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 _____ Кривенцова Л.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Изучить принцип сериализации, модифицировать программу так, чтобы она обрабатывала исключения.

Задание.

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

Выполнение работы.

Ход решения:

Для организации собственных исключений был создан класс *Except4Game.*

Определяются поля класса с модификатором доступа *private*:

- *ERROR type*; - переменная, хранящая элемент перечисления – тип ошибки.
- *std::string message*; - строка, хранящая текст сообщения об ошибке.

Поля инициализируются в конструкторе *Except4Game(ERROR tp):* значение, присваиваемое полю *type* передаётся методу в качестве аргумента, и в зависимости от типа, полю *message* присваивается определенная соответствующая строка.

Также в классе определен метод *std::string Except4Game::GetMessage()* – геттер для получения другими классами текст сообщения об ошибке.

В класс игры *Game* добавлены следующие методы с модификатором доступа *private*:

1. void *Saving()* – метод ничего не возвращает и не принимает в качестве аргумента. Предназначен для сохранения состояния игры в файл.

Считывает имя файла, в которое будет производиться запись. После записи через пробел габаритов поля, запускается вложенный цикл, который сохраняет состояние каждой клетки поля. После циклов записывается позиция игрока на поле.

Итоговая структура файла с сохранением имеет вид:

<Высота> <Ширина>

<Координата клетки x> <Координата клетки y> <Тип клетки> <Тип объекта, который хранит клетка> … <Особое поле>

…

<Координата клетки x> <Координата клетки y> <Тип клетки> <Тип объекта, который хранит клетка> … <Особое поле>

<Координата игрока x> <Координата игрока y> <Количество здоровье> <Наличие ключа> <Количество монет>

Особое поле (их может быть несколько) хранит дополнительную информацию в зависимости от объекта на клетке. Например, для врага это количество здоровья, а для сундука – наличие ключа.

2. *void LoadGame()*– метод ничего не возвращает и не принимает в качестве аргумента. Предназначен для загрузки состояния игры из файла.

Вначале с клавиатуры вводится название загрузочного файла (имя сохранения). Если файл с таким названием открыть не удалось, выбрасывается исключение (*throw Except4Game(FILE_ERROR*)). Далее, согласно структуре загрузочного файла происходит считывание параметров игры. Некоторые из них, имеющие ограничения (например, любые координаты должны быть больше либо равны нулю) проверяются на корректность, и в противном случае вызывается исключение (*throw Except4Game(BORDER_ERROR*)).

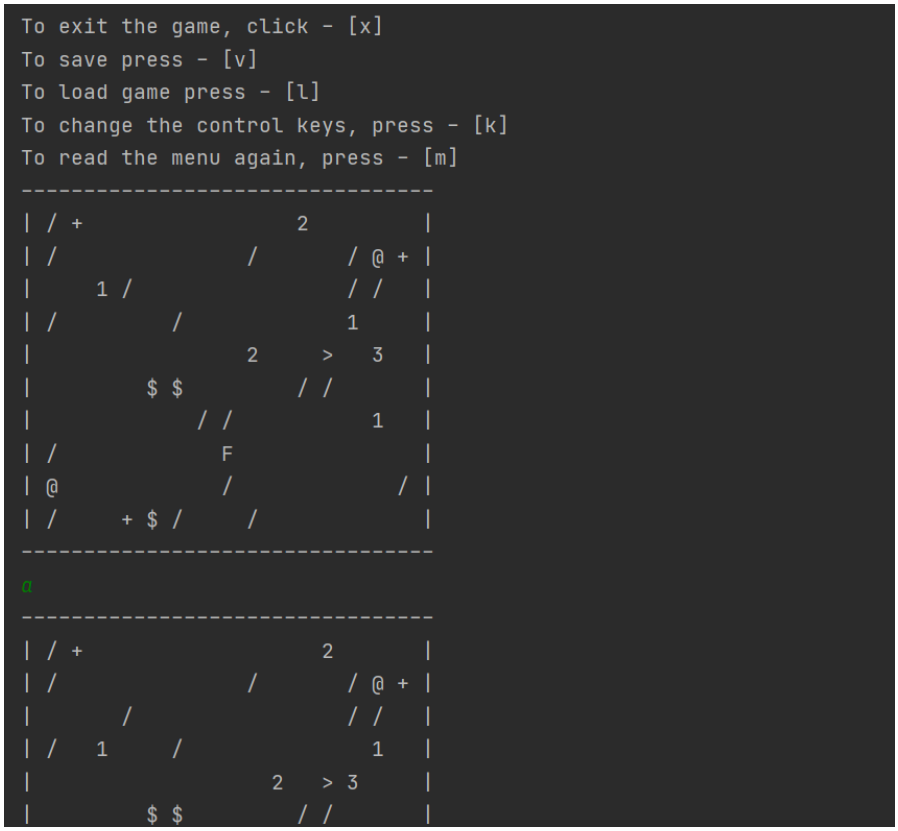
Так как каждый запуск игры ей выделяется разная область памяти, то нет смысла хранить адреса. Таким образом, на клетках создаются новые объекты (*game.GetField()[i][j].SetObj(new Box(bool(full)))*), полям которых через геттеры присваиваются соответствующие значения из файла.

