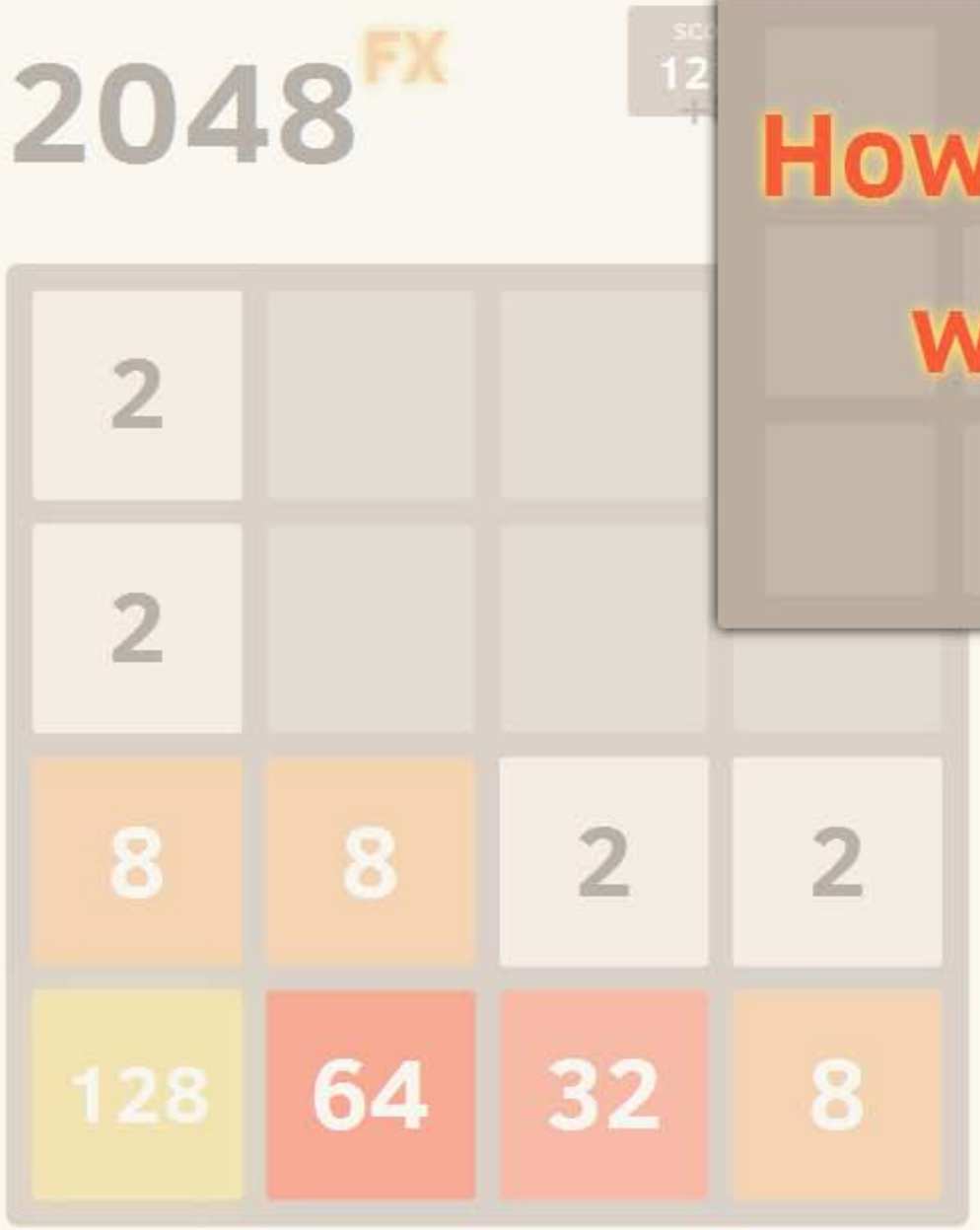


How to Build the Game 2048 with JavaFX and Java 8

Lessons Learned

JavaOne 2014 - CON2710

Bruno Borges ORACLE

José Pereda UVa Universidad de Valladolid

2048FX

Index What are we going to talk about

- Introductions
- The Game
- Building the JavaFX version
- Playing the Game
- Conclusions

Introductions

Bruno **Borges**

- Principal Product Manager
- Java Evangelist
- Oracle Latin America
- http://blog.brunoborges.com.br
- @brunoborges

José Pereda

- PhD, Structural Engineer
- Universidad de Valladolid, Spain
- JavaFX 8 Introduction by Example
- http://jperedadnr.blogspot.com
- @JPeredaDnr

2048 - The Game

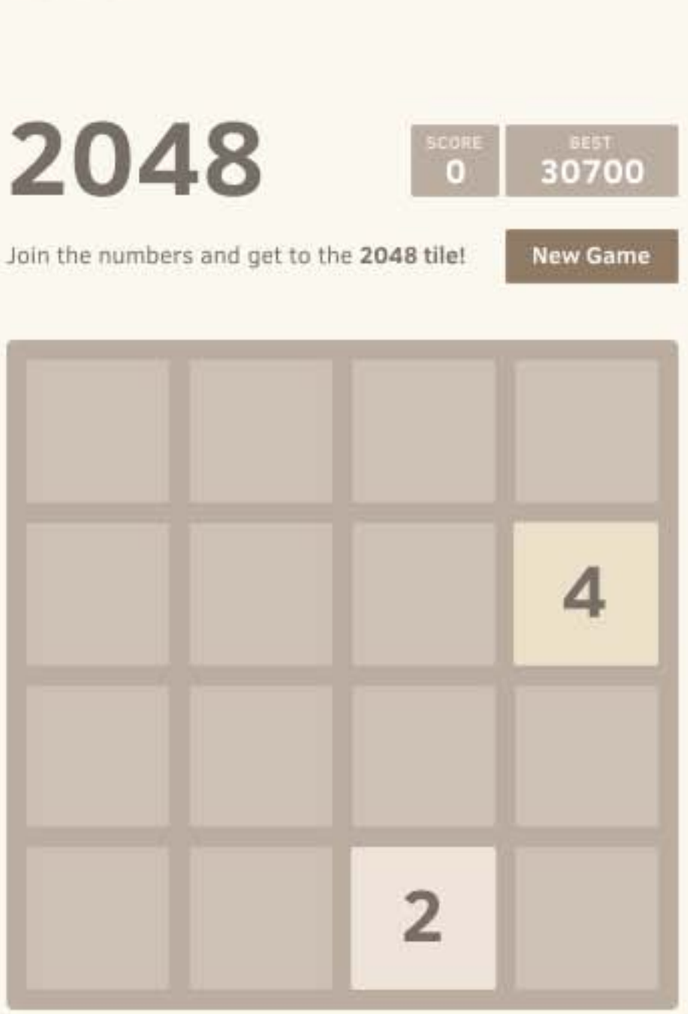
2048FX

Popular Game created by Gabriele Cirulli

2048 was built using HTML, CSS and JavaScript

Released as OSS under the MIT license

- Move numbers in a 4x4 grid
- When equal numbers clash while moving the blocks (up/down/left/right), they merge and numbers are added up
- You win when you get a number in a box with 2048 (match two blocks with 1024 each)



Playing 2048

2048 can be played every where!

