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

ОТЧЕТ

по лабораторной работе №1
по дисциплине «Объектно – ориентированное программирование»
Тема: Создание классов, конструкторов классов, методов классов, наследование

Студент гр. 8381 _____ Киреев К.А.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург
2020

Задание.

Разработать и реализовать набор классов:

- Класс игрового поля
- Набор классов юнитов

Игровое поле является контейнером для объектов представляющим прямоугольную сетку. Основные требования к классу игрового поля:

- Создание поля произвольного размера
- Контроль максимального количества объектов на поле
- Возможность добавления и удаления объектов на поле
- Возможность копирования поля (включая объекты на нем)
- Для хранения запрещается использовать контейнеры из stl

Юнит является объектов, размещаемым на поля боя. Один юнит представляет собой отряд. Основные требования к классам юнитов:

- Все юниты должны иметь как минимум один общий интерфейс
- Реализованы 3 типа юнитов (например, пехота, лучники, конница)
- Реализованы 2 вида юнитов для каждого типа (например, для пехоты могут быть созданы мечники и копейщики)
- Юниты имеют характеристики, отражающие их основные атрибуты, такие как здоровье, броня, атака.
- Юнит имеет возможность перемещаться по карте

Выполнение работы

Для выполнения лабораторной работы были созданы следующие классы:

Field

Это основной класс, представляющий из себя игровое поле. Для этого класса были реализованы конструкторы копирования и переноса. Реализовано создание поля произвольного размера. Реализован контроль максимального количества юнитов на поле. Реализованы методы добавления, удаления и

передвижения объектов на поле. Все основные методы класса приведены в таблице 1.

Таблица 1 – Основные методы класса Field

Метод	Назначение
Field(unsigned int NmRows, unsigned int NmColumns, unsigned int MaxObjects);	Конструктор класса, позволяющий создавать поле произвольного размера.
Field(const Field& field);	Конструктор копирования
Field(Field&& field)	Конструктор переноса
void setMaxObjects (unsigned int maxObjectsNewValue);	Устанавливает максимальное количество объектов на поле.
void addObject(unsigned int Nrows, unsigned int Ncols, IFieldComponent* object);	Добавляет объект на поле
void removeObject(unsigned int Nrows, unsigned int Ncols);	Удаляет объект с поля
void moveObject(unsigned int NrowsFrom, unsigned int NcolFrom, unsigned int NrowsTo, unsigned int NcolTo);	Двигает объект на поле
cellInterface* getObject(unsigned int rowNumber, unsigned int columnNumber);	Получает указатель на объект в определённой ячейке поля.
QString getStringInfo();	Получает поле в виде информации о нём, содержащую строку. (Используется для демонстрации работы программы).

Все методы класса сохраняют его инвариант – количество строк/столбцов является беззнаковым числом (большим, либо равным 0), никакой объект не может находиться вне размера поля.

cellInterface

Этот класс представляет собой интерфейс для клетки поля. Основные его методы приведены в таблице 2.

Таблица 2 – Основные методы графики класса cellInterface

Метод	Назначение
virtual bool *isMove*() const = 0;	Возвращает true, если объект можно передвигать, и false в ином случае.
virtual cellInterface* *copy*() = 0;	Копирует объект на поле и возвращает указатель на скопированный объект (используется в конструкторе копирования поля).

printInterface

Этот класс необходим для демонстрации работы программы. Определяет метод, который помогает узнать информацию о классе. Подробнее информация приведена в таблице 3.

Таблица 3 – Основные методы графики класса printInterface

Метод	Назначение
virtual QString *getClass*() = 0;	Возвращает имя класса в виде строки.

UnitInterface

Этот класс является базовым для всех юнитов. Определяет интерфейс, присущий каждому юниту: показатели брони, здоровья, урона. Подробная информация о методах в таблице 4.