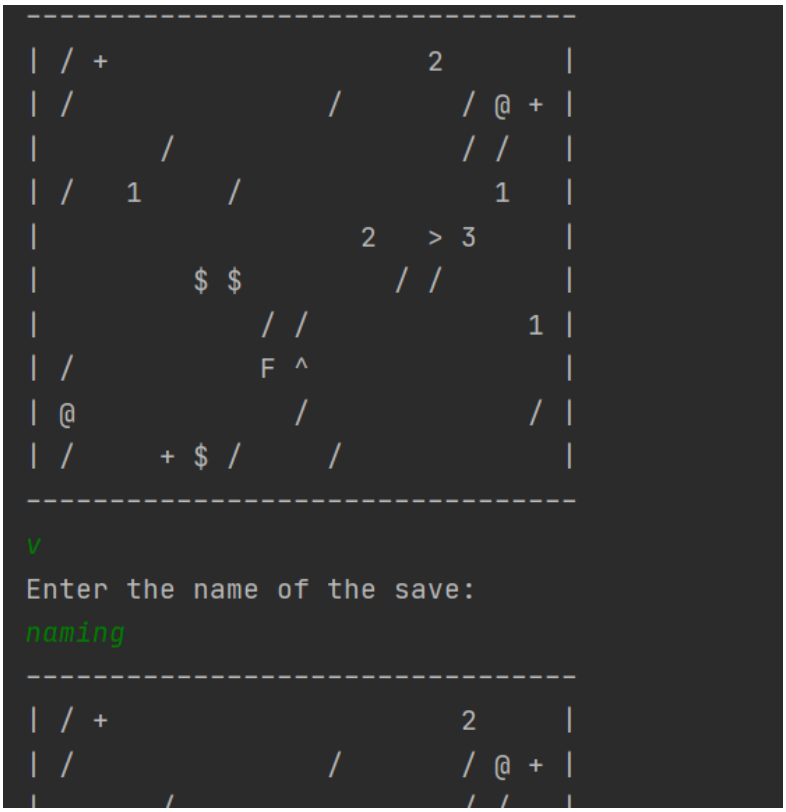
3. *void FileSafety(GO act).*

Описанные выше функции (сохранения и загрузке) в коде вызываются только через метод *FileSafety*, который отлавливает исключения, возникающие при работе с файлом с помощью *try – except*. Принимает на вход элемент перечисления действия игры. Если оно соответствует *SAVE* – происходит попытка вызвать функцию *Saving*, если *LOAD* – то вызывается *LoadGame*.

Результат работы программы:

Рис 1,2,3. – демонстрация начала игры.