Besides the web version, Cirulli has also iOS and Android versions

With JavaFX we can play now on:

- PCs and Laptops
- Tablets and Smartphones running Java SE 8
- Raspberry Pi and embedded devices with UI support

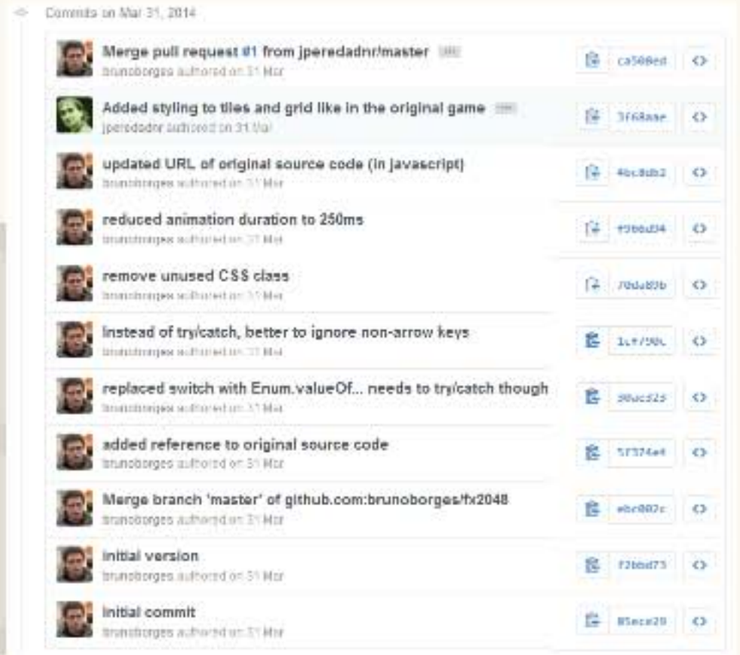
The story behind 2048(FX)

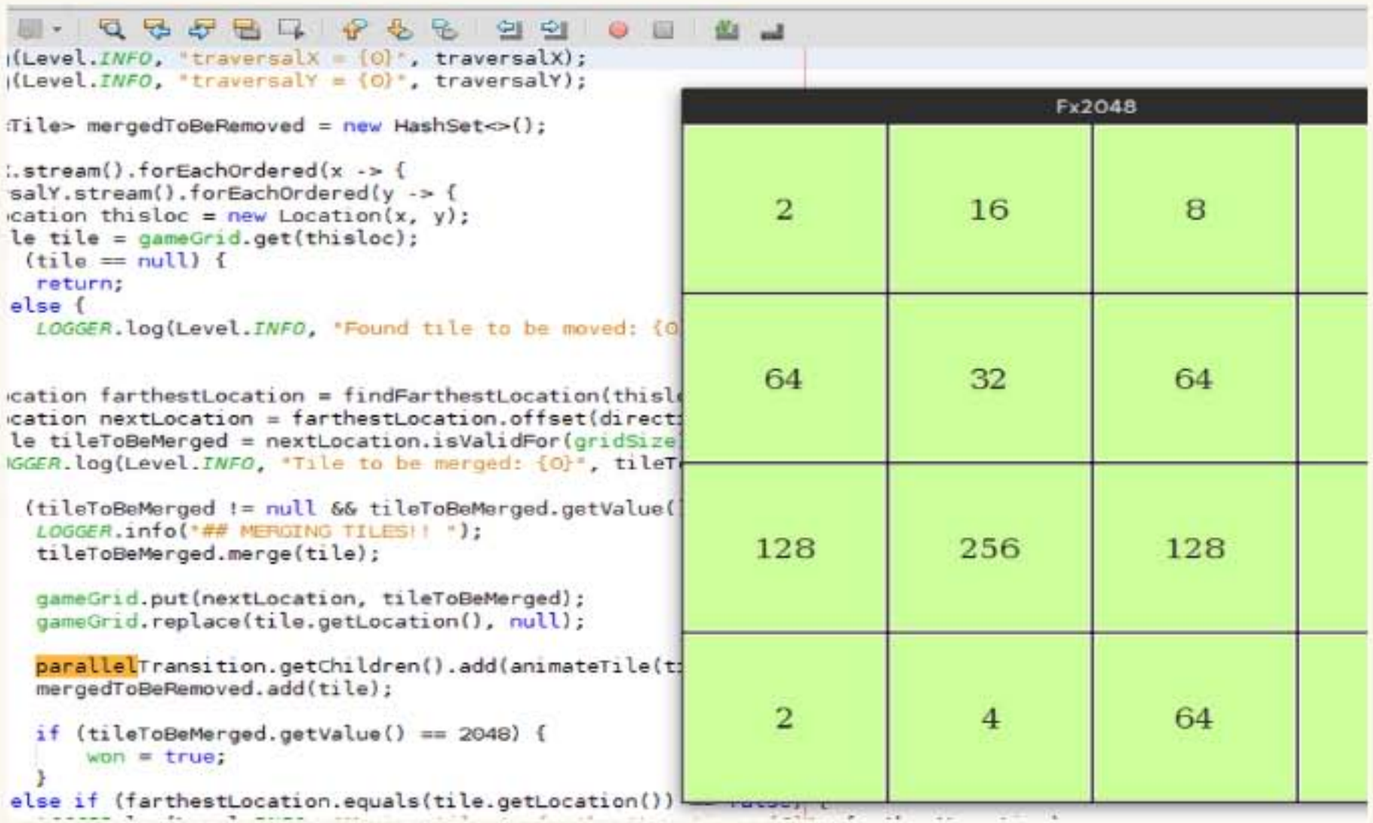
2048FX

Back in March, Bruno started a first version of the game in JavaFX

Totally addicted to the game by that time, Jose jumped in

- https://github.com/brunoborges/fx2048
- Around 60 forks/stars in GitHub
- Around 100 commits





2048FX Classes

Package `game2048` with these classes:

- `Game2048`, the Application class
- `GameManager`, contains the board and grid operator
- `Board`, contains the labels, the scores and the grid with the tiles
- `Tile`, a stylized label with a value and a location inside the grid
- `GridOperator`, to traverse the grid
- `Direction`, `Location`, helper classes
- `SessionManager`, `RecordManager`, helper classes

Live Compiling - The basis

TestBase (1)

We use an Enum to define four directions

```
6  import javafx.geometry.Pos;
7  import javafx.scene.Group;
8  import javafx.scene.control.Label;
9  import javafx.scene.input.KeyCode;
10 import javafx.scene.layout.VBox;
11
12 /**
13  *
14  * @author José Pereda
15  */
16 public class TestBase extends Group{
17
18     /*
19     DIRECTION
20     */
21     public enum Direction {
22
23         UP(0, -1), RIGHT(1, 0), DOWN(0, 1), LEFT(-1, 0);
24
25         private final int y;
26         private final int x;
27
28         Direction(int x, int y) {
29             this.x = x;
30             this.y = y;
31         }
32
33         public int getX() { return x; }
34
35
```

