# Теоретичні відомості

Для реалізації можливості вибору режиму гри було створено окрему форму з двома кнопками, які відповідають за кількість гравців, та чекбоксом для вибору режиму гри з таймером. Для створення ігрового поля було використано двовимірний масив з власного класу Grid який унаслідується від класу Button і додатково містить свої координати на полі. Також було використано обробник подій який реагував на натискання кнопки ігрового поля і за допомогою класу Presenter викликав відповідний метод класу Game який обробляв клік. Для гри створюється об’єкт типу IPlayer у який присвоюється об’єкт класу Player або EnemyBot (залежно від режиму гри)

Детальніше про алгоритм для кожного режиму гри:

* Гра з іншим гравцем на одному девайсі:

Було використано булеву змінну для чередування ходу. Після кожного ходу викликався метод IsWin який перевіряв чи виграв гравець і повертав булеве значення.

* Гра з іншим гравцем на одному девайсі з таймером:

Після кожного ходу не викликається метод IsWin натомість після закінчення часу викликався метод підрахунку балів який рахує кількість повних виграшних ліній (лінія з 5 в ряд дає одне очко) і визначає переможця.

* Гра проти комп’ютера:

Створюється екземпляр класу EnemyBot в якого викликається метод Move одразу після кліку на клітинку поля. Метод Move знаходить можливі ходи (сусідні клітинки з тими на яких є хрестик або нулик), потім оцінює ці ходи за допомогою методу GetGrade. Цей метод перевіряє чи співпадає лінія клітинок з шаблоном (елементом списку \_patternsStrForX та \_patternsStrForO) і присвоює їй оцінку відповідно до збігу. І в клітинку з найбільшою оцінкою здійснюється хід.

* Гра проти комп’ютера з таймером:

Після кожного ходу не викликається метод IsWin натомість після закінчення часу викликався метод підрахунку балів який рахує кількість повних виграшних ліній (лінія з 5 в ряд дає одне очко) і визначає переможця.