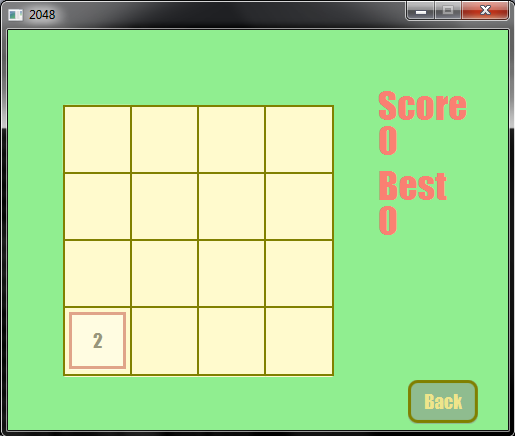
Интерфейс приложения

Стартовое меню:



Из него пользователь может попасть в игру:

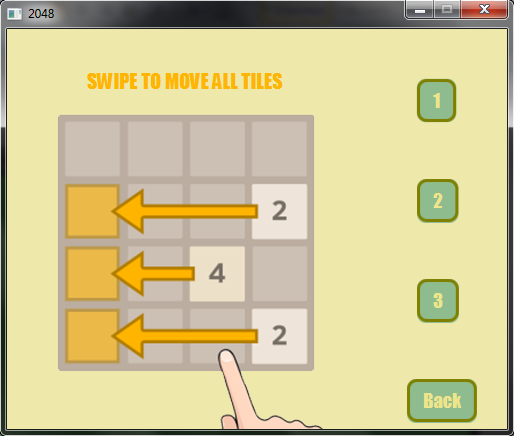


Из игры, посредством нажатия кнопки ”Back” пользователю будет представлено окошко для подтверждения действия (данное окошко также отображается при выходе из игры):

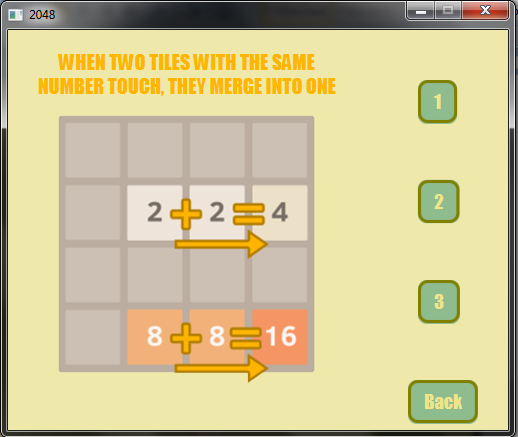


Из меню пользователь также имеет возможность попасть в окно обучения, которое состоит из трех переключаемых сцен.

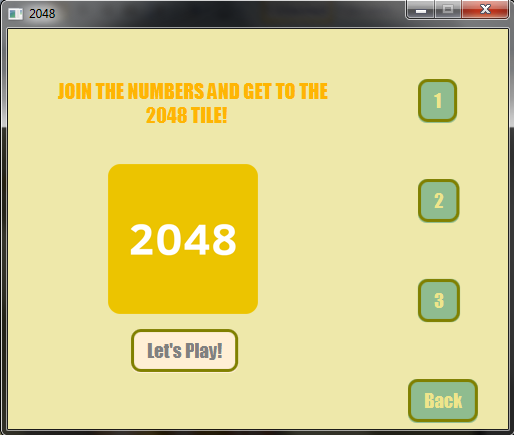
Первая:



Вторая:

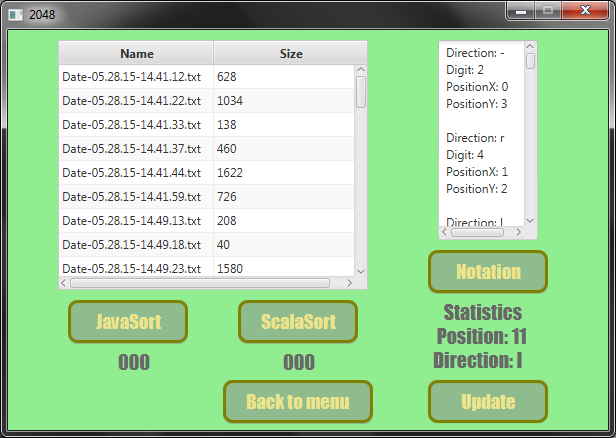


Третья:

