Сказка за 29 сентября, Понетайкин

Мы говорили о том, зачем разделять классы Control и Boundary

Control – сервер, Boundary – клиент, общаются по сети.

Main читает конфигурационный файл, в котором указаны по умолчанию некоторые параметры, например имя файла, в котором Boundary будет писать свои логи (если при прочтении не понимается команда, Boundary ругается на клавиатуру и файл, в Control ничего не посылается).

Control пишет в лог, что получил файл от Boundary, всю информацию про генерацию, пишет в лог, что все хорошо, возвращает в Boundary, и на этой основе Boundary взаимодействует с интерфейсом и возвращает код ошибки 0, 1. Идет сравнение двух txt-шных файлов (логов Control и Boundary).

В конфигурационном файле пишется 2 имени txt файлов для логов Control, Boundary. Перед каждой командой в логе пишется время получения команды и время отдачи результата. Так же дата в начале конфигурационного файла.

Команды по умолчанию (если гаусс пришел без параметров, то берем параметры по умолчанию из конфигурационного файла, по 1 все параметры и диагональная матрица).

Будут команды текущего расположения курсора генерации. Можно двигать на поле по пикселям текущую точку, говорим сгенерировать точку – генерирует на положении курсора. По умолчанию курсор стоит на середине поля, точка умолчания пишется в конфигурационный файл.

Есть команды передвинуть курсор на дельты X и Y, или на абсолютные координаты.

Main читает текущий файл, создает объект интерфейс, так же в конфигурационном файле написан командный файл для выполнения, он передается интерфейсу.

В конфигурационном файле написано число командных файлов и имена, их получает Boundary, читает параметры, все понимает – вызывает операции у Control, он выполняет и возвращает логи к Boundary.

Класс Control должен иметь все команды, которые он может выполнять.