1. Булевая статическая матрица размером х \* х, что хранит в себе текущее состояние
   1. Null – исходное значение полей
   2. True – ход крестиками
   3. False – ход ноликами
2. Класс/метод display - отображает текущее состояние матрицы
3. Метод humanMove - позволяет ходить человеку = считывает нажатие стрелок или пробела (камень); либо, вариант 2, - ввод адреса, куда ставить камень. Я за 1-е
4. Метод interpret(до такой-то глубины) = интерпретирует всю матрицу, ищет в ней линии по всем направлениям по заданным критериям. На основании выхлопа этого метода, метод think будет принимать решение, куда ставить следующий камень ком-ру.

Параметры метода:(1) N до какой длинны линию искать, (2) глубина сканирования (для больших линий, как то 5), т.к. не обязательно искать до одного камня «в ряд», можно ограничивать поиск до 3 или 2 камня):

* 1. есть ли хоть одна лини в заданное кол-во камней в ряд (например 5 = выигрышная комбинация = конец игры)
  2. сколько есть неприкрытых линий соперника длинной N-1 (неприкрытая линия/комбинация – та, что еще можно достроить либо в обе стороны)
  3. сколько есть прикрытых с одной (либо недостроенной изнутри) стороны линий соперника длинной N-1
  4. … N-2 соперника неприкрытые
  5. … N-2 неприкрытые наши, сколько
  6. Прикрытые линии N-2 соперника, сколько
  7. Прикрытые N-2 линии соперника, сколько
  8. Прикрытые с одной стороны наши линии N-2ки, сколько
  9. Неприкрытые N-3 соперника, сколько
  10. …
  11. Зависит от блины линии
  12. …
  13. Неприкрытый в максимуме направлений камень соперника
  14. Неприкрытый в максимуме направлений наш камень

1. Метод Think! Выбирает оптимальные наш ход

Ищем ход привод ящик к лучшему возможному решению, по приоритетности:

* 1. Вариант 1 («брутальный перебор» в хх ходов вперед) плохой, тк при большом поле - очень ресурсоемкий уже на 3-4м ходу, нужно хоть чуть-чуть поумнее сделать {
     1. Ходим случайно
     2. Interpret
     3. Выбираем ход с наилучшим значением interpret
     4. Ходим соперником случаймо
     5. Interpret
     6. Выбираем ход соперника тоже по наилучшему результату interpret

}

Можно походить на несколько ходов вперед используя логику челокева, как и наша (по максу interpret-a), запомнить ходы, проверить исход и если он был неудачный, то переходить на актуальном ходу.

* 1. Вариант 2 («умный») юзаем метод think!

Логика принятия решения по следующей модели:

1. Выиграть (постивить N в ряд)
2. Не проиграть (не дать постивать N в ряд сопернику)
3. Поставить своїх N-1 неприкрытых камня
4. Прикрыть неприкрытые N-2 линии соперника (=Не дать сопернику поставить неприкрытую N-1 в следующем ходу)
5. Строить две неприкрытых свои N-2
6. Строить одну свою неприкрытую N-2
7. Строить свою неприкрытую с одной стороны N-2 (под вопросом)
8. Строить N-1 линию неприкрытую с одной стороны (под вопросом)
9. Строить N-2 неприкрытую с одной стороны
10. …
11. … ?
12. Прикрывает N-3 соперника по диагонали (обдумать – может парЫ, а не парУ)
13. Прикрывает N-3 соперника по прямой (обдумать – может парЫ, а не парУ)
14. Ставит камень рядом по диагонали с соперником (под вопросом)
15. Ставит камень там, где максимум неприкрытых направлений
16. Случайное незанятое место на матрице
17. Иначе exception – конец игры = ничья
18. Метод move – ставит в матрицу камень того или иного игрока
19. Main: по кругу: {
    1. Если interpret нашел N единиц, то конец игры
    2. Display
    3. humanMove
    4. move
    5. interpret
    6. think
    7. move

}