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

ОТЧЕТ

по практической работе №1

по дисциплине «Объектно-ориентированное программирование»

Тема: Создание классов, конструкторов классов, методов классов. Наследование

Студент гр. 8383 _____ Федоров И.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы.

Реализовать и разработать начальный набор классов.

Постановка задачи.

- 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.

Был создан абстрактный класс `IUnit`, хранящий в себе методы, присущие всем юнитам. Был создан класс `GameObj`, который хранит в себе характеристики и методы, присущие всем объектам. Для

Класс `Unit` унаследован от интерфейса `IUnit` и `GameObj`. От класс `Unit` унаследованы 3 абстрактных класса: `TBattleCharacter`,

TMagicCharacter, TSiegiCharacter. Каждый из этих абстрактных класс отвечает за свой род войск, от каждого из них унаследованы по два "реальных" класса воинов двух расс.

Был создан класс **GameField**, отвечающий за хранение поля, хранит в себе матрицу клеток, информацию о себе (размер, число объектов на поле). Помимо методов инициализации, удаления и добавления объекта на поле имеет метод move, который предназначен для передвижения юнита (класс Unit). Unit в своем методе move обращается к полю, передавая ему координаты назначения, поле же является friend-дом для Unit, поэтому имеет доступ к его координатам. Поле в своем методе move решает, может ли юнит переместиться в указанный квадрат, не давая тем самым переходить юниту за пределы поля, а так же на занятые клетки. Для класс GameField был реализован конструктор копирования GameField(const GameField& other);

Для создания юнитов использовался паттерн "абстрактная фабрика". Был создан интерфейс AFactory, имеющий методы по созданию трех видом юнитов: TBattleCharacter, TMagicCharacter, TSiegiCharacter. От этого интерфейса наследуются два класса: HumanFactory и ChaosFactory, соответственно создающие юнитов своей рассы. Благодаря тому, что все "физические юниты" наследованы от абстрактных классов-типов войск, возможно реализовать данные фабрики.

Полная uml-диаграмма представлена на рис. 1.

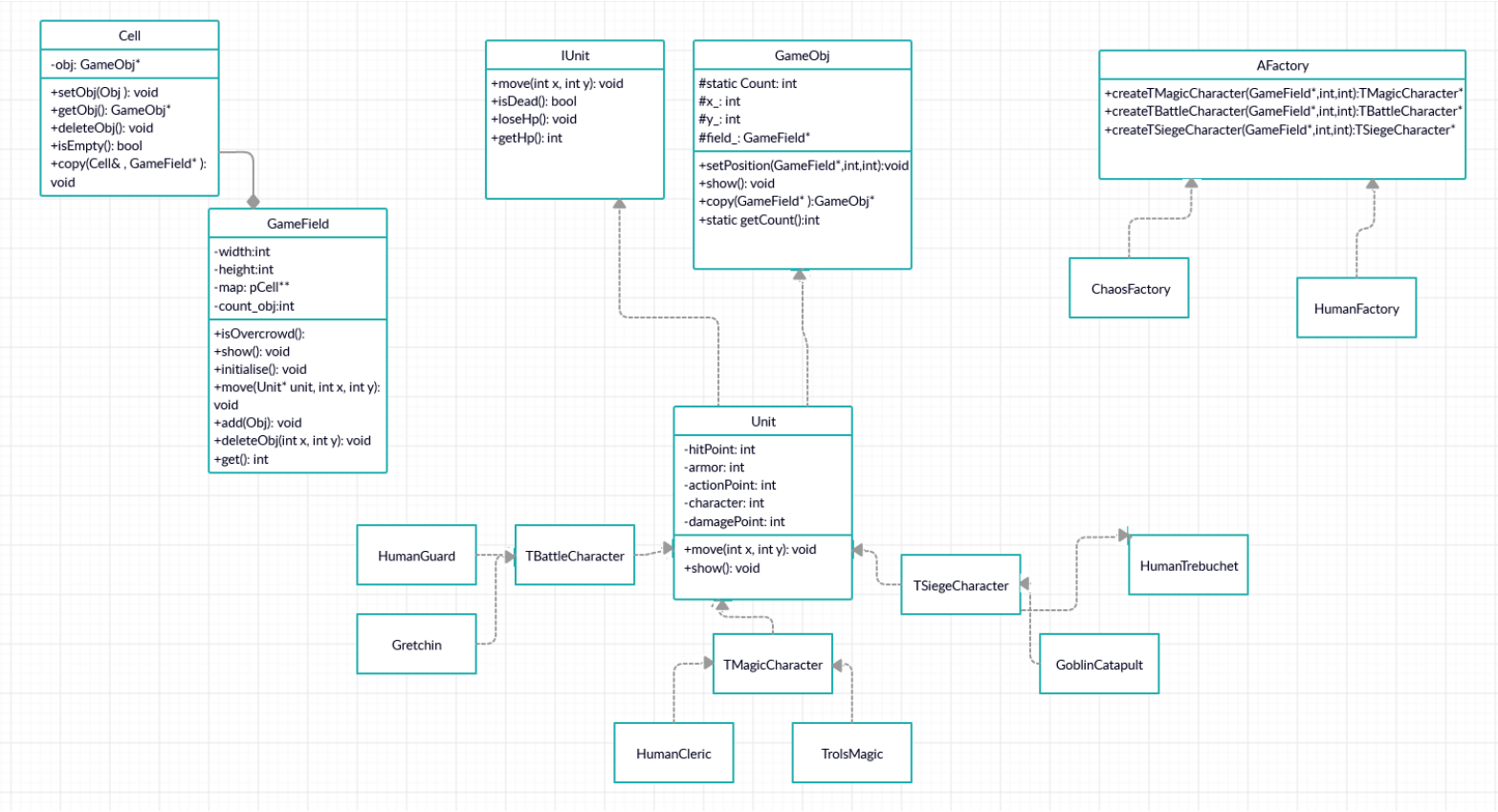

Рисунок 1 - UML диаграмма.

