**Порядок работы**

1. Продумать иерархию программы
2. Продумать UML-диаграммы классов для архитектуры программы
3. Реализовать первые классы и методы, отвечающие за переходы между окнами
4. Продумать UML-диаграммы классов для бота, игрока, второго игрока, правила игры или фигур (условия матов, шахов, возможных и недопустимых ходов,подсветка ходов, взятие на проходе), настроек и отдельных функций игры, также оставить возможность добавления звуков в игру
5. Реализовать их (4) пункт, добавить примитивные спрайты (хит-боксы, прямоугольники, которые в будущем будут заменены на спрайты
6. Реализовать Графический интерфейс, расставить все объекты
7. Добавить Спрайты
8. Тестить
9. Можно добавить звуки в конце

**Порядок проектирования:**

1) Piece/Figure, // нужна для класса ChessBoard и в настройках

2) ChessBoard, // нужна для всех экранов

3) ControlPanel // таймер, история ходов, кнопочки всякие

4) AI

// Дальше обобщённо

5) Game, // разделим на два потока - рендер (ради часиков) и логику

Settings,

Editor,

Study, // этюды

6) Menu,

7) Application

Piece/Figure:

params:

chessboard, // доска, к которой фигура привязана (может и не быть)

side, // то бишь цвет

methods:

getAvailableTurns(), // может например возвращать 8x8 двумерный массив с bool значениями

СhessBoard:

params:

mode, // типо edit, play и fixed

pieces/figures,

size, // визуальный размер доски

methods:

getPiece(coords),

removePiece(coords), // убирает с доски и возвращает фигуру

placePiece(piece, coords),

Application:

main

class Game extends Thread {

public void run\_logic() {}; # основной поток

public void run\_render() {}; # основной поток

@override

public void run()

{

run\_render();

}

# Класс Game должен содержать переопределённый метод public void run(), в

# котором запустим рендер

# 1 способ

#Thread myThready = new Thread(class\_that\_implements\_Runnable\_interface);

#myThready.start();

# 2 способ, используем его.

#object\_of\_this\_extended\_class = new class\_that\_extends\_from\_Thread();

#object\_of\_this\_extended\_class.start();

}

class Menu {

public void GoMenu()

{

Game = new Game();

Game.start(); // запускает render

Game.run\_logic(); // запускает logic

}

};

// Никитин вариант

class Game {

public void go() { // или run, хз

Thread renderThread = new Thread(new Runnable(

) { render(); }

)

// или же Thread renderThread = new Thread(() -> {render();}

renderThread.run();

logic();

}

private void logic() {}

private void render() {}

}

**Наши соглашения:**

run - методы для запуска потоков

go - методы для запуска непотоков

<https://habr.com/ru/post/164487/>