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. 8383 _____ Костарев К.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

Цель работы.

Создать первоначальные элементы игры – игровое поле и юнитов с инструментарием для взаимодействия с ними, используя ООП (классы и наследование).ы

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.

В процессе выполнения лабораторной работы была придумана идея реализовать градостроительный симулятор, где игровое поле представляет собой

некоторую местность, юниты – это здания, здания подразделяются на некоторые типу в зависимости от специализации (жилые, коммерческие, службы), в качестве базы выступает ратуша, которая может быть модернизирована до мэрии. Здания добывают или отнимают в зависимости от своего вида ресурсы города, которые необходимо добывать. В дальнейшем в следующих лабораторных работах эти мысли будут дополняться.

class Board, class BoardStd

В первую очередь были реализованы классы Board и BoardStd (файлы GameBoard.h, GameBoard.cpp, GameBoardStd.h, GameBoardStd.cpp), где первый представляет собой игровое поле, а второй класс содержит функцию для ввода данных пользователем из консоли для создания игрового поля Board.

Private Поля класса Board:

1) int width, height – длина и высота игрового поля Board;
2) int objectOnBoard – счетчик числа элементов на игровом поле;
3) int coin – «казна», количество денег;
4) Cell*** cells – двумерный массив клеток игрового поля размера height*width.

Public методы класса Board:

1) Board(int w, int h) – конструктор, принимает аргументы длина и высота из функции класса BoardStd;
2) Board(const Board& old) – конструктор копирования;
3) Board(Board&& old) – конструктор перемещения;
4) ~Board() – деструктор;
5) int getWidth(), int getHeight() – геттеры длины и высоты;
6) void newBuild(), void delBuild() – строительство и разрушение здания на клетке;
7) void info() – информация о игровом поле и казне, по желанию пользователя информация о любой клетке.

class Cell

Представляет собой одну клетку игрового поля (файлы Cell.h, Cell.cpp).

Private поля:

1) IUnit* buildingOnCell – здание, расположенное на клетке, с базовым интерфейсом зданий IUnit.

Public методы:

1) Cell(), Cell(const Cell& oldCell) – конструктор и конструктор копирования;
2) void addBuild(char type) – функция строительства здания на клетке, вид здания определяется первой буквой названия вида (см. Buildings ниже);
3) void aboutCell() – печатает информацию о клетке, если есть на ней здание, то все его характеристики (о характеристиках будет ниже);
4) int aboutCell_Saldo() – возвращает сумму расхода (на обсулуживание) или дохода (налоги), которую дает здание.
5) void removeBuild() – функция разрушения здания на клетке;
6) char typeBuild() – возвращает первый символ названия вида, к которому относится здание на клетке.

class FactoryOfBuildings

Фабрика создания зданий определенного вида. Содержит public функцию IUnit* creator(TypeOfBuild typeOfB) которая принимает в качестве аргумента название вида здания, которое необходимо построить, из класса перечисления TypeOfBuild и возвращает новый объект класса.

enum TypeOfBuild

Класс перечислений всех названий видов зданий в игре (DACHA, HRUSHCHOVKA, FERMA, OFFICE, VODOKANAL, SCHOOL, POWERHOUSE). Описан в файле Naming.h.

class IUnit

Общий интерфейс зданий. Пока содержит только лишь виртуальную функцию TypeOfBuild typeOfBuild(), которая возвращает вид здания и определяется в классах соответствующих видов.

class Building: public IUnit

Абстрактный класс, который содержит поля, общие для всех зданий:

1) int saldo, energy, water, eat – доход/расход, потребляемая/вырабатываемая энергия, вода и пища

Также для этих полей определены соответствующие геттеры и сеттеры.

class Apartment: public Building

Абстрактный класс, который содержит поля, общие для жилых домов:

1) int workers, students, kids – взрослые рабочие, студенты, дети в доме.

Каждый тип людей имеет специфические потребности: для студентов должна быть школа, рабочие должны иметь рабочие места.

Также для этих полей определены соответствующие геттеры и сеттеры, а также метод getPopulation(), который возвращает общую численность населения дома.

class Production: public Building

Абстрактный класс, который содержит поля, общие для предприятий:

1) int vacancy – число вакантных мест у организации

Также для этих полей определены соответствующие геттеры и сеттеры.

class Services: public Building

Абстрактный класс, который содержит поля, общие для служебных муниципальных зданий. Но в рамках этой лабораторной работы они не были придуманы.

class Dacha: public Appartment

Класс, содержащий конструктор и определение виртуальной функции класса IUnit.

Дача – это здание, вмещающее в себя небольшое число человек, обязательно как минимум с 1 рабочим, но необязательно с детьми и студентами. Соответственно, потребляет немного ресурсов и дает небольшой доход и зависит напрямую от населения дома. Численность населения генерируется рандомной функцией.

class Hruschovka: public Appartment

Класс, содержащий конструктор и определение виртуальной функции класса IUnit.