Таблица 4 – Основные методы графики класса UnitInterface

Метод	Назначение
bool *isMove*() const override { return true; }	Переопределяет метод isMove класса-родителя cellInterface. Возвращает true.
void setAllAttribues(int health, int armor, int damage);	Устанавливает значения всех атрибутов у юнита.

InfantryInterface

Это класс, определяющий интерфейс для пехоты. Не переопределяет и не реализует никаких методов.

CavalryInterface

Это класс, определяющий интерфейс для кавалерии. Не переопределяет и не реализует никаких методов.

ArcherInterface

Это класс, определяющий интерфейс для лучников. Не переопределяет и не реализует никаких методов.

UnitCritInterface

Это класс, определяющий интерфейс для юнитов, наносящих критический урон. Имеет показатель critFactor множителя критического урона.

UnitDamageAbsorberInterface

Это класс, определяющий интерфейс для юнитов, которые способны поглощать часть урона. Имеет показатель absorptionFactor коэффициента поглощения.

DamageAbsorberInfantry

Класс пехоты, способной поглощать урон.

CritInfantry

Класс пехоты, способной наносить критический урон.

DamageAbsorberCavalry

Класс кавалерии, способной поглощать урон.

CritCavalry

Класс кавалерии, способной наносить критический урон.

DamageAbsorberArcher

Класс лучника, способного поглощать урон.

CritArcher

Класс лучника, способного наносить критический урон.

Эти классы переопределяют методы getClass() и copy() их родителей.

abstractFactory

Это интерфейс абстрактной фабрики. Основные методы описаны в таблице 5.

Таблица 5 – Основные методы графики класса abstractFactory

Метод	Назначение
virtual InfantryInterface* *createInfantry*() const= 0;	Создаёт объект пехоты и возвращает указатель на него.
virtual CavalryInterface* *createCavalry*() const= 0;	Создаёт объект кавалерии и возвращает указатель на него.
virtual ArcherInterface* createArcher() const = 0;	Создаёт объект лучника и возвращает указатель на него.

CritUnitsFactory

Это класс конкретной фабрики, поставляющей юнитов со способностью нанесения критического урона. Основные методы приведены в таблице 6.

Таблица 6 – Основные методы графики класса CritUnitsFactory

Метод	Назначение
virtual InfantryInterface * *createInfantry*() const override	Создаёт объект CritInfantry и возвращает указатель на него.
virtual CavalryInterface * *createCavalry*() const override	Создаёт объект CritCavalry и возвращает указатель на него.
virtual ArcherInterface * createArcher() const override	Создаёт объект CritArcher и возвращает указатель на него.

DamageAbsorbersFactory

Это класс конкретной фабрики, поставляющей юнитов со способностью поглощения урона. Основные методы приведены в таблице 7.

Таблица 7 – Основные методы графики класса DamageAbsorbersFactory

Метод	Назначение
virtual InfantryInterface * *createInfantry*() const override	Создаёт объект DamageAbsorberInfantry и возвращает указатель на него.
virtual CavalryInterface * *createCavalry*() const override	Создаёт объект DamageAbsorberCavalry и возвращает указатель на него.
virtual ArcherInterface * createArcher() const override	Создаёт объект DamageAbsorberArcher и возвращает указатель на него.

Тестирование программы.

Для демонстрации работы программы были разработаны 3 примера.

Пример 1.

В этом примере происходит создание поля размером 5x5 с максимальным числом объектов на нем 5. На это поле добавляется лучник со способностью критического урона и поглощающая урон кавалерия.

Результат работы примера приведён на рисунке 1.