На рисунках 2-4 приведены некоторые примеры работы. Юнит умеет только двигаться. (1-я работа имела консольный вывод, а не графический как в лр. 2 и лр. 3).

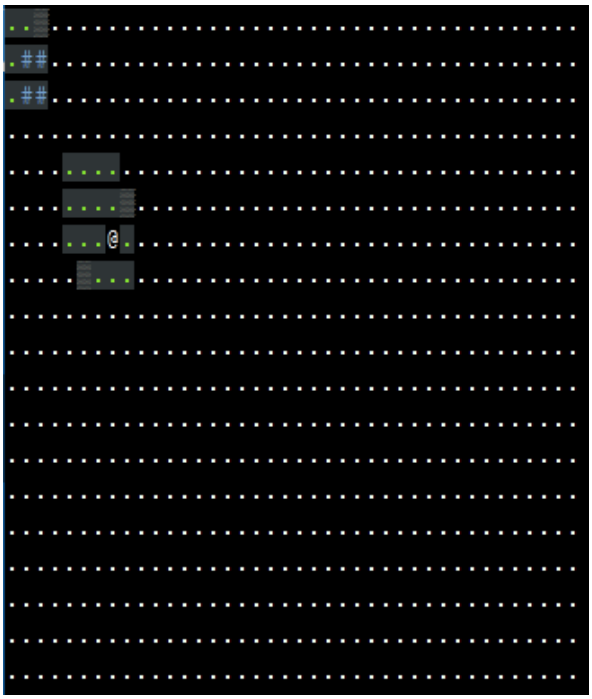

Рисунок 2 - пример работы.

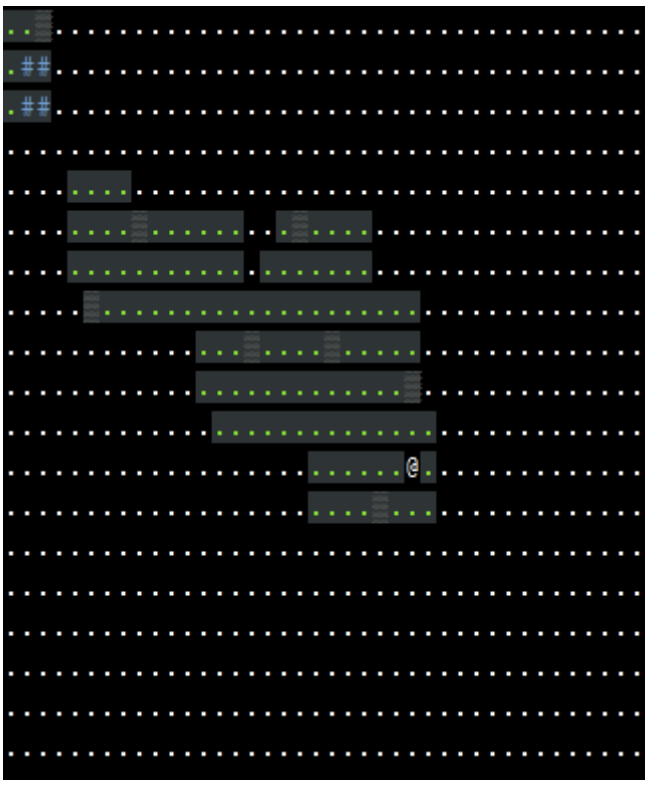

Рисунок 3 - Пример работы

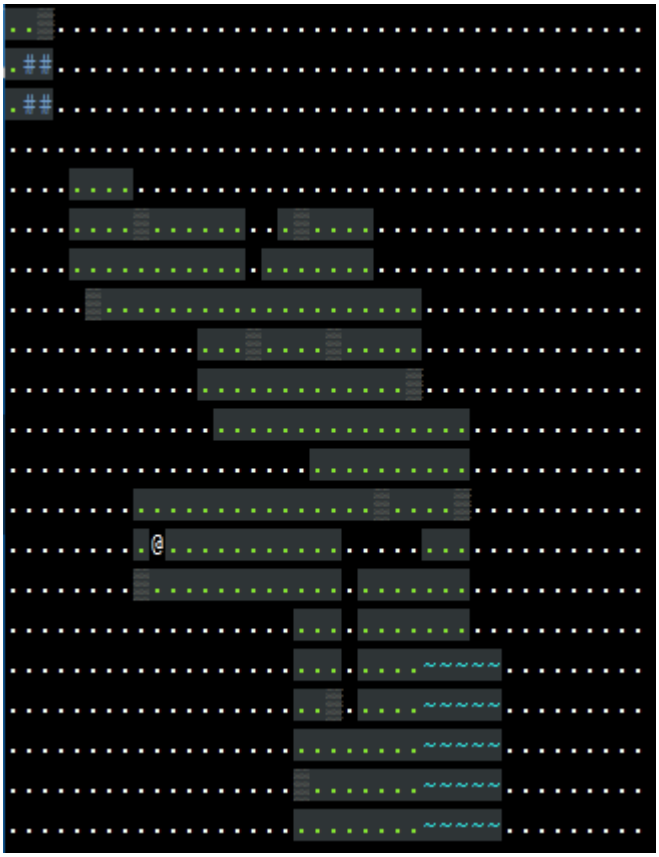

Рисунок 4 - Пример работы.

Выводы.

В ходе выполнения лабораторной работы были реализованы наборы классов для поля и разных видов юнитов. Для создания юнитов был использован паттерн "абстрактная фабрика", для игрового поля был реализован конструктор копирования.