Compilation done 19:0

```
Location{x=2, y=1}

Location{x=3, y=1}
```

Live Compiling - The basis

TestBase (2)

Location contains two coordinates x,y

```
36
37          public static Direction valueFor(KeyCode keyCode) {
38              return valueOf(keyCode.name());
39          }
40      }
41
42      /*
43      LOCATION
44      */
45      private class Location {
46
47          private final int x;
48          private final int y;
49
50          public Location(int x, int y) {
51              this.x = x;
52              this.y = y;
53          }
54
55          public int getX() { return x; }
56
57          public int getY() { return y; }
58
59          public double getLayoutY(int CELL_SIZE) {
60              if (y == 0) {
61                  return CELL_SIZE / 2;
62              }
63              return (y * CELL_SIZE) + CELL_SIZE / 2;
64
```

Compilation done 49:0

Location{x=2, y=1}

Location{x=3, y=1}

Live Compiling - The basis

TestBase (3)

Tile is a Label with a Value and a Location

```
12  import javafx.scene.layout.VBox;
13
14  /**
15   *
16   * @author José Pereda
17   */
18  public class TestBase extends Group{
19
20      private class Tile extends Label {
21
22          private Integer value;
23          private Location location;
24          private Boolean merged;
25
26          public Tile(Integer value) {
27              final int squareSize = Board.CELL_SIZE - 13;
28              setMinSize(squareSize, squareSize);
29              setMaxSize(squareSize, squareSize);
30              setPrefSize(squareSize, squareSize);
31              setAlignment(Pos.CENTER);
32
33              this.value = value;
34              this.merged = false;
35              setText(value.toString());
36              getStyleClass().addAll("game-label", "game-tile-" + value);
37          }
38
39          public void merge(Tile another) {
40
```

Compilation done 25:0

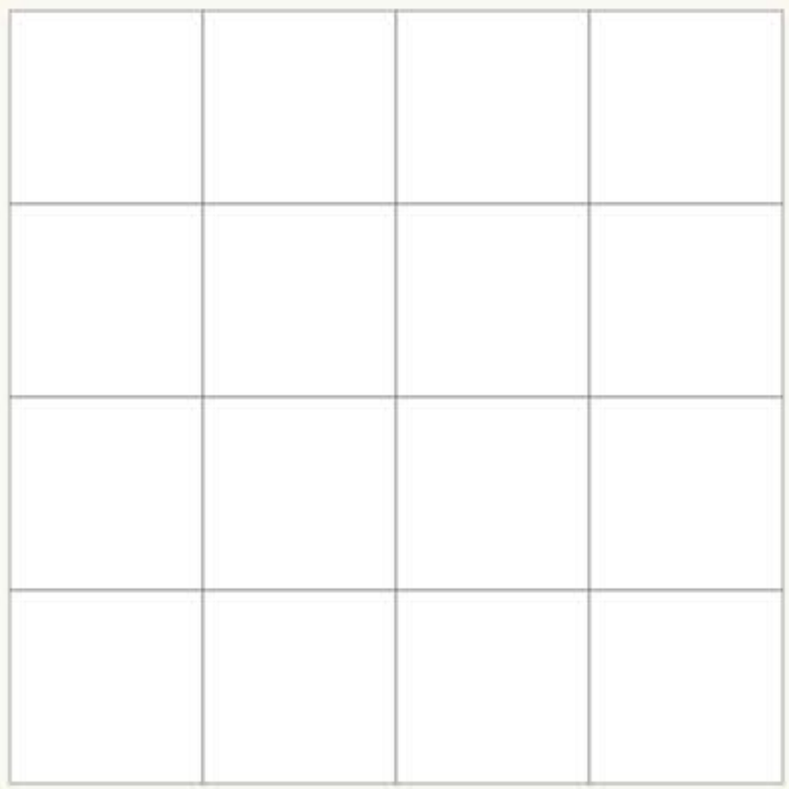
Live Compiling - The Board

TestBoard (1)

Creating a 4x4 grid with an old fashioned normal nested for loop

```
38
39      }
40
41      private Rectangle createCell(int i, int j){
42          Rectangle cell = new Rectangle(i * CELL_SIZE, j * CELL_SIZE, CELL_SI
43          cell.setFill(Color.WHITE);
44          cell.setStroke(Color.GREY);
45          if(step>=10){
46              cell.setArcHeight(CELL_SIZE/6d);
47              cell.setArcWidth(CELL_SIZE/6d);
48              cell.getStyleClass().add("game-grid-cell");
49          }
50          return cell;
51      }
52      private void createGrid() {
53          // old fashion
54          for(int i=0; i<4; i++){
55              for(int j=0; j<4; j++){
56                  gridGroup.getChildren().add(createCell(i, j));
57              }
58          }
59
60          if(step>=10){
61              gridGroup.getStyleClass().add("game-grid");
62              hBottom.getStyleClass().add("game-backGrid");
63          }
64
65          gridGroup.setManaged(false);
66          gridGroup.setLayoutX(BORDER_WIDTH);
67
```

