

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

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Объектно-ориентированное программирование»
Тема: Интерфейсы, динамический полиморфизм

Студентка гр. 1303

Андреева Е.А.

Преподаватель

Жангиров Т.Р.

Санкт-Петербург

2022

Цель работы.

Реализация базовых и дополнительных классов игры, интерфейсов, с прописанными в них конструкторами и методами для реализации логики игры.

Задание.

Реализовать систему событий. Событие - сущность, которая срабатывает при взаимодействии с игроком. Должен быть разработан класс интерфейс общий для всех событий, поддерживающий взаимодействие с игроком. Необходимо создать несколько групп разных событий реализуя унаследованные от интерфейса события (например, враг, который проверяет условие, будет ли воздействовать на игрока или нет; ловушка, которая безусловно воздействует на игрока; событие, которое меняет карту; и.т.д.). Для каждой группы реализовать конкретные события, которые по разному воздействуют на игрока (например, какое-то событие заставляет передвинуться игрока в определенную сторону, а другое меняет характеристики игрока). Также, необходимо предусмотреть событие “Победа/Выход”, которое срабатывает при соблюдении определенного набора условий.

Реализовать ситуацию проигрыша (например, потери всего здоровья игрока) и выигрыша игрока (добрался и активировал событие “Победа/Выход”)

Требования:

- Разработан интерфейс события с необходимым описанием методов
- Реализовано минимум 2 группы событий (2 абстрактных класса наследников события)
- Для каждой группы реализовано минимум 2 конкретных события (наследники от группы события)
- Реализовано минимум одно условное и безусловное событие (условное - проверяет выполнение условий, безусловное - не проверяет).
- Реализовано минимум одно событие, которое меняет карту (меняет события на клетках или открывает расположение выхода или делает

какие-то клетки проходимыми (на них необходимо добавить события) или не непроходимыми

- Игрок в гарантированно имеет возможность дойти до выхода

Примечания:

- Классы событий не должны хранить никакой информации о типе события (никаких переменных и функций дающие информации о типе события)
- Для создания события можно применять абстрактную фабрику/прототип/строитель

Выполнение работы.

В программе были реализованы следующие классы: *Event*, *GoodEvents*, *BadEvents*, *Enemy*, *Trap*, *Food*, *Exit*, *Money*, *GoodCell*, *Factory*, *EnemyFactory*, *TrapFactory*, *MoneyFactory*, *FoodFactory*, *GoodCellFactory*, *ExitFactory*.

Класс *Event* (является интерфейсом), в нем определена виртуальная функция *playerReact()*, с помощью нее далее будет реализована реакция игрока на событие в клетке.

От класса *Event* наследуются 2 абстрактных класса *GoodEvents* и *BadEvents*. В классе *GoodEvents* определена функция *addGood()*, с помощью которой игроку будет что-то добавляться. Функция далее используется в конкретных классах «хороших» событий. Также реализована функция *addPoints()*, которая добавляет игроку очки, если он наступил на «хорошую» клетку. В классе *BadEvents* определена функция *remBad()*, с помощью которой у игрока будет что-то отниматься. Функция далее используется в конкретных классах «плохих» событий.. Также реализована функция *remPoints()*, которая отнимает очки у игрока, если он наступил на «плохую» клетку.

Далее были созданы классы конкретных событий (*Enemy*, *Trap*, *Food*, *Exit*, *Money*, *GoodCell*), в каждом из них реализована функция *playerReact()* по своему:

Класс *Enemy* – «враг», отнимает у игрока защиту и здоровье. Также в методе проверяется, достаточно ли у игрока защиты, здоровья и очков, чтобы продолжить игру. Если нет, то у игрока меняется поле *dead* на значение *true*.

Класс *Trap* – «ловушка», отнимает у игрока здоровье. Также в методе проверяется, достаточно ли у игрока защиты, здоровья и очков, чтобы продолжить игру. Если нет, то у игрока меняется поле *dead* на значение *true*.

