

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. 3342

Львов А.В.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2024

Цель работы

Разработать класс игры, который будет включать в себя весь игровой цикл.

Задание

Создать класс игры, который реализует следующий игровой цикл:

Начало игры

Раунд, в котором чередуются ходы пользователя и компьютерного врага. В свой ход пользователь может применить способность и выполняет атаку. Компьютерный враг только наносит атаку.

В случае проигрыша пользователь начинает новую игру

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 самой игрой можно использовать обертки над командами

При работе с файлом используйте идиому RAII.

Выполнение работы

Класс `Game` представляет собой класс игры, который реализует весь игровой цикл. Метод `NewRound` начинает первый раунд и затем проверяет состояние игры: если игрок выиграл в раунде, то начинается следующий с сохранением состояния игрока, в ином случае - игра начинается заново.

Метод `CycleGame` представляет собой непосредственно раунд, в котором чередуются ходы игрока и компьютера. С помощью методов `getCoordinates`, `getNumberShips`, `getLengths` считывают координаты, количество кораблей, их длины соответственно.

С помощью метода `attack` считываются координаты для атаки, и вызывается метод `attack` класса `Field`, указатель на который хранит в себе класс `GameState`. Если оказывается, что все корабли игрока или компьютера уничтожены, то игра заканчивается.

Класс `GameState` представляет собой некое хранилище для менеджеров и поля. В конструкторе создаются указатели на объекты данных классов. То есть, класс игры взаимодействует с этими сущностями посредством получения их через `GameState`.

Также `GameState` реализует методы для сохранения и загрузки состояния игры в файл.

Для сохранения и загрузки был создан класс `File`, который реализует идиому RAII.

Тестирование

Для проверки работоспособности программы был запущен код, и был сыгран один раунд.

```
===== Battleship =====
0 - New game
1 - Load game
0
Count of ships: 2
Ship sizes: 3 2
Next ability: Scanner
Введите координаты для корабля длины 3: 5 7
Выберите расположение корабля:
0 - Vertical
1 - Horizontal
1
  0 1 2 3 4 5 6 7 8 9
0 . . . . . . . . .
1 . . . . . . . . .
2 . . . . . . . . .
3 . . . . . . . . .
4 . . . . . . . . .
5 . . . . . . . . .
6 . . . . . . . . .
7 . . . . . S S S .
8 . . . . . . . . .
9 . . . . . . . . .

Введите координаты для корабля длины 2: 3 2
Выберите расположение корабля:
0 - Vertical
1 - Horizontal
0
```

Для наглядности поле компьютера было видимым.

```

===== Your field =====
  0 1 2 3 4 5 6 7 8 9
0  . . . . . . . . .
1  . . . . . . . . .
2  . . . S . . . . .
3  . . . S . . . . .
4  . . . . . . . . .
5  . . . . . . . . .
6  . . . . . . . . .
7  . . . . . S S S .
8  . . . . . . . . .
9  . . . . . . . . .

===== Computer's field =====
  0 1 2 3 4 5 6 7 8 9
0  . . . . . . . S S .
1  . . . . . . . . .
2  . . . . . . . . .
3  . . . . . . . . .
4  . . . . . . . . .
5  . . . . . . . . .
6  . . . . . . . . .
7  . . . . . . . . .
8  . . . . . . S S S
9  . . . . . . . . .

Computer attacks 7 0
Miss

```

Choose action:

- 0 - Save
- 1 - Load
- 2 - Attack
- 3 - Ability
- 4 - Exit

3

Enter coordinates for scanning: 7 0

There is a ship!

Enter coordinates for attack: 7 0

Attack ended!