Compilation done 52:0



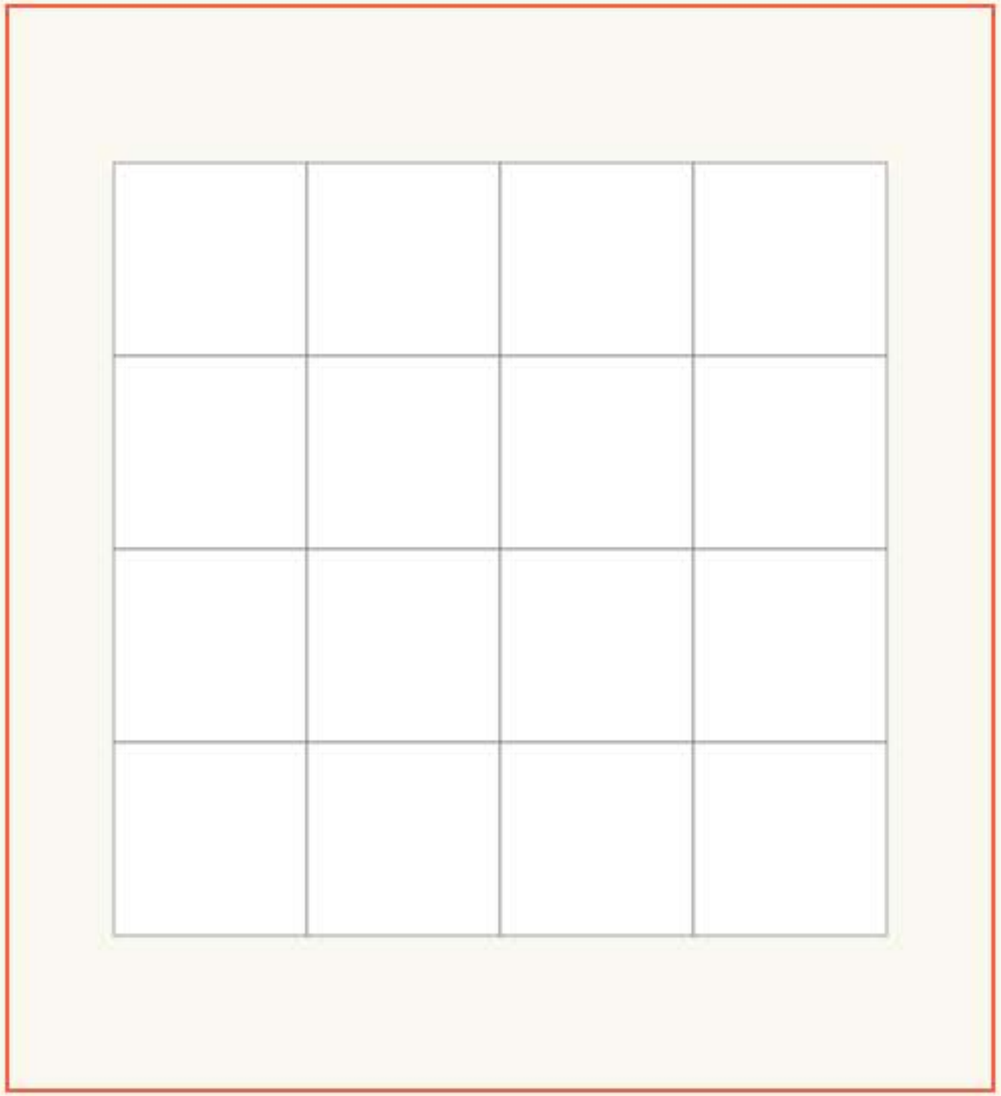
Live Compiling - The Board

TestBoard (2)

Creating a 4x4 grid with a nested enhanced for-each loop

```
40
41      private Rectangle createCell(int i, int j){
42          Rectangle cell = new Rectangle(i * CELL_SIZE, j * CELL_SIZE, CELL_SI
43          cell.setFill(Color.WHITE);
44          cell.setStroke(Color.GREY);
45          if(step>=10){
46              cell.setArcHeight(CELL_SIZE/6d);
47              cell.setArcWidth(CELL_SIZE/6d);
48              cell.getStyleClass().add("game-grid-cell");
49          }
50          return cell;
51      }
52      private void createGrid() {
53          // old fashion
54          List<Integer> intX = Arrays.asList(0,1,2,3);
55          List<Integer> intY = Arrays.asList(0,1,2,3);
56          for(Integer i:intX){
57              for(Integer j:intY){
58                  gridGroup.getChildren().add(createCell(i, j));
59              }
60          }
61
62          if(step>=10){
63              gridGroup.getStyleClass().add("game-grid");
64              hBottom.getStyleClass().add("game-backGrid");
65          }
66
67          gridGroup.setManaged(false);
68
```

Compilation done 52:0

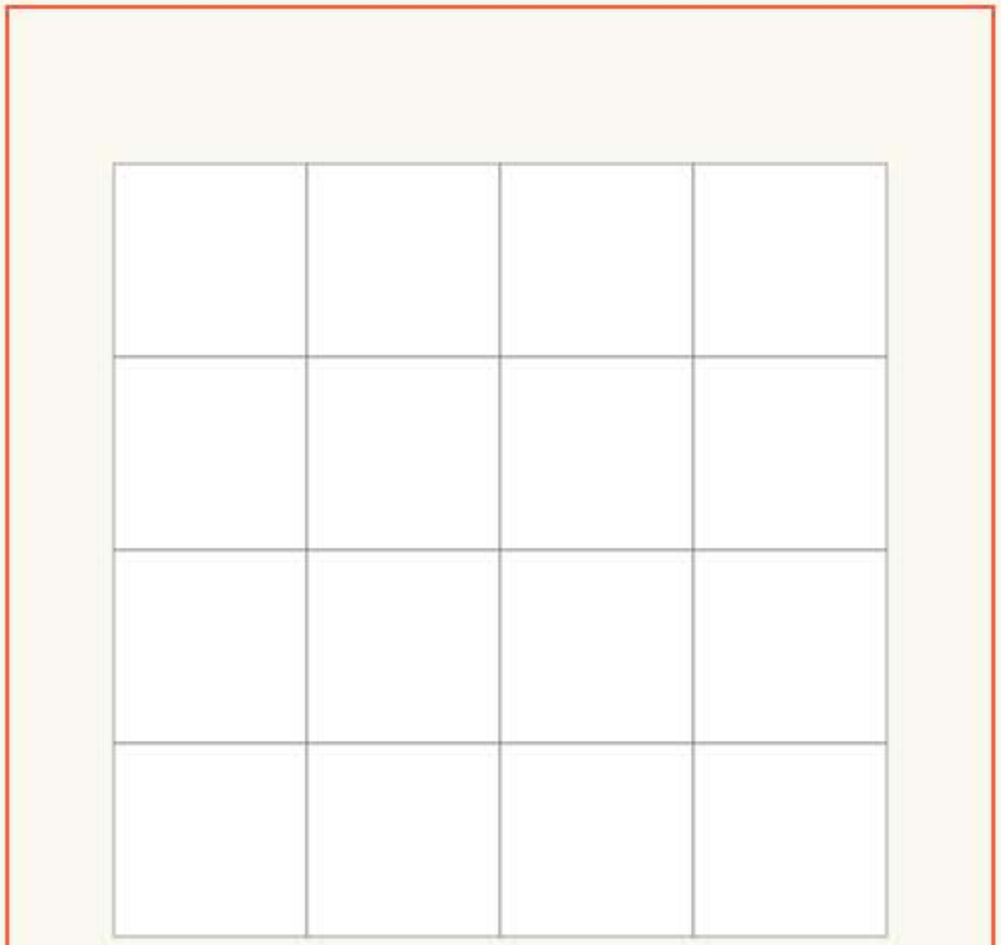


Live Compiling - The Board

TestBoard (3)

Creating a 4x4 grid with Java 8 IntStream (int primitive specialization of Stream) mapped to a nested Stream of Rectangles

```
40
41      private Rectangle createCell(int i, int j){
42          Rectangle cell = new Rectangle(i * CELL_SIZE, j * CELL_SIZE, CELL_SI
43          cell.setFill(Color.WHITE);
44          cell.setStroke(Color.GREY);
45          if(step>=10){
46              cell.setArcHeight(CELL_SIZE/6d);
47              cell.setArcWidth(CELL_SIZE/6d);
48              cell.getStyleClass().add("game-grid-cell");
49          }
50          return cell;
51      }
52      private void createGrid() {
53          // Java 8
54          IntStream.range(0, 4)
55              .mapToObj(i->IntStream.range(0,4)
56                  .mapToObj(j->createCell(i, j))) // Stream<Stream<Rectangle>>
57              .flatMap(Stream::distinct) // flatMap maps and flatten, giving S
58              .forEach(gridGroup.getChildren()::add);
59
60          if(step>=10){
61              gridGroup.getStyleClass().add("game-grid");
62              hBottom.getStyleClass().add("game-backGrid");
63          }
64
65          gridGroup.setManaged(false);
66          gridGroup.setLayoutX(BORDER_WIDTH);
67          gridGroup.setLayoutY(BORDER_WIDTH);
68
Compilation done                                                    52:0
```



