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. 1384 _____ Белокобыльский И. В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2022

Цель работы.

Научиться реализовывать сериализацию классов и запись их состояния в файлы. На практике применить знания по обработке и генерации пользовательских исключений

Задание.

Реализовать систему классов позволяющих проводить сохранение и загрузку состояния игры. При загрузке должна соблюдаться транзакционность, то есть при неудачной загрузки, состояние игры не должно меняться. Покрыть программу обработкой исключительных состояний.

Требования:

Реализована загрузка и сохранение состояния игры

Сохранение и загрузка могут воспроизведены в любой момент работы программы.

Загрузка может произведена после закрытия и открытия программы.

Программа покрыта пользовательскими исключениями.

Пользовательские исключения должны хранить полезную информацию, например значения переменных при которых произошло исключение, а не просто сообщение об ошибке. Соответственно, сообщение об ошибке должно учитывать это поля, и выводить информацию с учетом значений полей.

Исключения при загрузке обеспечивают транзакционность.

Присутствует проверка на корректность файла сохранения. (Файл отсутствует; в файле некорректные данные, которые нарушают логику; файл был изменен, но данные корректны с точки зрения логики).

Примечания:

Исключения должны обрабатываться минимум на фрейм выше, где они были возбуждены

Для реализации сохранения и загрузки можно использовать мементо и посетителя

Для проверки файлов можно рассчитывать хэш от данных.

Выполнение работы.

Для реализации хранения данных об объекте был создан класс Snapshot, который считывает снимок из строки и записывает данные из него в std::map, а также из std::map генерирует строку.

Класс SnapshotsCollector контролирует объекты класса Snapshot. Он принимает на вход std::map состоящей из типов снимков и самих снимков. По определенному алгоритму он записывает содержание этих снимков в .env файл, а также реализует считывание из этих файлов. Для считывания он может принимать как конкретное название файла, так и любую другую строку для загрузки последнего сохранения.

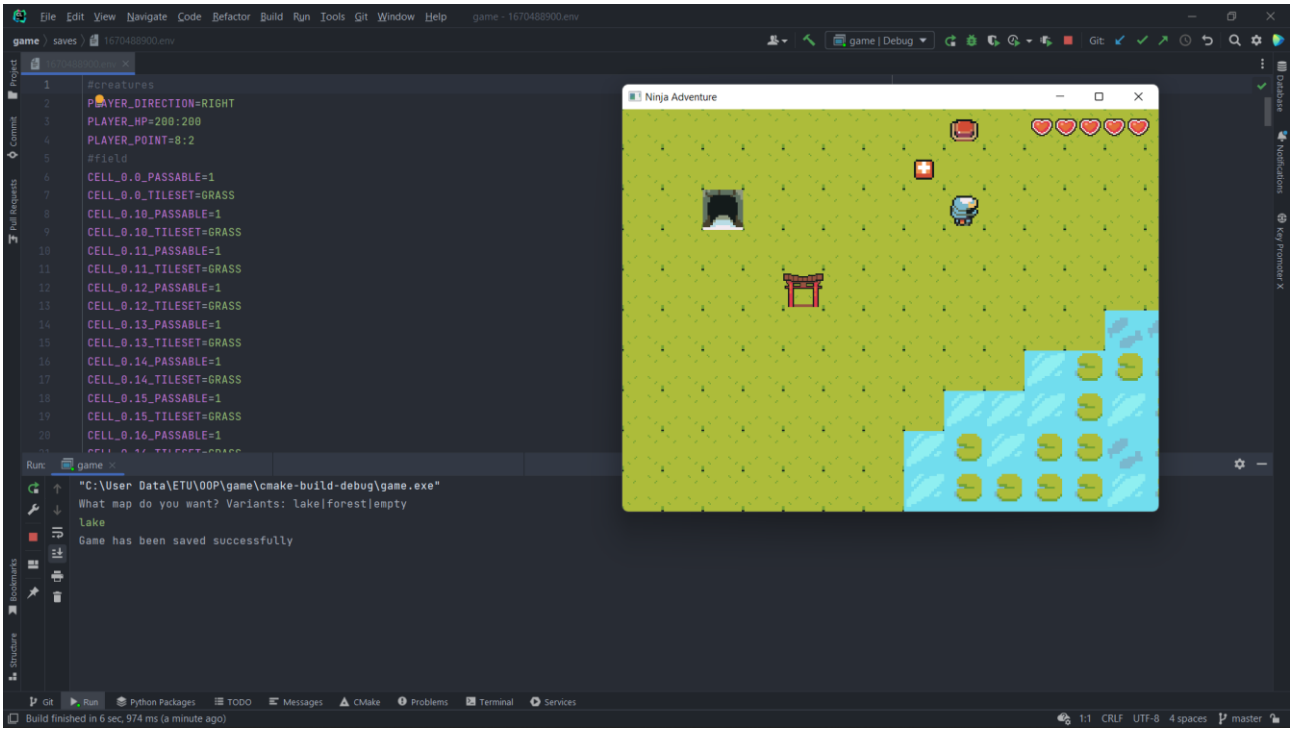
Для самой сериализации были созданы классы KernelSerializer и EventsSerializer. Первый сериализует поле и игрока, а также делегирует второму сериализацию событий. Сохраняются всевозможные состояния поля: ширина, высота, внешний вид, проходимость и события на клетках. Так как события реализованы в виде цепочки обязанностей, конкретные события записываются через «->», а внутри они разделяются «::» на фабрику и тип, который принимает фабрика для генерации события. Десериализация происходит аналогично: данные о поле и об игроке записываются KernelSerializer, а события делегируются EventsSerializer.