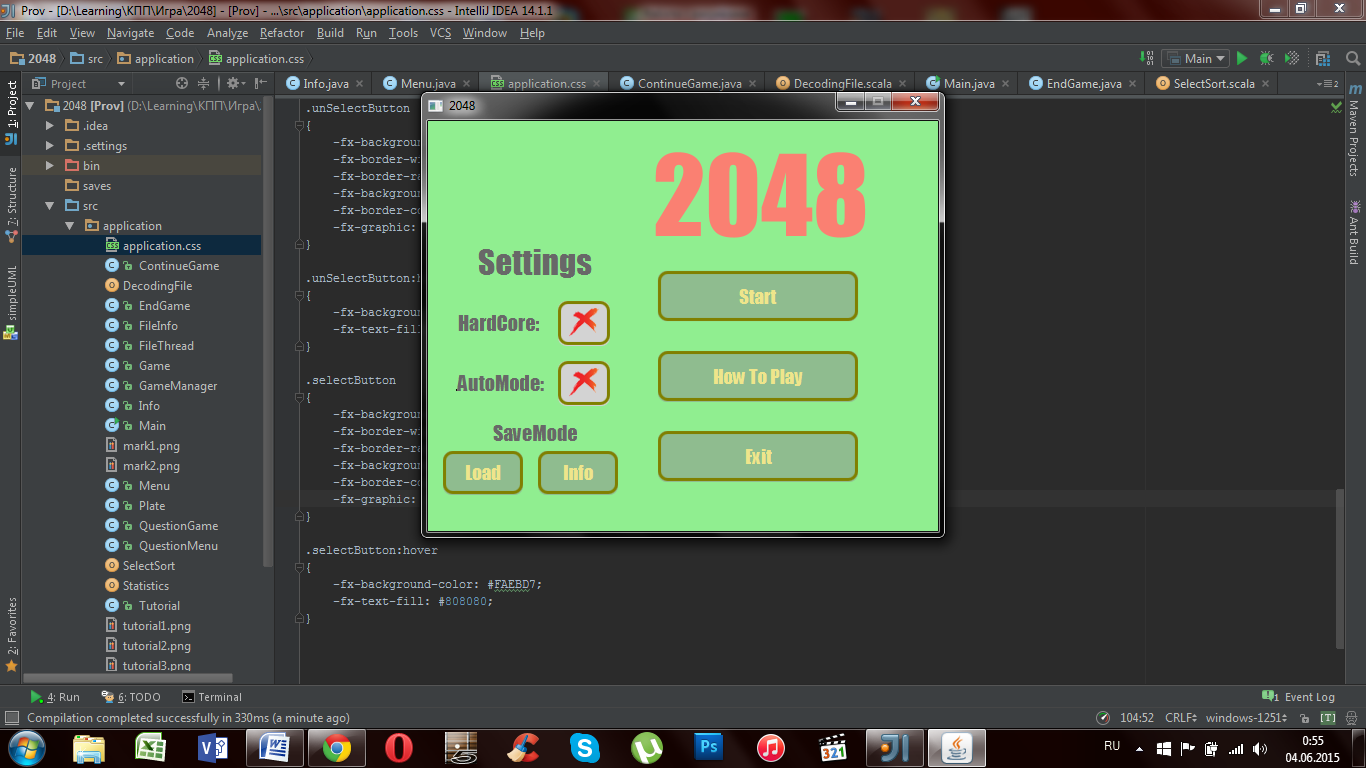


Из третей сцены пользователь может сразу перейти к игре.