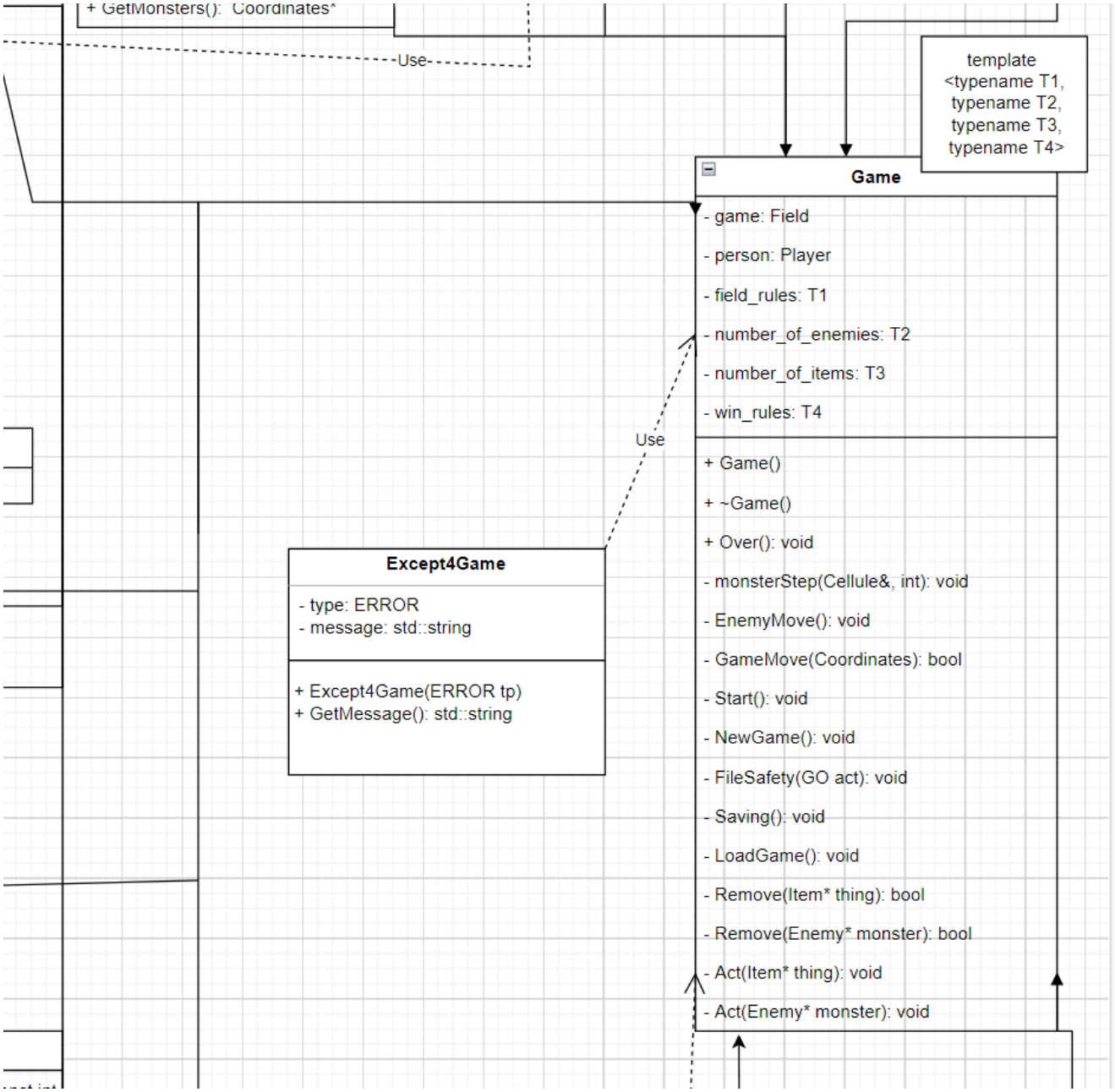



v
Enter the name of the save:
naming


l
Enter the name of the save:
naming1
x
See you later, player!
Enter the name of the save:
naming1

UML-диаграмма межклассовых отношений:

Рис 4. – UML-диаграмма классов.



Выводы.

Был изучен принцип сериализации, программа модифицирована таким образом, что обрабатываются исключения при попытке чтения некорректного файла.