**2048 (1)**

Привет! Надеюсь ты уже успел устать от обычных задач и с нетерпением ждешь большую!  
Сегодня напишем java реализацию игры 2048. Вкратце, если ты о ней ничего не слышал, целью игры является получение плитки номиналом 2048 на игровом поле 4х4. Подробнее можешь прочитать в википедии *https://ru.wikipedia.org/wiki/2048\_(%D0%B8%D0%B3%D1%80%D0%B0)*

Для начала нам понадобятся такие классы:  
1. Controller — будет следить за нажатием клавиш во время игры.  
2. Model — будет содержать игровую логику и хранить игровое поле.  
3. View — обеспечит отображение текущего состояния игры на экран.  
4. Main — будет содержать только метод main и служить точкой входа в наше приложение.

Создай их!

Требования:  
1. Должен быть создан класс Controller.  
2. Должен быть создан класс Model.  
3. Должен быть создан класс View.  
4. Должен быть создан класс Main с публичным статическим методом main (String[] args).

**2048 (2)**

В игре 2048 поле состоит из 16 плиток, каждая из которых имеет определенный вес.  
Кроме веса у плитки еще будет собственный цвет и цвет текста которым будет отображаться вес плитки.  
Цвета плиток находятся в диапазоне от светло-серого до красного, а цвет текста будет зависеть от цвета плитки.

Создадим класс Tile описывающий одну плитку.  
В нем нам понадобятся:  
1. Поле value типа int, уровень доступа по умолчанию.  
2. Конструктор с параметром, инициализирующий поле value.  
3. Конструктор без параметров (*значение поля*value*должно быть равно нулю*).  
4. Метод isEmpty, возвращающий true в случае, если значение поля value равно 0, иначе — false.  
5. Метод getFontColor, возвращающий новый цвет(*объект типа Color*) (0x776e65) в случае, если вес плитки меньше 16, иначе — 0xf9f6f2.  
6. Метод getTileColor, возвращающий цвет плитки в зависимости от ее веса в соответствии с нижеприведенными значениями:  
0: (0xcdc1b4);  
2: (0xeee4da);  
4: (0xede0c8);  
8: (0xf2b179);  
16: (0xf59563);  
32: (0xf67c5f);  
64: (0xf65e3b);  
128: (0xedcf72);  
256: (0xedcc61);  
512: (0xedc850);  
1024: (0xedc53f);  
2048: (0xedc22e);

для любых других значений: (0xff0000).

Вышеперечисленные методы не должны быть приватными.

Требования:  
1. В классе Tile должно присутствовать поле value типа int с уровнем доступа по умолчанию.  
2. Конструктор класса Tile с одним параметром типа int должен инициализировать поле value.  
3. После создания объекта типа Tile с помощью конструктора без параметров, значение поля value должно быть равно нулю.  
4. Метод isEmpty должен возвращать true в случае, если значение поля value равно 0, иначе - false.  
5. Метод getFontColor должен быть реализован в соответствии с условием задачи.  
6. Метод getTileColor должен возвращать цвет плитки в зависимости от ее веса.

### 2048 (3)

Займемся реализацией класса **Model**. Он будет ответственен за все манипуляции производимые с игровым полем.

Но чтобы как-то манипулировать игровым полем, неплохо было бы для начала его создать!

**Нам понадобятся:**  
1. Приватная константа FIELD\_WIDTH **=** 4, определяющая ширину игрового поля.  
2. Приватный двумерный массив **gameTiles** состоящий из объектов класса **Tile**.  
3. Конструктор без параметров инициализирующий игровое поле и заполняющий его пустыми плитками.

Требования:  
1. В классе Model должно быть создано private static final поле FIELD\_WIDTH со значением равным четырем.  
2. В классе Model должно быть создано private поле gameTiles типа Tile[][].  
3. Конструктор без параметров класса Model должен заполнять массив gameTiles новыми объектами типа Tile.  
4. Массив gameTiles должен иметь размерность FIELD\_WIDTH x FIELD\_WIDTH.

### 2048 (4)

Игра 2048 начинается на поле, где две плитки уже имеют какие-то начальные значения. А наше поле пока пусто :(.

Прежде чем бросаться писать код, давай подумаем как это можно было бы реализовать.

Предлагаю создать приватный метод addTile, который будет смотреть какие плитки пустуют и менять вес одной из них,

выбранной случайным образом, на 2 или 4 (на 9 двоек должна приходиться 1 четверка). Получить случайный объект из списка

можешь использовав следующее выражение: (размерСписка \* случайноеЧислоОтНуляДоЕдиницы).

Также получение свободных плиток можно вынести в отдельный приватный метод getEmptyTiles, возвращающий список

свободных плиток в массиве gameTiles.

После реализации функционала добавления новых плиток, добавим в конструктор два вызова метода addTile,

выполняя начальное условие задачи.

P.S. Пожалуй стоит весь код из конструктора переместить в метод resetGameTiles, для того, чтобы при необходимости

начать новую игру, не приходилось создавать новую модель, а можно было бы просто вернуться в начальное состояние

вызвав его. Уровень доступа должен быть шире приватного.

P.P.S. Для вычисления веса новой плитки используй выражение (Math.random() < 0.9 ? 2 : 4).

Требования:  
1. Метод getEmptyTiles должен возвращать список пустых плиток в массиве gameTiles.  
2. Метод addTile должен изменять значение случайной пустой плитки в массиве gameTiles на 2 или 4 с вероятностью 0.9 и 0.1 соответственно.  
3. Метод resetGameTiles должен заполнять массив gameTiles новыми плитками и менять значение двух из них с помощью двух вызовов метода addTile.  
4. В конструкторе класса Model должен содержаться вызов метода resetGameTiles.