Сцена информации:



Проект, запущенный под Windows:



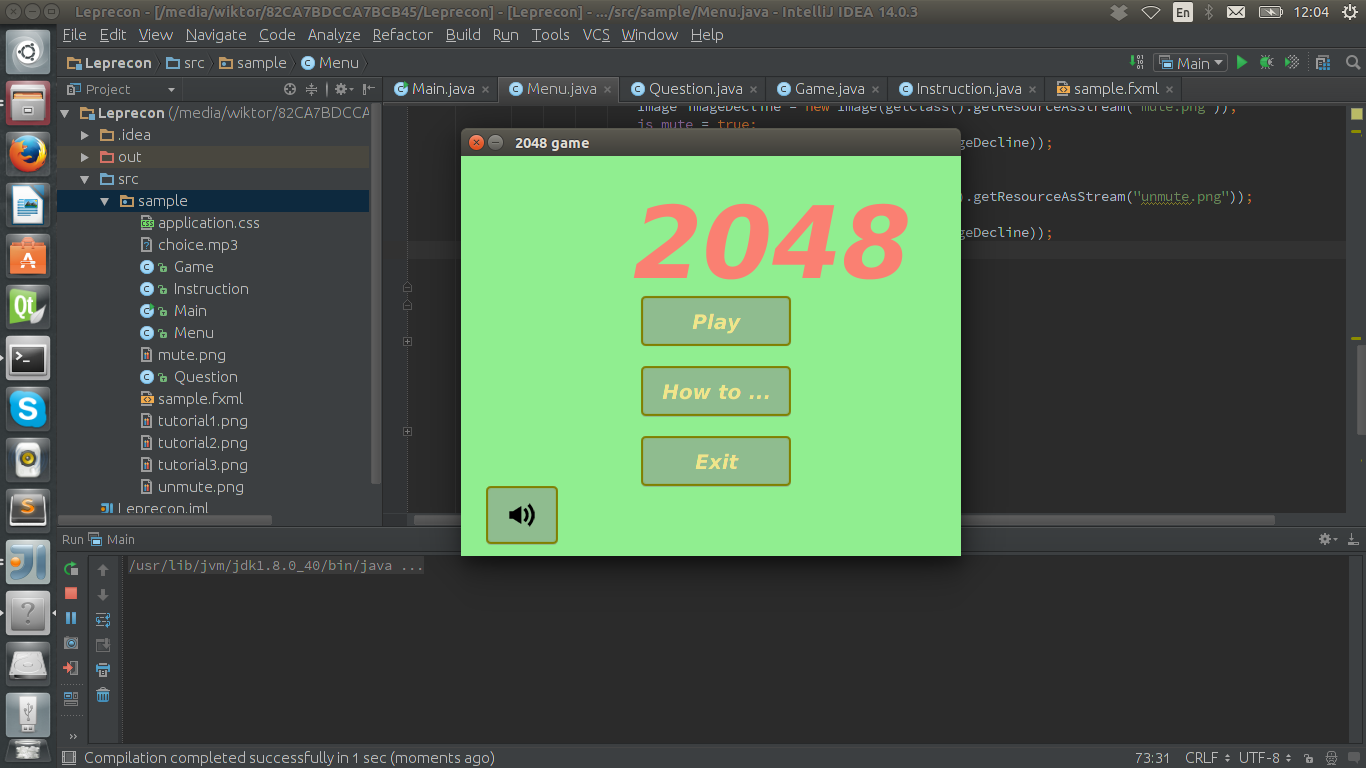