Live Compiling - The Board

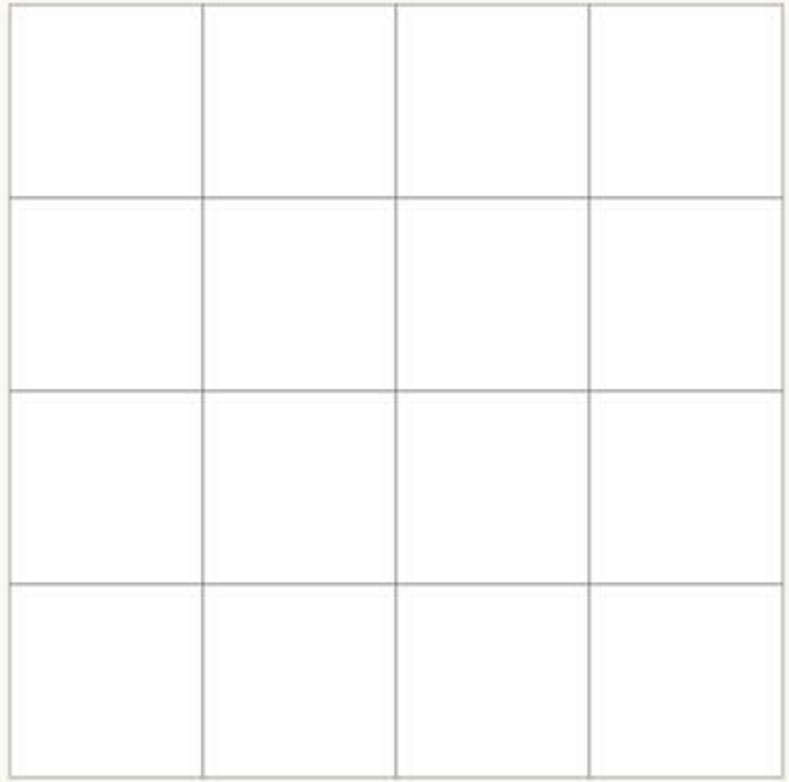
2048FX

TestBoard (4)

Creating a 4x4 grid with Java 8 IntStream and a terminal operation, forEach, that performs an action for each element of the stream

```
40
41      private Rectangle createCell(int i, int j){
42          Rectangle cell = new Rectangle(i * CELL_SIZE, j * CELL_SIZE, CELL_SI
43          cell.setFill(Color.WHITE);
44          cell.setStroke(Color.GREY);
45          if(step>=10){
46              cell.setArcHeight(CELL_SIZE/6d);
47              cell.setArcWidth(CELL_SIZE/6d);
48              cell.getStyleClass().add("game-grid-cell");
49          }
50          return cell;
51      }
52      private void createGrid() {
53          // Java 8
54          IntStream.range(0, 4).boxed() // stream of integers instead of IntSt
55              .forEach(i->IntStream.range(0,4).boxed()
56                  .forEach(j->gridGroup.getChildren().add(createCell(i, j))));
57
58          if(step>=10){
59              gridGroup.getStyleClass().add("game-grid");
60              hBottom.getStyleClass().add("game-backGrid");
61          }
62
63          gridGroup.setManaged(false);
64          gridGroup.setLayoutX(BORDER_WIDTH);
65          gridGroup.setLayoutY(BORDER_WIDTH);
66
67          hBottom.setMinSize(GRID_WIDTH, GRID_WIDTH);
68
```

Compilation done 52:0



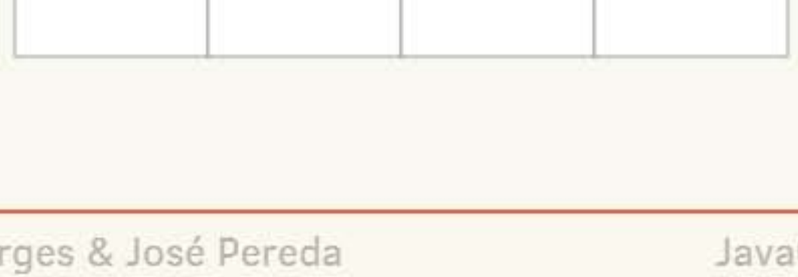
Live Compiling - The Board

TestBoard (5)

Creating a 4x4 grid with Java 8 IntStream and an Iterable forEach, that performs in order an action for each element of the Iterable

```
40
41      private Rectangle createCell(int i, int j){
42          Rectangle cell = new Rectangle(i * CELL_SIZE, j * CELL_SIZE, CELL_SI
43          cell.setFill(Color.WHITE);
44          cell.setStroke(Color.GREY);
45          if(step>=10){
46              cell.setArcHeight(CELL_SIZE/6d);
47              cell.setArcWidth(CELL_SIZE/6d);
48              cell.getStyleClass().add("game-grid-cell");
49          }
50          return cell;
51      }
52      private void createGrid() {
53          // Java 8
54          List<Integer> traversalX = IntStream.range(0, 4).boxed().collect(Col
55          List<Integer> traversalY = IntStream.range(0, 4).boxed().collect(Col
56          traversalX.forEach(i->
57              traversalY.forEach(j->
58                  gridGroup.getChildren().add(createCell(i, j))));
59
60          if(step>=10){
61              gridGroup.getStyleClass().add("game-grid");
62              hBottom.getStyleClass().add("game-backGrid");
63          }
64
65          gridGroup.setManaged(false);
66          gridGroup.setLayoutX(BORDER_WIDTH);
67          gridGroup.setLayoutY(BORDER_WIDTH);
68
```

Compilation done 52:0



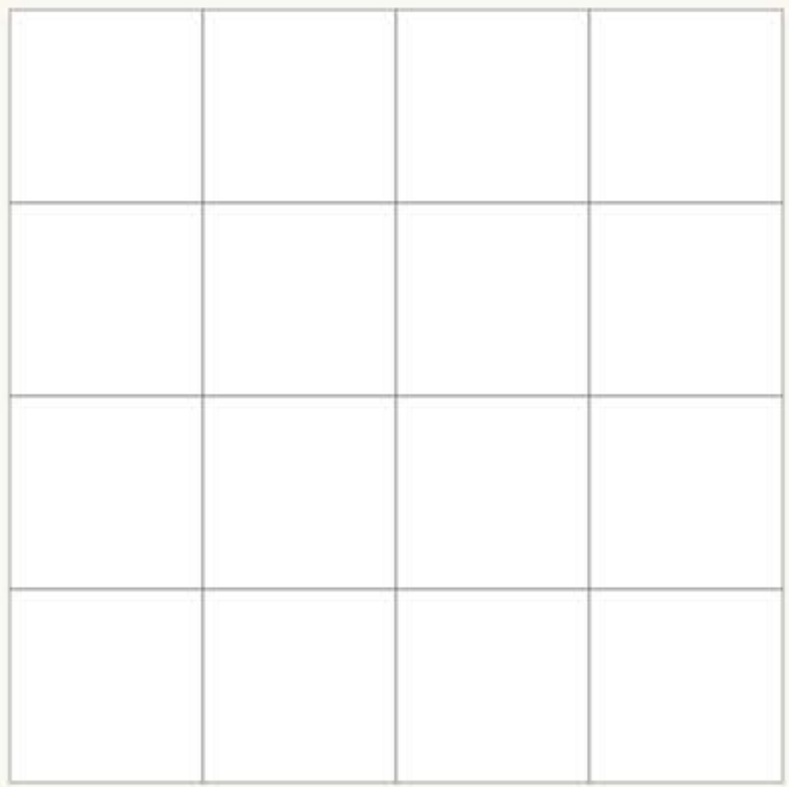
Live Compiling - The Board

TestBoard (6)