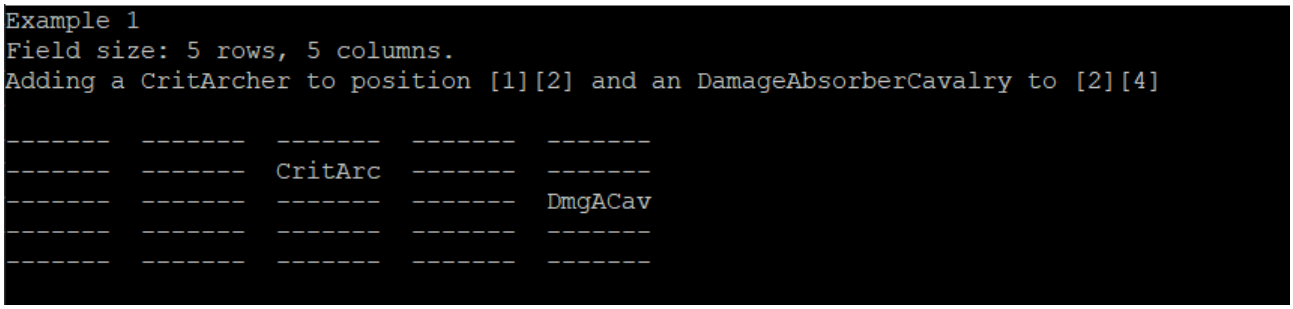

Рисунок 1 - пример 1

Пример 2.

В этом примере происходит создание поля размером 3x4 с максимальным числом объектов 4. На поле добавляется пехота со способностью критического урона, а затем перемещается на другое место.

Результат работы примера приведён на рисунке 2.

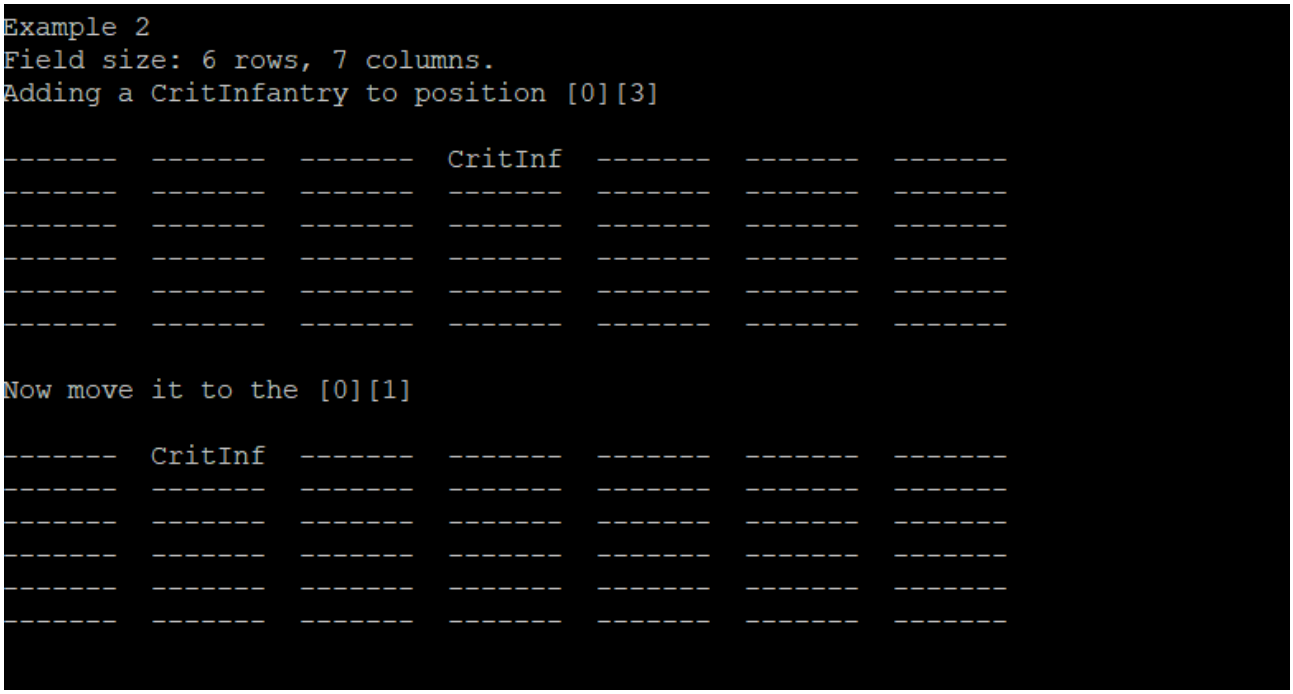

Рисунок 2 - пример 2

Пример 3.

