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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Объектно-ориентированое программирование»
Тема: Создание классов, конструкторов и методов.

Студент гр. 1382 _____ Исайкин Г. И.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2022

Цель работы.

Изучить понятия класса, его методов и полей, научиться реализовывать простейшие классы и осуществлять межклассовые отношения.

Задание.

Реализовать прямоугольное игровое поле, состоящее из клеток. Клетка - элемент поля, которая может быть проходима или нет (определяет, куда может стать игрок), а также содержит какое-либо событие, которое срабатывает, когда игрок становится на клетку. Для игрового поля при создании должна быть возможность установить размер (количество клеток по вертикали и горизонтали). Игровое поле должно быть зациклено по вертикали и горизонтали, то есть если игрок находится на правой границе и идет вправо, то он оказывается на левой границе (аналогично для всех краев поля).

Реализовать класс игрока. Игрок - сущность контролируемая пользователем. Игрок должен иметь свой набор характеристик и различный набор действий (например, разные способы перемещения, попытка избежать событие, и так далее).

Требования:

- Реализован класс игрового поля
- Для игрового поля реализован конструктор с возможностью задать размер и конструктор по умолчанию (то есть конструктор, который можно вызвать без аргументов)
- Реализован класс интерфейс события (в данной лабораторной это может быть пустой абстрактный класс)
- Реализован класс клетки с конструктором, позволяющим задать ей начальные параметры.
- Для клетки реализованы методы реагирования на то, что игрок перешел на клетку.

- Для клетки реализованы методы, позволяющие заменять событие. (То есть клетка в ходе игры может динамически меняться)
- Реализованы конструкторы копирования и перемещения, и соответствующие им операторы присваивания для игрового поля и при необходимости клетки
- Реализован класс игрока минимум с 3 характеристиками. И соответствующие ему конструкторы.
- Реализовано перемещение игрока по полю с проверкой допустимости на переход по клеткам.

Примечания:

- При написании конструкторов учитывайте, что события должны храниться по указателю для соблюдения полиморфизма
- Для управления игроком можно использовать медиатор, команду, цепочку обязанностей

Выполнение работы.

Для выполнения работы было сделано 6 классов:

- Cell. Класс является реализацией клетки игрового поля. Он имеет 3 поля в доступе private и 12 методов в доступе public. Начнём с полей. Первое поле - CellEvent *event. Оно хранит событие, которое произойдёт тогда, когда Игрок ступит на эту клетку. Второе поле - bool passable. Оно хранит информацию, является ли поле проходимым. Третье поле — bool busy. Даёт информацию, не стоит ли здесь игрок. Среди всех методов есть четыре конструктор — Cell(), Cell(bool pass), Cell(const Cell& c) и Cell(Cell&& c), и два оператора присваивания — Cell& operator=(const Cell& c) и Cell& operator=(Cell&& c). Другие методы: CellEvent *get_event() - получить значение поля event, void event_change(CellEvent* new_event) — поменять значение поля event,

bool is_passable() и bool is_busy() - получить значения passable и busy соответственно и void set_passable(bool new_passable) и set_busy(bool new_busy) — устанавливают значения passable и busy соответственно.

- Field. Это класс, который описывает игровое поле. Он состоит из 3 полей с доступом private и из 10 методов доступа public. Поля — это unsigned int width, unsigned int length и std::vector <std::vector <Cell>> map, отвечающие соответственно ширине и длине карты и двойному вектору её клеток. Среди метдов 4 конструктора - Field(unsigned int w, unsigned int l), Field(), Field(const Field& f) и Field(Field&& f), и 2 оператора присваения Field& operator=(const Field& f) и Field& operator=(const Field& f). Методы Cell *get_cell(long int x, long int y) и Cell *get_cell(std::pair <long int, long int> coord) возвращают указатель на клетку под соответствующими координатами. Методы unsigned int get_length() и unsigned int get_width() возвращают длину и ширину соответсвенно.
- Player. Это класс игрока. В нём есть 8 полей: Field *field — указатель на игровое поле, int hit_point — количество единиц здоровья, int strength — сила игрока, int endurance — выносливость игрока, int dexterity — ловкость игрока, int armor — броня игрока, unsigned int coins — количество монет у игрока, std::pair <long int, long int> coord — координаты, которые указывают на местоположения игрока. Для полей hit_point, strength, endurance, dexterity, armor и coins есть по три метода на каждый: для установление нового значения (set_...(value)), для прибавление/убавлении от текущего (add_...(value)) и для получения значения (get_...()). Есть 6 конструкторов Player(Field *f, int h, int s, int e, int d, int a, unsigned int c), Player(Field *f), Player(int h, int s, int e, int d, int a, unsigned int c), Player(), Player(const Player& p) и Player(Player&& p) и 2 оператора присваивания Player& operator=(const Player& p) и Player& operator=(Player&& p). void event_trigger() - метод,

приводящий событие клетки, на которой стоит игрок, в действие. bool move(direction d) — метод, двигающий игрока на соседнюю клетку соответствуя напрвлению (direction d — один из 8 направлений). Если у нужной клетки значение passable истина, игрок передвигается на неё и метод возращает true, если нет, то false. void set_field(Field *new_field, bool save_Y_on_old_field) — метод меняющий поле field. Переменная save_Y_on_old_field отвечает, нужно ли сохранять значение busy = true на клетки старого поля или нет. void set_coord(std::pair <long int, long int> new_coord) — метод, перемещающий игрока на новые координаты. std::pair <long int, long int> get_coord() - возрощяет координаты игрока. Field *get_field() возвращает поле field.

- CellView. Класс с 1 методом — char cell_view(Cell *c). Он возвращает буквенное представление клетки поля.
- FieldView. У класса 3 поля - unsigned int width, unsigned int length и std::vector< std::vector< char >> map — длина, ширина и двойной вектор символов, а именно буквенных представлений клеток игрового поля. В классе есть 3 метода: void set_field(Field *field) — заполняет двойной вектор map, std::vector< std::vector< char >> &get_field() - возвращает ссылку на двойной вектор, и void print() - печатает двойной вектор виде матрицы в консоль.

  UML-диаграмму межклассовых отношений см. в приложении А.

Выводы.

Были изучены понятия класса, его методов и полей, получены навыки реализовывать простейшие классы и осуществлять межклассовые отношения.

ПРИЛОЖЕНИЕ А

UML-ДИАГРАММА МЕЖКЛАССОВЫХ ОТНОШЕНИЙ