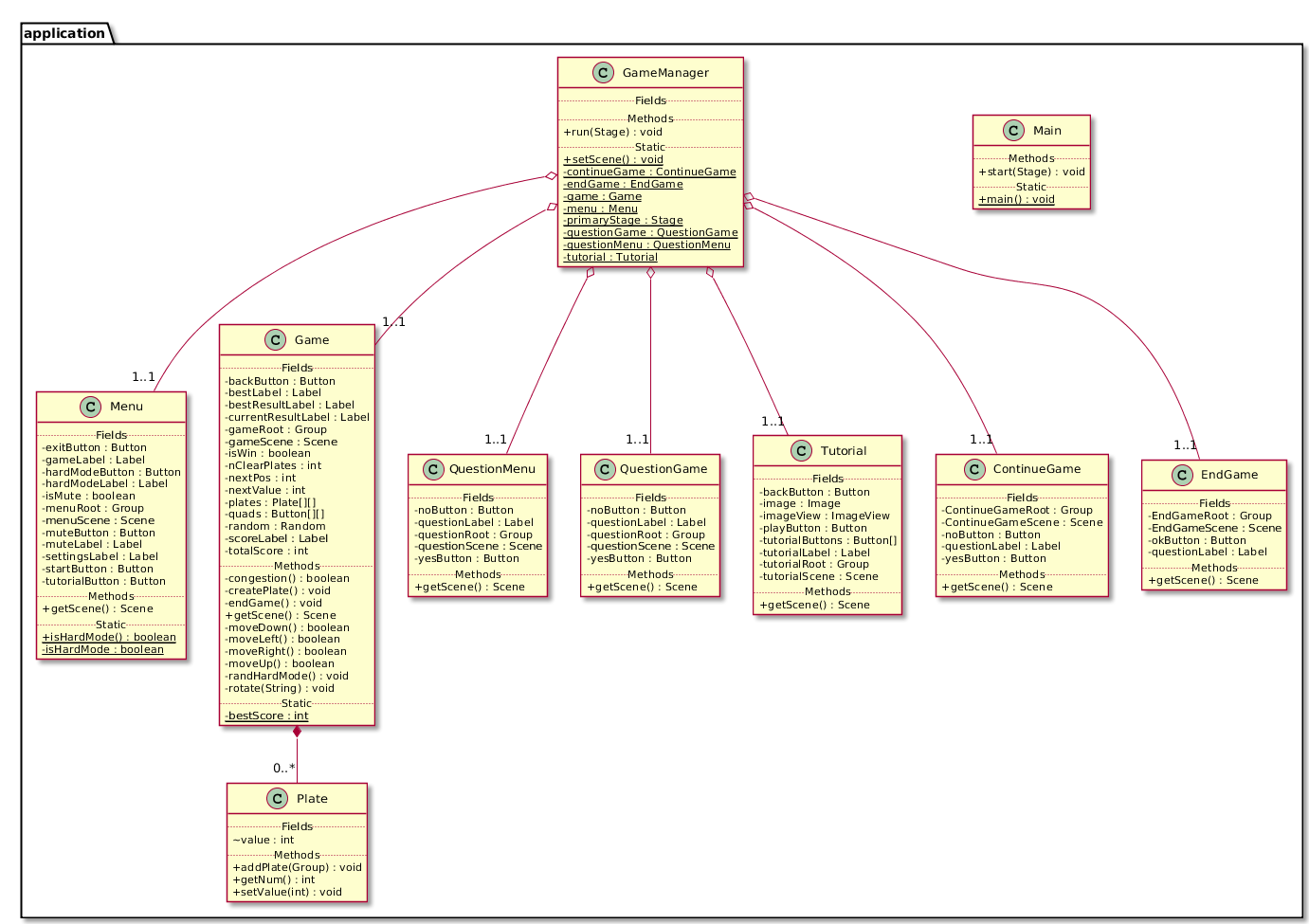
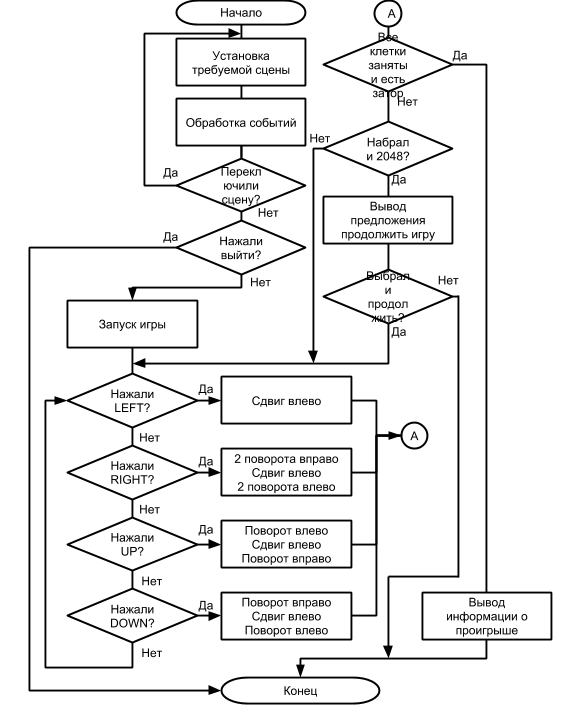
Проект, запущенный под Linux:

Диаграмма классов

Блок схема основного алгоритма

Сохранение и воспроизведение

Нотация

Пример файла с записанной игровой сессией:

410 l203 d230 u210 u420 l130

Первый символ – направление сдвига (вверх/влево/вправо/вниз)

Второй символ – сгенерированное значение (2 или 4)

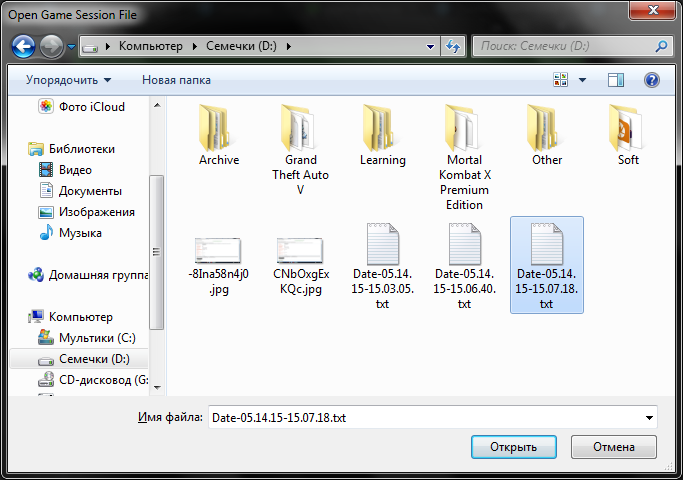
Третий символ – позиция появления сгенерированного значения (столбец)

Четвертый символ – позиция появления сгенерированного значения (строка)

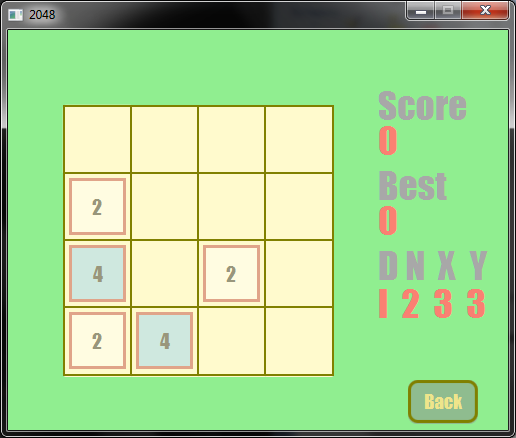
Сохранение игры происходит при каждой установки игровой сцены. Для выбора файла для последующего воспроизведения существует кнопка Load:



По нажатию на данную кнопку осуществляется переход к дереву каталогов, из которого можно выбрать файл с сохраненной сессией:



Во время повтора записанной игры, справа внизу, на игровом поле отображается нотация:



Из данного примера видно, что после сдвига влево (D - l) появилась цифра 2 (N - 2), в 3-ем столбце (X - 3) и в 3-ей строке (Y - 3).

Одним из аналогов хранения информации о записанной игре является сохранение матрицы (массива) значений, которые находятся в ячейках в заданные моменты времени. Отображение записанной игры заключается лишь в последовательном переключении матриц (массивов). Однако реализованный метод записи сохраненной игры более компактен и позволяет хранить информацию об играх в файлах гораздо меньшего размера.

Работа с потоками

Режим воспроизведения сохраненной игры реализован в отдельном потоке. Данный поток представлен классом FileThread, который наследуется от класса Thread. В классе, реализующем поток чтения файла, переопределен метод run(). В нем описываются все инструкции необходимые для считывания информации с файла и передачи ее в функцию saveModePlay для воспроизведения сохраненной игровой сессии.