Creating a 4x4 grid with traverseGrid, a "functional array", based on a binary operator applied to every element of the grid

```
50          }
51          return cell;
52      }
53
54      private final List<Integer> traversalX = IntStream.range(0, 4).boxed().c
55      private final List<Integer> traversalY = IntStream.range(0, 4).boxed().c
56
57      public void traverseGrid(IntBinaryOperator func) {
58          traversalX.forEach(x ->
59              traversalY.forEach(y -> func.applyAsInt(x, y)));
60      }
61
62      private void createGrid() {
63          // Java 8
64          traverseGrid((i, j) -> {
65              gridGroup.getChildren().add(createCell(i, j));
66              return 0;
67          });
68
69          if(step>=10){
70              gridGroup.getStyleClass().add("game-grid");
71              hBottom.getStyleClass().add("game-backGrid");
72          }
73
74          gridGroup.setManaged(false);
75          gridGroup.setLayoutX(BORDER_WIDTH);
76          gridGroup.setLayoutY(BORDER_WIDTH);
77
78
```

Compilation done 62:0



Live Compiling - The GameManager

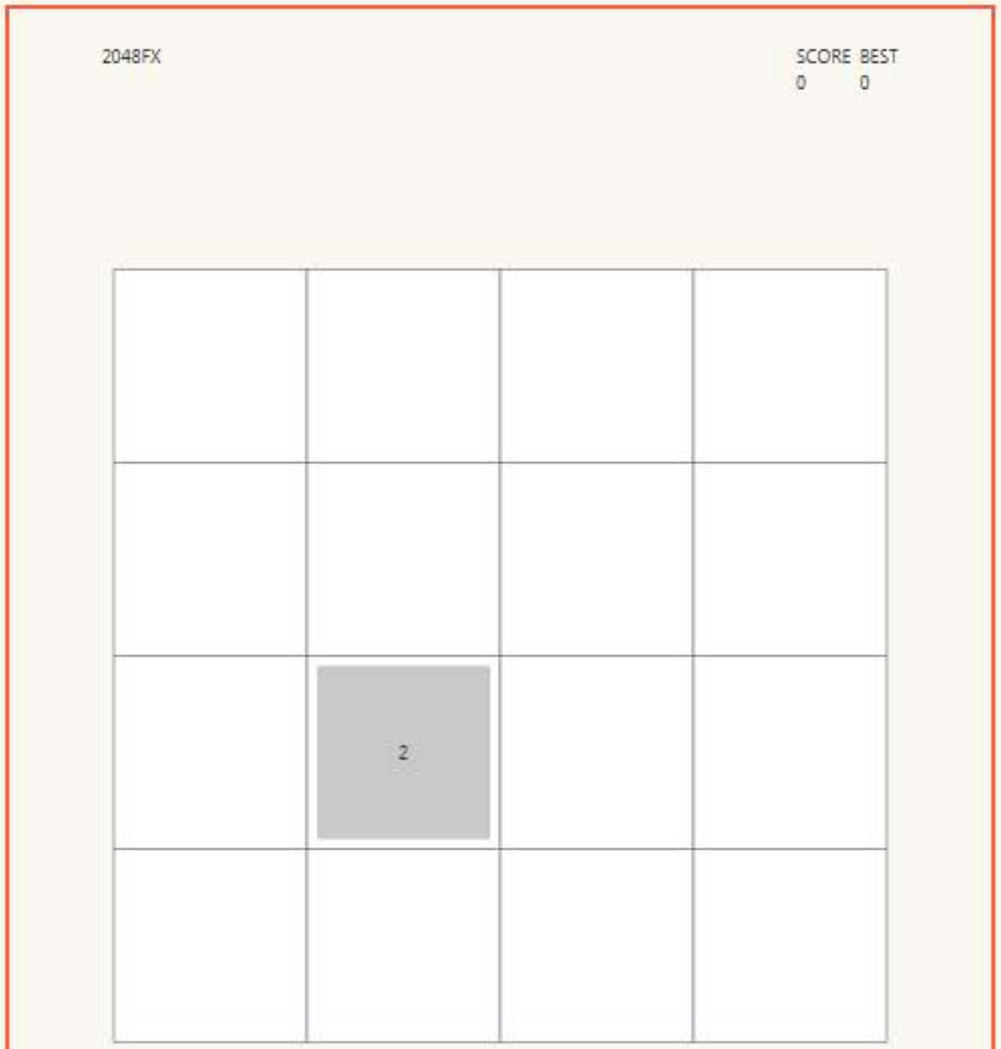
2048FX

TestGameManager (1)

We add the board and start the game by adding a tile at any position

```
22  import javafx.scene.Scene;
23
24  public class TestGameManager extends Group{
25
26      private final Group gameManager=new Group();
27      private final int step=9;
28
29      private Board board;
30      private final List<Location> locations = new ArrayList<>();
31      private final Map<Location, Tile> gameGrid = new HashMap<>();
32      private final ParallelTransition parallelTransition = new ParallelTransi
33      private volatile boolean movingTiles = false;
34      private int tilesWereMoved=0;
35      private final Set<Tile> mergedToBeRemoved = new HashSet<>();
36
37      public TestGameManager(){
38          board=new Board(step);
39          gameManager.getChildren().add(board);
40          startGame();
41
42
43          addGame();
44      }
45
46      private void initializeGameGrid() {
47
48      }
49
50      private void startGame() {
51
```

Compilation done 36:0



Live Compiling - The GameManager

2048FX

TestGameManager (2)