	0	1	2	3	4	5	6	7	8	9
0	~	~	~	~	~	~	~	W	~	~
1	~	~	~	~	~	~	~	~	~	~
2	~	~	~	~	~	~	~	~	~	~
3	~	~	~	~	~	~	~	~	~	~
4	~	~	~	~	~	~	~	~	~	~
5	~	~	~	~	~	~	~	~	~	~
6	~	~	~	~	~	~	~	~	~	~
7	~	~	~	~	~	~	~	~	~	~
8	~	~	~	~	~	~	~	~	~	~
9	~	~	~	~	~	~	~	~	~	~

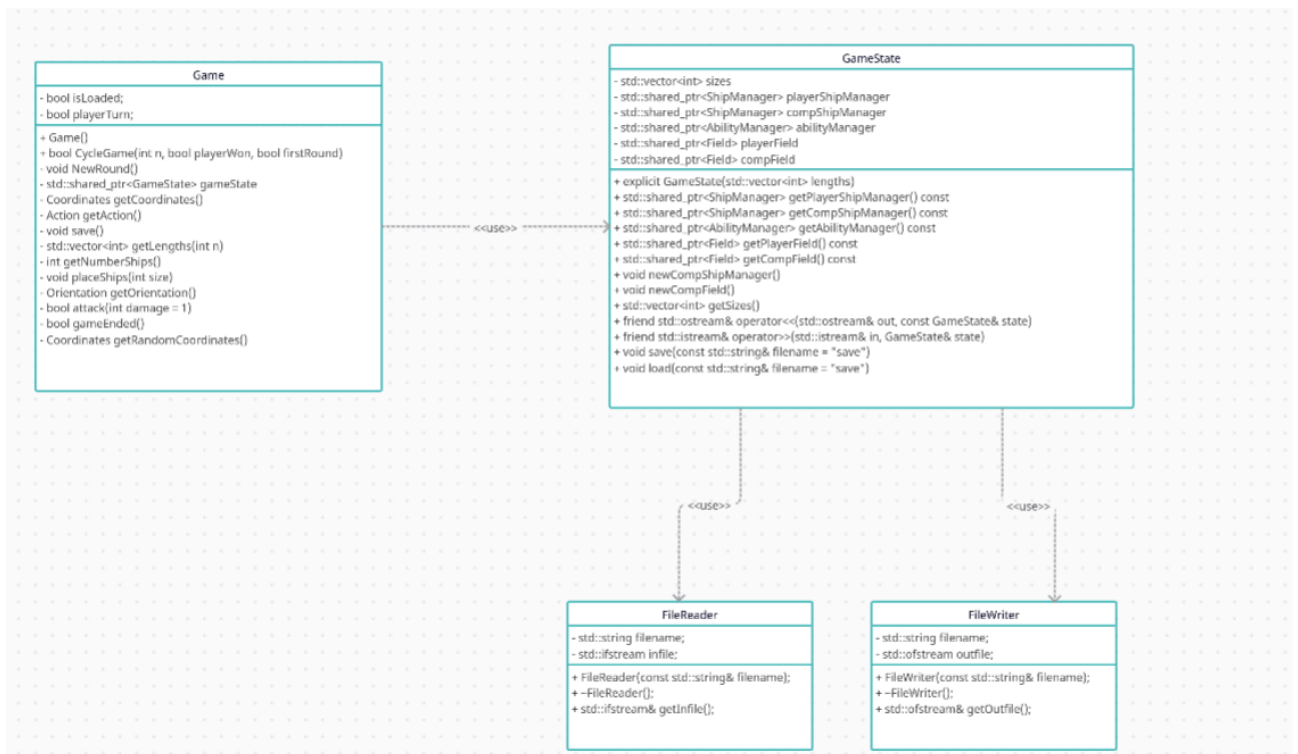
```

Double damage used!
Enter coordinates for attack: 9 8
Ability added!
Added new ability: Double damage
Attack ended!

  0 1 2 3 4 5 6 7 8 9
0 ~ ~ ~ ~ ~ ~ ~ X X ~
1 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
2 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
3 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
4 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
5 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
6 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
7 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~
8 ~ ~ ~ ~ ~ ~ ~ X X X
9 ~ ~ ~ ~ ~ ~ ~ ~ ~ ~

You won!
===== Computer's field =====
  0 1 2 3 4 5 6 7 8 9
0 . . . . . . . . . .
1 . . . S S S . . . .
2 . . . . . . . . . .
3 . . . . . . . . . .
4 . . . . . S . . . .
5 . . . . . S . . . .
6 . . . . . . . . . .
7 . . . . . . . . . .
8 . . . . . . . . . .

```

Выводы

В ходе выполнения лабораторной работы, был класс игры и класс состояния игры. Программа была успешно протестирована. Была реализована UML-диаграмма классов.