Т.к. первый блок информации отличается от последующих (отсутствует символ направления сдвига), считывание первого блока информации из потока чтения in не реализуется в пользовательском потоке FileThread. Далее создается объект класса AnimationTimer, работа которого синхронизируется с работой потока FileThread через функцию synchronized. В данную функцию передается объект, который требуется синхронизировать(AnimationTimer) c пользовательским потоком FileThread. Затем AnimationTimer с помощью функции wait() ожидает выполнения метода notifyAll, для последующей записи считанных из файла значений в требуемые переменные.

Далее поток засыпает на секунду, чтобы дать возможность успеть применить требуемые изменения. Затем поток FileThread снова считывает блок информации из файла с сохраненной игрой. Таким образом, реализуется многопоточность.

Работа с функциональным языком Scala

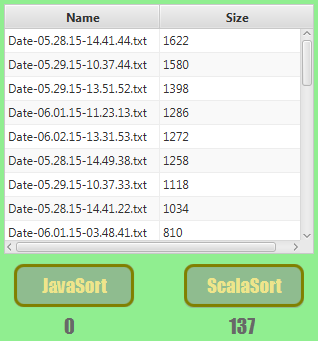
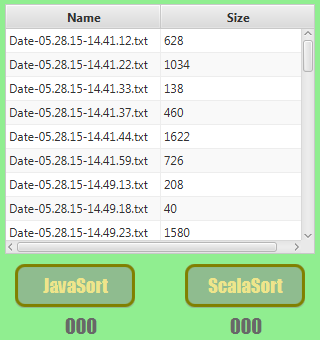
Сортировка игр по выбранному критерию

С помощью функционального языка программирования Scala, а также используя средства Java необходимо произвести сортировку сохраненных игр по одному из критериев.

Игра 2048 предполагает собой, что чем больше времени пользователь играет, то есть чем больше ходов он совершил, тем больше очков он заработал. Следовательно, в файл записывается большее количество информации. Таким образом, критерием по которому будет выполняться сортировка, является размер файла.

В отдельном классе Sort реализована сцена, при переключении на которую, пользователь может увидеть сформированную с помощью класса tableView таблицу из названий файлов и их размеров. В данной сцене доступны функции сортировки с помощью Java и Scala. Каждый из способов занимает определенное количество времени, что вычисляется с помощью функции System.currentTimeMillis(); и отображается с помощью соответствующих элементов графического интерфейса приложения.

Как видно из результатов полученного времени сортировок, Java справляется с данной задачей в разы быстрее Scala.



Приведение статистики по игровым сессиям

В сцене, отображающей информацию об играх (таблица сохранений, функции сортировки) имеется также функция обновления статистики игр.

Статистика ведется по двум показателям:

1. Наиболее часто генерируемое направление сдвига
2. Наиболее часто генерируемая позиция появившейся цифры

В классе Info имеется обработчик нажатия клавиши Update, работа которого заключается в составлении списка из критериев для приведения статистики. Данную информацию обработчик получает из функции statisticsParsing(DataInputStream in), где парсятся файлы сохранений. После парсинга данные отправляются в соответствующую Scala-функцию, где происходит их обработка.

Возвращенные из вышеописанной функции результаты устанавливаются в statisticsLabel и отображаются пользователю над кнопкой Update.



Подмена нотации игровой сессии

Подмена разработанной нотации псевдокодом для повышения её читаемости неискушённым знаниями нотации пользователем происходит в соответствующей функции Scala.

Взаимодействие пользователя с данной частью функционала программы происходит через кнопку NotationButton и поле для текста NotationArea. По нажатию на данную кнопку пользователю предлагается выбор файла, из которого необходимо считать данные об игровой сессии и конвертировать в требуемый вид.

После выбора файла специальная функция statistcisParsing(DataInputStream in) формирует список из строк, элементом которой являются блоки информации о каждом шаге сохраненной игры.

Данный список передается в соответствующую функцию Scala для обработки. В ней список разбивается на головной элемент head и подсписок output. Затем функция уходит в рекурсию с параметрами: оставшаяся часть(подсписок), и конвертированный блок информации.

Таким образом, формируется строка из конвертированных блоков информации, которую и возвращает функция.