First approach to move tiles: get a list of tiles, remove them, and add them at an offset valid location

```
49      }
50
51      private void startGame() {
52          Tile tile0 = Tile.newRandomTile(step);
53          tile0.setLocation(new Location(1,2));
54          board.addTile(tile0);
55      }
56
57      private void redrawTilesInGameGrid() {
58
59      }
60
61      public void move(Direction direction) {
62          // Get a list of tiles, remove them from the board,
63          // create new tiles at an offset location if valid (limits, no other
64          List<Tile> tiles=board.getGridGroup().getChildren().stream()
65                  .filter(g->g instanceof Tile).map(t->(Tile)t)
66                  .collect(Collectors.toList());
67          board.getGridGroup().getChildren().removeAll(tiles);
68          tiles.forEach(t->{
69              Tile newTile = Tile.newTile(t.getValue(),step);
70              final Location newLoc=t.getLocation().offset(direction);
71              if(newLoc.isValidFor() && !tiles.stream().filter(t2->t2.getLocat
72                  newTile.setLocation(newLoc);
73              } else {
74                  newTile.setLocation(t.getLocation());
75              }
76              board.addTile(newTile);
77
```

Compilation done 62:0

Live Compiling - The GameManager

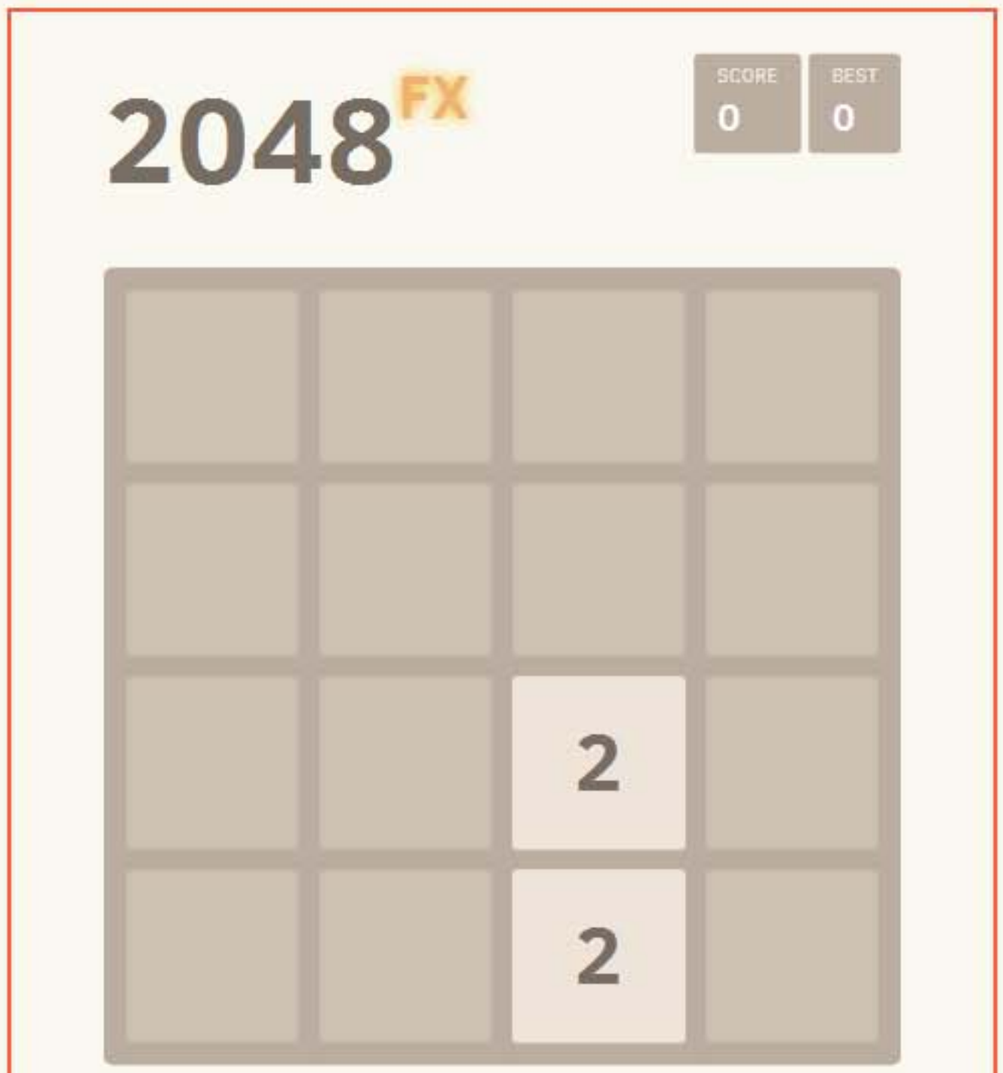
2048FX

TestGameManager (3)

Now to start the game we get a real random location and a random value for the tile (90% 2, 10% 4)

```
56    private void initializeGameGrid() {
57        gameGrid.clear();
58        locations.clear();
59        for(int i=0; i<4; i++){
60            for(int j=0; j<4; j++){
61                Location location = new Location(i,j);
62                locations.add(location);
63                gameGrid.put(location, null);
64            }
65        }
66    }
67
68    private void startGame() {
69        Tile tile0 = Tile.newRandomTile(step);
70        List<Location> locCopy=locations.stream().collect(Collectors.toList(
71        Collections.shuffle(locCopy);
72        tile0.setLocation(locCopy.get(0));
73        gameGrid.put(tile0.getLocation(), tile0);
74        Tile tile1 = Tile.newRandomTile(step);
75        tile1.setLocation(locCopy.get(1));
76        gameGrid.put(tile1.getLocation(), tile1);
77
78        redrawTilesInGameGrid();
79    }
80
81    private void redrawTilesInGameGrid() {
82         gameGrid.values().stream().filter(Objects::nonNull).forEach(board::
83    }
84
```