Хрущевка – это здание, вмещающее в себя достаточное число человек, обязательно как минимум с 50 рабочими, и необязательно с детьми и студентами. Соответственно, потребляет много ресурсов и дает средний доход и зависит напрямую от населения дома. Численность населения генерируется рандомной функцией.

class Ferma: public Appartment

Класс, содержащий конструктор и определение виртуальной функции класса IUnit.

Ферма производит еду и имеет небольшое число вакантных мест, и приносит средний доход. Пока что количество производимой еды генерируется рандомной функцией, но это будет изменено в последующих лабораторных работах, когда клетки будут иметь определенный тип местности и ресурсы.

class Office: public Appartment

Класс, содержащий конструктор и определение виртуальной функции класса IUnit.

Офис дает большое число вакантных мест, и приносит большой доход, но дорого стоит и потребляет много ресурсов.

class School: public Services

Класс, содержащий конструктор и определение виртуальной функции класса IUnit.

Школа дает студентам образование, но приносит расходы и потребляет много ресурсов.

class Powerhouse: public Appartment

Класс, содержащий конструктор и определение виртуальной функции класса IUnit.

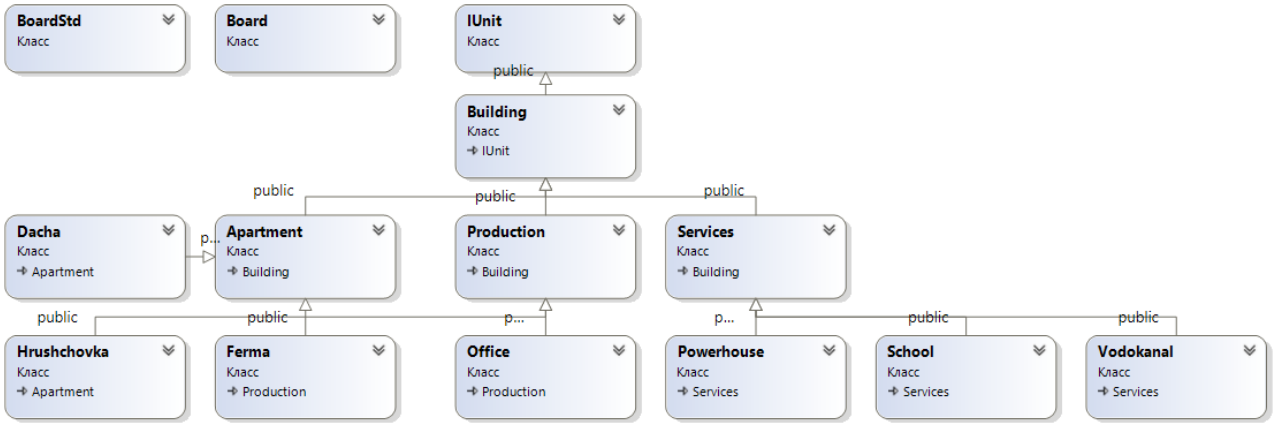
Электростанция производит электричество и приносит расходы. Пока что количество производимой электроэнергии генерируется рандомной функцией, но это будет изменено в последующих лабораторных работах, когда клетки будут иметь определенный тип местности и ресурсы.

class Vodokanal: public Appartment

Класс, содержащий конструктор и определение виртуальной функции класса IUnit.

Водонапорная башня ВОДОКАНАЛ производит воду и приносит расходы. Пока что количество производимой воды генерируется рандомной функцией, но это будет изменено в последующих лабораторных работах, когда клетки будут иметь определенный тип местности и ресурсы.

UML-диаграмма:



Демонстрация работы.

Пример 1. Пустое поле, добавление элемента и вывод информации

```
Enter width and height:
5 6
ADD BUILD, Enter x, y:
2 5
D, H, F, O, P, S, V:D
Buildings on board: 1
Coin: 1988
0 0 0 0 0
0 D 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
You want to know more about the individual cell? (1 - yes, 0 - no)
1
Enter x, y:
2 5
Saldo: 12
Energy: -20
Water: -16
Eat: -8
Workmans: 1
Students: 1
Kids: 2
```

Пример 2. Удаление элемента и вывод информации

```
Enter width and height:
7 7
ADD BUILD, Enter x, y:
2 3
D, H, F, O, P, S, V:F
DEL BUILD, Enter x, y:
2 3
Buildings on board: 0
Coin: 1900
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
```

Пример 3. Копирование города, изменение первого, вывод информации о скопированном

```
Enter width and height:
6 6
ADD BUILD, Enter x, y:
3 4
D, H, F, O, P, S, V:O
Buildings on board: 0
Coin: 2000
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

Выводы.

В ходе выполнения данной лабораторной работы были созданы классы (в т. ч. абстрактные и интерфейсы) и реализовано наследование классов для первоначальной работы игры.