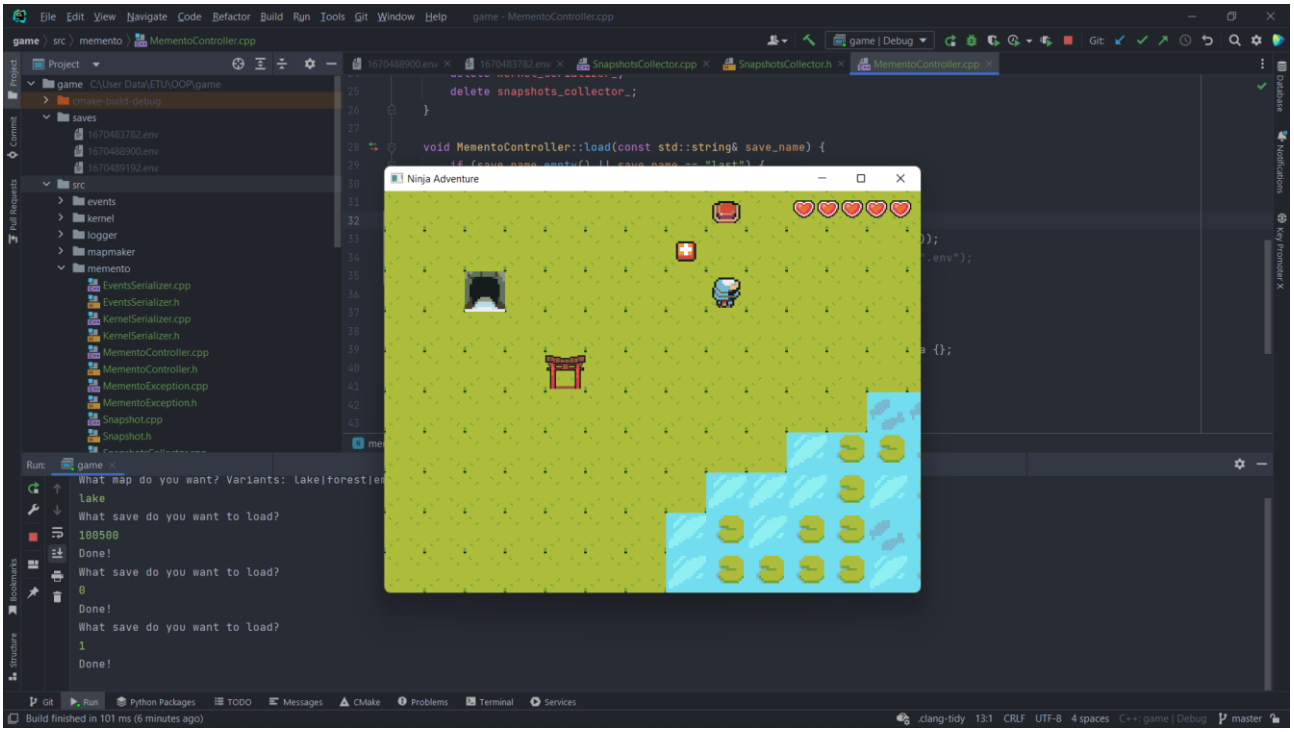
Контролирует поведение всех объектов класс MementoController, который реализует взаимодействие KernelSerializer и SnapshotsController: первый реализует принятие / применение состояний, а второй – запись в файл и получение данных из него.

Для генерации пользовательских ошибок был создан класс MementoException, который хранит в себе состояние модуля memento, во время которого произошла ошибка, а также саму ошибку.

Тестирование программы.

Сохранение и загрузка карты «Озеро»





Выводы.

В рамках данной лабораторной работы была доработана программа на языке C++ из предыдущей работы. Реализован алгоритм сериализации состояний игры, а также получение данных из файла и запись в него. Применены на практике механизмы реализации пользовательских исключений

ПРИЛОЖЕНИЕ 1

UML-диаграмма классов: