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

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Объектно-ориентированное программирование»

Тема: Связывание классов

Студент гр. 3385 _____ Хорчев Г.К.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы

Изучить связывание классов на языке C++. Реализовать классы игры и состояния игры. Реализовать механизм сохранения данных после завершения программы.

Задание

- Создать класс игры, который реализует следующий игровой цикл:

a. Начало игры
b. Раунд, в котором чередуются ходы пользователя и компьютерного врага. В свой ход пользователь может применить способность и выполняет атаку. Компьютерный враг только наносит атаку.
c. В случае проигрыша пользователь начинает новую игру
d. В случае победы в раунде, начинается следующий раунд, причем состояние поля и способностей пользователя переносятся.

- Класс игры должен содержать методы управления игрой, начало новой игры, выполнить ход, и т.д., чтобы в следующей лаб. работе можно было выполнять управление исходя из ввода игрока.
  - Реализовать класс состояния игры, и переопределить операторы ввода и вывода в поток для состояния игры. Реализовать сохранение и загрузку игры. Сохраняться и загружаться можно в любой момент, когда у пользователя приоритет в игре. Должна быть возможность загружать сохранение после перезапуска всей программы.
  - **Примечание:**
  - Класс игры может знать о игровых сущностях, но не наоборот
  - Игровые сущности не должны сами порождать объекты состояния
  - Для управления самое игрой можно использовать обертки над командами
  - При работе с файлом используйте идиому RAII.

Выполнение работы

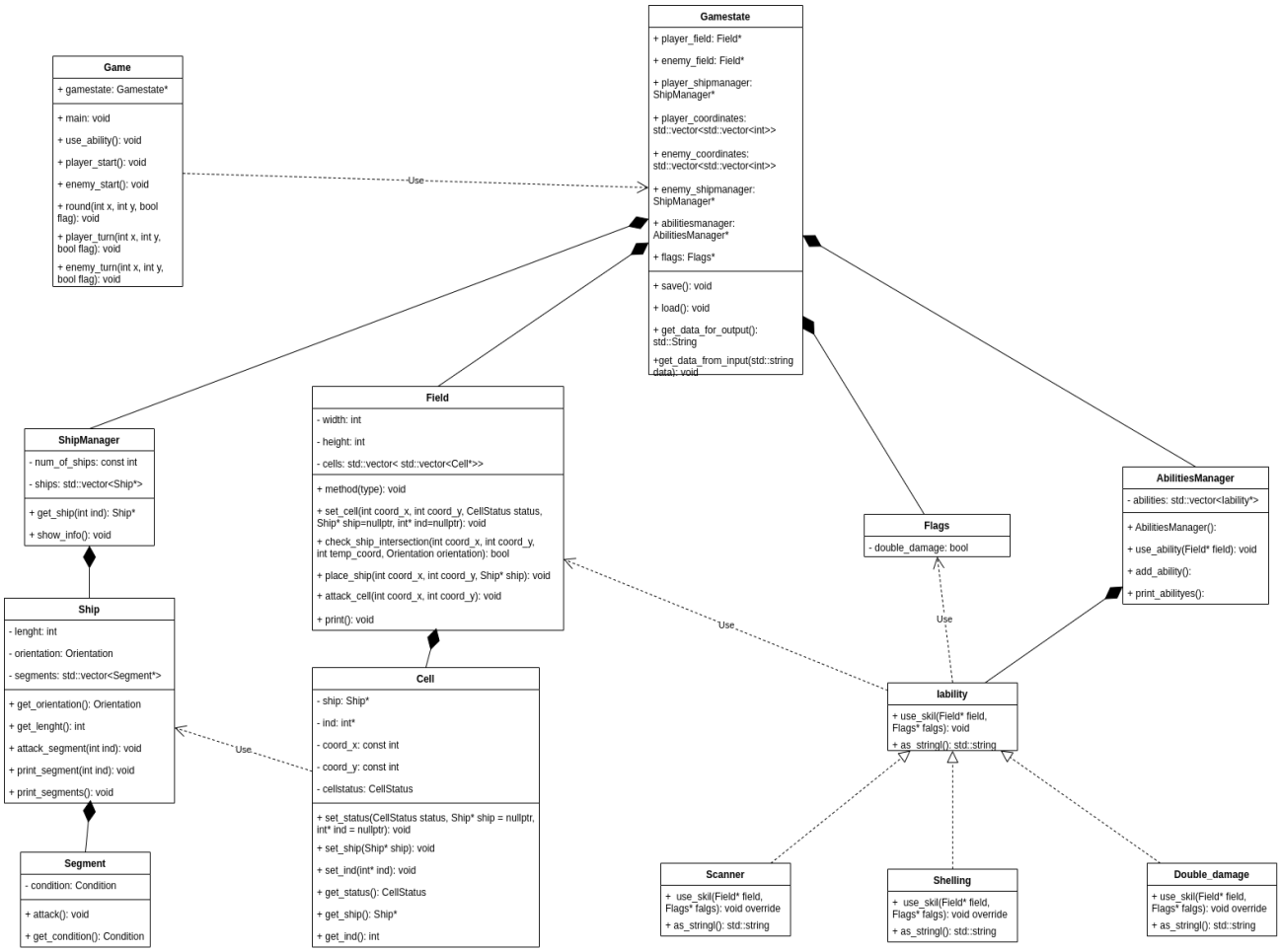

Рисунок 1 – UML-диаграмма классов

Класс Gamestate

Для реализации связывания создан класс **GameState**, который ответственен за хранение данных игры. В этом классе находятся соответствующие поля:

Field player_field;*
Field enemy_field;*
std::vector<std::vector<int>> player_ships_coordinates;
std::vector<std::vector<int>> enemy_ships_coordinates;
ShipManager player_shipmanager;*
ShipManager enemy_shipmanager;*
AbilitiesManager abilitiesmanager;*
Flags flags;*

На данном этапе ввод данных осуществляется через main, в последующем за ввод будет отвечать отдельный класс.

В классе реализован метод сохранения через запись данных в файл. Аналогично реализована загрузка сохранений. В файле также находится хеш-сумма, позволяющая предотвратить изменение сохранений пользователем. Данные методы - *void load(), void save().*

Класс Game.

Класс имеет всего одно поле - указать на состояние игры.

+ GameState gamestate*

Класс game оперирует с данными, хранящимися в gamestate. С помощью методотов:

void main();

void player_start();

void enemy_start();

void round(int coord_x, int coord_y, bool ability_flag=false);

Можно управлять игрой - расставить корабли игрока, противника, начать раунд. Для игры нужны данные от пользователя, на данном этапе они вводятся через cin, что будет далее заменено на специальный класс ввода данных.

В любой момент игрок может сохраниться или загрузиться (если файл с сохранением существует).

Основная логика игры реализована в методе void main(), который представляет собой бесконечный цикл - если проигрывает игрок, игра начинается заново, если проигрывает компьютер, игра продолжается.

Выводы

Было изучено связывание классов в C++, реализован механизм сохранения данных.