Класс *Food* – «еда», добавляет игроку здоровье. Также в методе проверяется, достаточно ли у игрока очков для победы. Если да, то у игрока вызывается метод *setOpenExit()*, который открывает расположение выхода.

Класс *Money* – «монеты», добавляет игроку очки. Также в методе проверяется, достаточно ли у игрока очков для победы. Если да, то у игрока вызывается метод *setOpenExit()*, который открывает расположение выхода.

Класс *Exit* – «выход», меняет поле игрока *win* на *true*, то есть присваивает игроку победу.

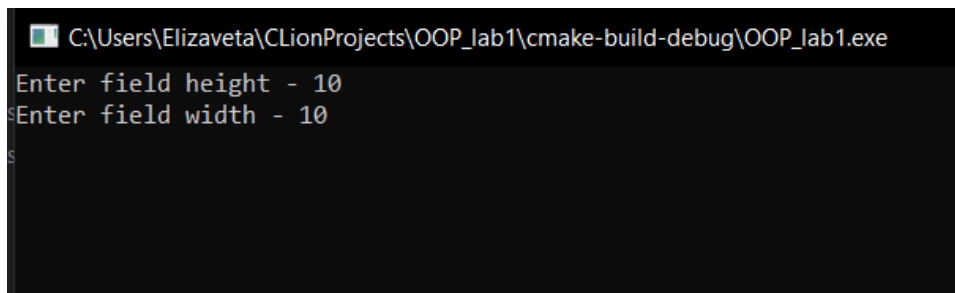
Класс *GoodCell* – обычные клетки, которые ничего не делают.

Для создания объектов классов событий был создан класс *Factory* и классы наследники от *Factory* – *FoodFactory*, *MoneyFactory*, *ExitFactory*, *GoodCellFactory*, *EnemyFactory*, *TrapFactory*. В классе *Factory* определена функция *createEvent()*, которая далее реализуется в классах наследниках и которая позволяет создать объекты классов конкретных событий. Абстрактная фабрика задаёт интерфейс создания всех доступных типов событий.

В классе *PrintCell* был добавлен вывод клеток, содержащих определенное событие.