В этом примере происходит создание поля размера 5x3 с максимальным числом объектов 6. Далее, на поле добавляется поглощающая урон пехота. После этого, этот объект пехоты удаляется с поля.

Результат работы примера приведён на рисунке 3.

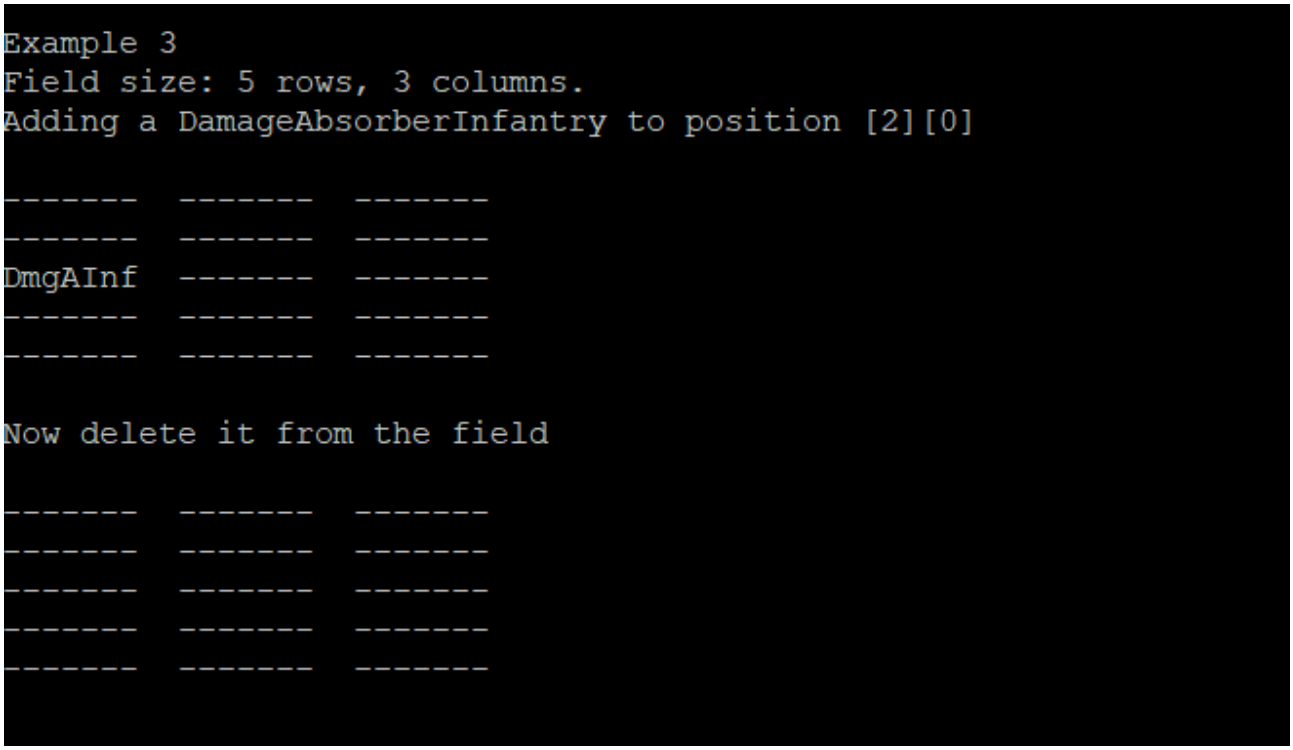

Рисунок 3 - пример 3

Выводы.

В ходе выполнения лабораторной работы была написана программа, в которой были реализованы классы поля, интерфейсы юнитов, абстрактной фабрики юнитов, итератор, конкретные классы юнитов, интерфейс клетки поля и разработаны 3 демонстрационных примера.