Паттерн Команда (*Command*)

**Вводные слова:**

Программа является простой игрой в крестики нолики против бота. Он вышел тупенький, потому что он все делает через рандом, без стратегии (все-таки главное - реализация паттерна, а не гениального искусственного интеллекта).

Каждая вторая, если не каждая первая, статья в интернете про паттерн Command так или иначе затрагивает видеоигры, поэтому и для индивидуального проекта я выбрал сделать небольшую игру.

Обычно паттерн Command используют когда:

- Надо передавать в качестве параметров определенные действия, вызываемые в ответ на другие действия. То есть, когда необходимы функции обратного действия в ответ на определенные действия

- Необходимо обеспечить выполнение очереди запросов, а также их возможную отмену

И, по моему скромному мнению, знаменитая игра в крестики нолики может извлечь большую выгоду из этого паттерна.

(**Доп. комментарий:** начать новую игру после победы или поражения можно с помощью меню в левом верхнем углу. Ничья за победу или поражение не считается.)

**Применение паттерна:**

В программе паттерн применяется для передачи действий, вызываемых в ответ на действия игрока, а также для создания очереди ходов с последующей функцией их отмены.

**Описание диаграммы:**

Итак, в этой программе есть интерфейс команды - ICommand, есть ее реализации в виде классов Command\_AI\_Move и Command\_Player\_Move для соответственно действий бота и игрока.

Основные действия начинаются в классе Program, который является основной входной точкой в программу.

Интерфейс IGame в основном используется в самой windows форме, в то время как класс Game используется другими классами для просмотра поля или, например, проверки кто победитель в текущей игре. В Game также используется Random, как я и говорил, для хода бота (класс Random определен не мной, а системой, поэтому его ни в коде, ни в диаграмме почти нет).

Класс Invoker, очевидно, является инициатором команды - он вызывает команду для выполнения действия. Он также пользуется классом Stack для того, чтобы последовательно записывать команды, чтобы потом их можно было легко отменить одну за другой (хоть несколько, хоть все подряд). Класс Stack, как и Random - системный.

ModelClient - в широком смысле - клиент, но возможно для данного случая правильнее будет использовать слово игрок, хмм...

Ну и остался только Presenter или Представитель, он является чем-то вроде помощника для Представления изменений на экране.