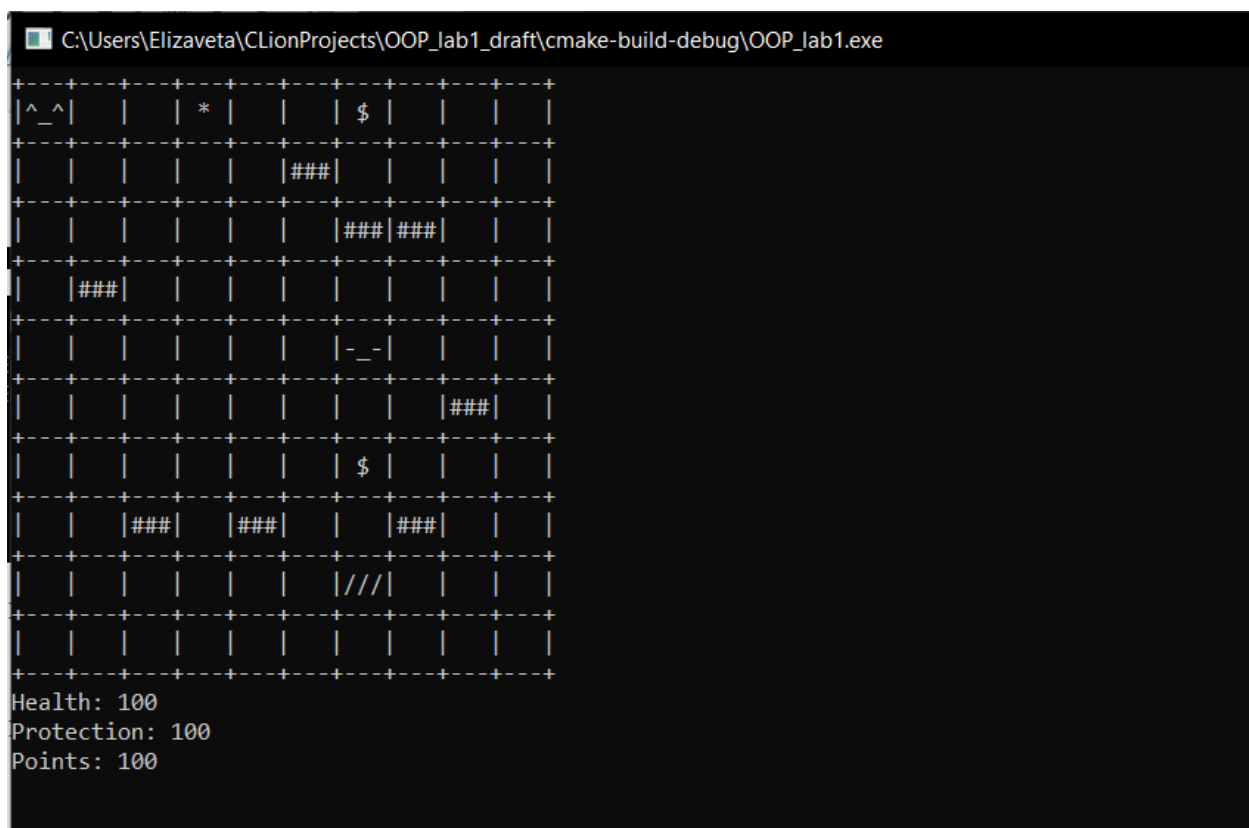
В классе *Field* также был реализован метод *setFieldEvents()*, который расставляет события на клетки.

Проверка работоспособности программы.



```
C:\Users\Elizaveta\CLionProjects\OOP_lab1\cmake-build-debug\OOP_lab1.exe
Enter field height - 10
Enter field width - 10
```

Рис. 1 Ввод размеров поля.



```
C:\Users\Elizaveta\CLionProjects\OOP_lab1_draft\cmake-build-debug\OOP_lab1.exe
+---+---+---+---+---+---+---+---+---+---+
| ^_^ | | | * | | | $ | | | |
+---+---+---+---+---+---+---+---+---+---+
| | | | | | ### | | | |
+---+---+---+---+---+---+---+---+---+---+
| | | | | | ### ### | | |
+---+---+---+---+---+---+---+---+---+---+
| ### | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
| | | | | | -_- | | | |
+---+---+---+---+---+---+---+---+---+---+
| | | | | | | | ### |
+---+---+---+---+---+---+---+---+---+---+
| | | | | | | $ | | | |
+---+---+---+---+---+---+---+---+---+---+
| | | ### | ### | | ### | |
+---+---+---+---+---+---+---+---+---+---+
| | | | | | ||| | | |
+---+---+---+---+---+---+---+---+---+---+
| | | | | | | | | |
+---+---+---+---+---+---+---+---+---+---+
Health: 100
Protection: 100
Points: 100
```

Рис. 2 Вывод поля соответствующих размеров

```
C:\Users\Elizaveta\CLionProjects\OOP_lab1_draft\cmake-build-debug\OOP_lab1.exe

+---+---+---+---+---+---+---+---+---+
|   |   |   |   | * |   |   | $ |   |   |
+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | ### |   |   |
+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | ### | ### |   |
+---+---+---+---+---+---+---+---+---+
|   | ### |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | ^_^ |   |   |
+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   | ### |
+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   | $ |   |   |
+---+---+---+---+---+---+---+---+---+
|   |   | ### |   | ### |   |   | ### |
+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   | /// |   |   |
+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+
Health: 90
Protection: 90
Points: 100
```

Рис.3 Игрок попал на клетку с врагом.

UML-диаграмма межклассовых отношений:

