота поля. В классе реализован конструктор, который заполняет поля x и y значениями.

Класс Celln необходим для сбора информации с клетки поля. Поля bool player; bool bolg; bool smaug; bool nazgul; bool medicine; bool weapon; bool armor; bool entrance; bool exit; bool wall; int hp, dmg, armor\_v; int value; принимают значения, когда конструктор принимает клетку и проверяет ее на информацию. В классе Cell добавлены проверки на наличие игрока, врага Bolg, врага Smaug, врага Nazgul, предмета Аптечки, предмета Оружия, предмета Брони, входа, выхода, стены. В классе Celln, если на клетке сущность, то собирается информация о здоровье, оружии, броне. Если на клетке предмет Аптечка, то собирается его емкость. Если на клетке предмет Оружие, то собирается его урон. Если на клетке предмет Броня, то собирается его прочность.

Класс Information содержит поле Field\* field – игровое поле, поле Celln\*\*\* celln – массив, который будет содержать информацию о клетках игрового поля, Fieldn\* fieldn – будет содержать информацию о размере поля, string c\_name – название файла для записи и чтения информации о клетках, string f\_name - название файла для записи и чтения информации о размерах поля. Конструктор класса Information принимает игровое поле. Метод writeCells() – записывает информацию о клетках в файл c\_name. Заполняется массив celln информацией о каждой клетке игрового поля. Информация о каждой клетке игрового поля записывается в файл. Метод readCells() – считывает информацию о клетках из файла. Если файл с сохранением не открылся, то обрабатывается ошибка. Массив celln заполняется информацией из файла. Метод writeField() – записывает информацию о размерах поля в файл f\_name. fieldn заполняется информацией о размерах поля. Информация о размерах поля записывается в файл. Метод readField() – считывает информацию о размерах поля из файла. Если файл с сохранением не открылся, то обрабатывается ошибка. fieldn заполняется информацией из файла.

В классе Game, если получена команда от контроллера о сохранении игры, то осуществляется запись информации о клетках и размерах поля в файлы с помощью методов writeCells() и writeField(). Если получена команда от контроллера о загрузке сохраненной игры, то считывается информация о клетках и размерах поля из файла. Метод load() – загружает сохранение игры. Builder с помощью информации о клетках и размерах поля и метода fill\_Field\_info() восстанавливает игровое поле. Перед восстановлением поля проходят проверки сохранения. Размер поля, наличие игрока на поле, возможность пройти игру. Если все верно, то игровое поле восстанавливается из сохранения и применяется в игре. Иначе обрабатывается ошибка. Если сохранение загружено, то переопределяются координаты игрока. Если обработаны ошибки, то игра продолжается на предыдущем поле.

Разработанную диаграмму классов UML см. в Приложении А.

**Выводы.**

Таким образом, реализовано сохранение игры.

**ПРИЛОЖЕНИЕ А UML ДИАГРАММА КЛАССОВ**

