Лабораторная работа №1

Дана некая игра-стратегия, которую хорошие и грамотные люди написали на языке программирования C#. Во время написания они всячески использовали принципы ООП, соблюдали понятный и правильный код-стайл и даже, страшно представить, комментировали код.

Правила игры:

Два игрока противостоят друг другу на карте, разбитой на квадраты (или клетки). Клетки считаются с левого верхнего угла, координата X - справа-налево, координата Y сверху-вниз

(X, Y)
(0,0) (1,0) ...
(0,1) (1,1) ...
...

Каждый квадрат имеет определённый тип местности:

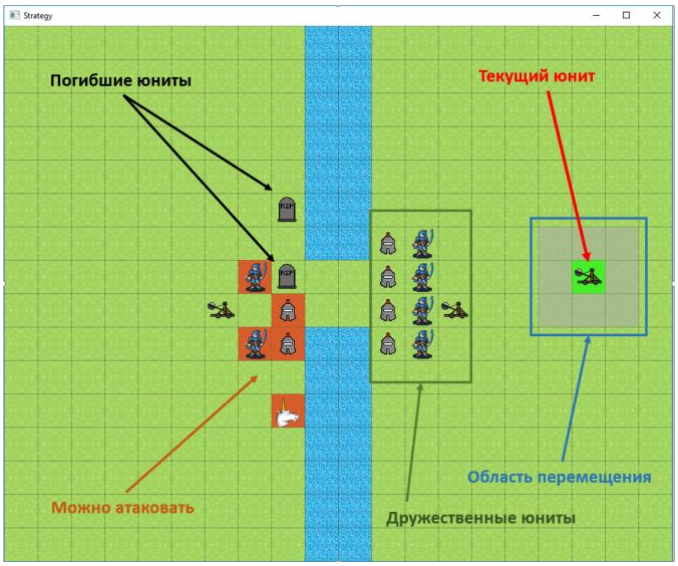
1. *Трава - юнит может находиться и перемещаться на этот квадрат.*
2. *Вода - юнит не может переместиться в эту клетку.*

Каждый игрок имеет набор юнитов. Юнит - это игровой объект, который может перемещаться по карте и наносить повреждения юнитам противника.

Характеристики юнита зависит от его типа. Всего существуют 4 типа юнита:

- *Лучник. Может стрелять на 5 клеток и перемещаться на 3 клетки. Наносит 50 урона, если цель находится далеко от него, и в два раза меньше, если цель находится рядом. Имеет 50 здоровья.*
- *Катапульта. Может стрелять на 10 клеток, но перемещается только на 1 клетку. Наносит 100 урона в дальнем бою, и в два раза меньше в ближнем. Имеет 75 здоровья.*
- *Всадник. Перемещается на 10 клеток, может атаковать только тех врагов, которые стоят рядом с ним. Наносит 75 урона и имеет 200 здоровья.*
- *Мечник. Перемещается на 5 клеток, атакует только ближних врагов. Наносит 50 урона и имеет 100 здоровья.*

На одной клетке может находиться только 1 юнит. Также, если у юнита остаётся 0 очков здоровья, то он умирает и больше не перемещается/атакует, но продолжает занимать клетку.



Написанную игру они залили на github в публичный репозиторий и были таковы. Прошло много времени, много воды утекло, Microsoft купила github и во время перетаскивания БД на новые серверы нерадивые инженеры её частично затерли.

К сожалению, написанная стратегия попала под удар и когда сервис заработал на новой инфраструктуре, репозиторий с игрой оказался пустым.

К счастью, сохранился один форк этой игры (правда, почему-то, без истории изменений), и теперь кому-то придётся восстановить из него написанную ранее игру.

Проблема состоит в том, что у хозяина этого форка очень специфичные вкусы, и он яростно ненавидит ООП и красивый код (комплексы из-за травмы детства), поэтому восстанавливать придётся в итоге очень кривой и странный код, который на данный момент представлен следующими классами:

Классы объектов:

- Grass - класс травы на местности
- Water - класс воды на местности
- Archer - класс лучника
- Catapult - класс катапульты
- Horseman - класс всадника
- Swordsman - класс мечника

Класс контроллера:

- GameController - контроллер, который предоставляет необходимую информацию для игры.

Дополнительные классы

- Map - содержит информацию обо всех игровых объектах
- Player - класс с информацией об игроке
- Coordinates - вспомогательный класс для передачи координат.

Ну и теперь о самом веселом – этот кто-то, кто будет восстанавливать, Вы☺

Ваша задача отрефакторить этот код согласно принципам ООП и здравой логики, а также допилить к ней интерфейс * (дизайните, как душе угодно). В качестве помощи и возможности проверить свои изменения - используйте проект Strategy.Domain.Tests. Рекомендуется начать Вашу работу с запуска и проверки всех тестов. Помимо этого, данный проект является Вашим ориентиром. Вы можете менять внутреннее содержимое библиотеки Strategy.Domain, но публичный интерфейс должен оставаться тем же.

Если Вам пришлось что-то поменять в проекте с тестами, значит любой код, который использует библиотеку может не скомпилироваться.

Однако, следующие изменения будут допустимы в рамках данного задания:

- Перемещать файлы в Strategy.Domain. Тогда необходимо будет поправить using'и в Strategy.Domain.Tests. Это не критично.
- Переименовать параметры методов в GameController. Также изменить тип с object'а на что-то другое. Это затронет больше изменений в Strategy.Domain.Tests, но они не будут сложными.

Подсказки:

- Основной рабочий инструмент - наследование и полиморфизм!
- Предполагайте, что в будущем можно будет создать новых юнитов и типы местности. Чем меньше изменений потребуется внести в проект, тем лучше.
- Код также содержит много кривых конструкций, которые не связаны с ООП. Его тоже желательно отрефакторить.

- GameController может иметь реализацию каждого метода в 1 строчку и при этом не иметь непубличных методов.

1. Базовое задание – рефакторинг имеющегося кода
2. Дополнительное задание на звездочку – дописать интерфейс, либо переписать весь проект на другой язык (включая интерфейс).

Каждое выполненное задание со **звездочкой** дополнительно прокачивает Вас как специалиста, а ещё, даёт бонусы:

- **1 задание со звездочкой** – 1 обязательное задание из любой следующей лабораторной можно не делать.
- **3 задания со звездочкой** – можно не делать 1 любую лабораторную в семестре.