Compilation done 68:0



Live Compiling - The GameManager

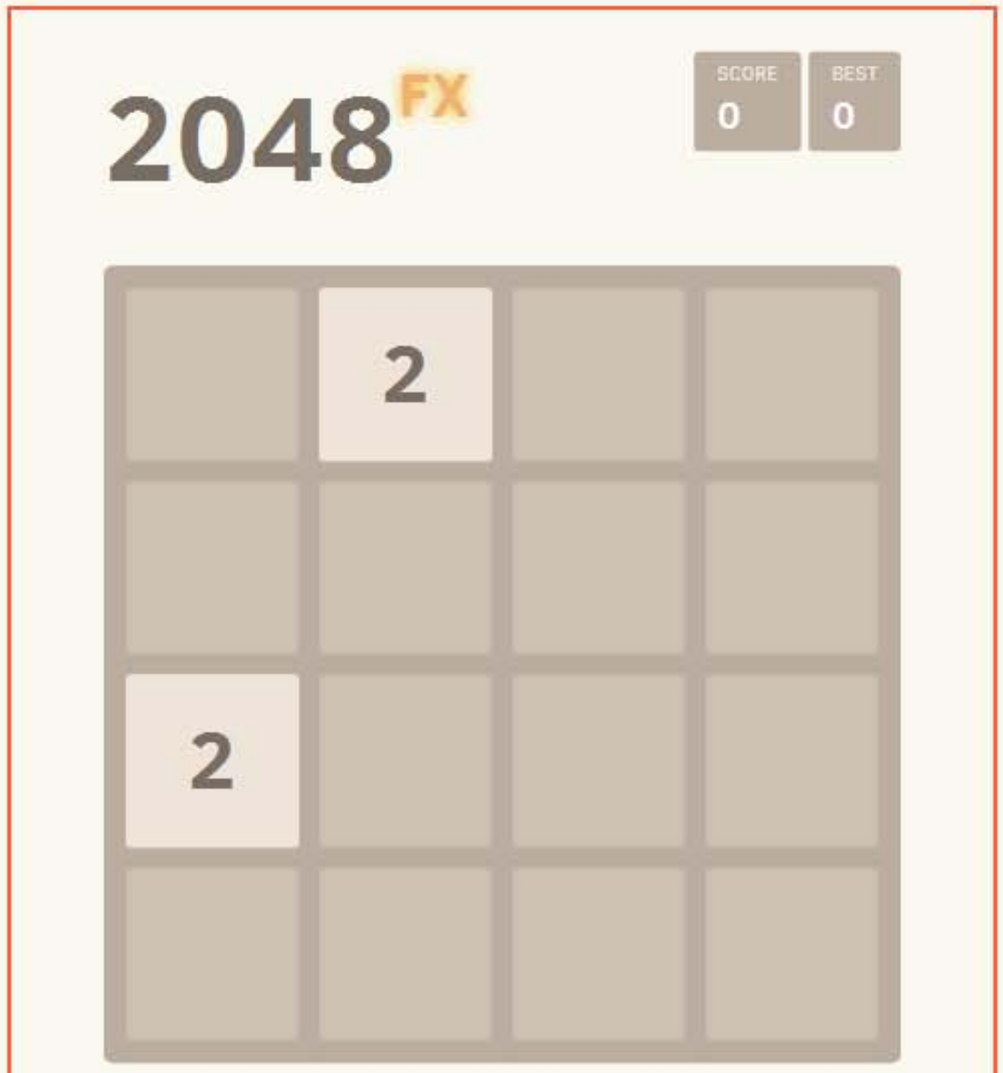
2048FX

TestGameManager (4)

New approach to move tiles: sort lists according direction and use traverseGrid. Every valid tile will be moved to the furthest location available.

```
77          gameGrid.put(tile1.getLocation(), tile1);
78
79          redrawTilesInGameGrid();
80      }
81
82      private void redrawTilesInGameGrid() {
83          gameGrid.values().stream().filter(Objects::nonNull).forEach(board:
84      }
85
86      public void move(Direction direction) {
87          synchronized (gameGrid) {
88              if (movingTiles) {
89                  return;
90              }
91          }
92          GridOperator.sortGrid(direction);
93          tilesWereMoved = GridOperator.traverseGrid((i,j)->{
94              Tile t=gameGrid.get(new Location(i,j));
95              if(t!=null){
96                  final Location newLoc=findFarthestLocation(t.getLocation(),
97                  if(!newLoc.equals(t.getLocation())){
98                      parallelTransition.getChildren().add(animateExistingTil
99                      gameGrid.put(newLoc, t);
100                     gameGrid.replace(t.getLocation(),null);
101                     t.setLocation(newLoc);
102                     return 1;
103                 }
104             }
105
```

Compilation done 90:0



Live Compiling - The GameManager

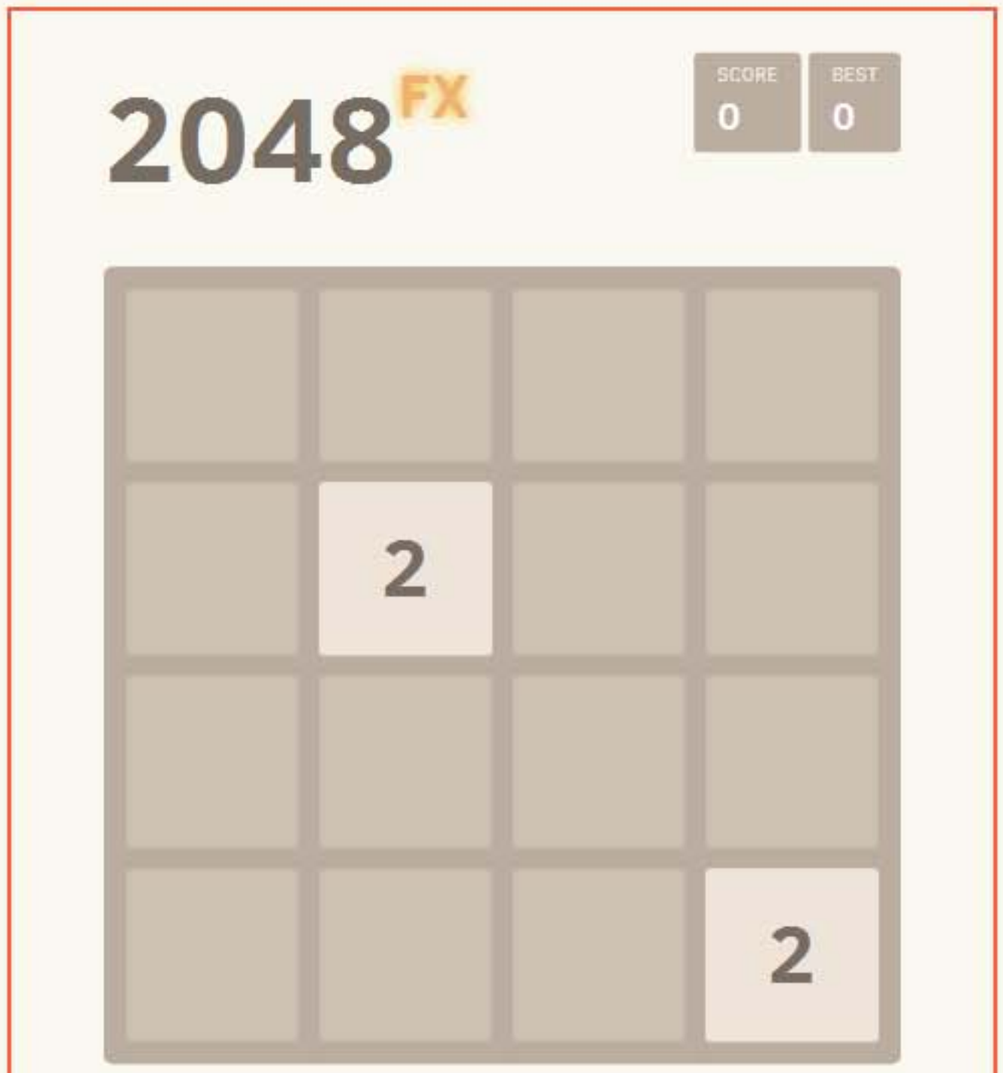
2048FX

TestGameManager (5)

And we try to move them one more step to see if two tiles are mergeable.

```
83          gameGrid.values().stream().filter(Objects::nonNull).forEach(board:
84      }
85
86      public void move(Direction direction) {
87          synchronized (gameGrid) {
88              if (movingTiles) {
89                  return;
90              }
91          }
92          GridOperator.sortGrid(direction);
93          board.setPoints(0);
94          tilesWereMoved = GridOperator.traverseGrid((i,j)->{
95              Tile t=gameGrid.get(new Location(i,j));
96              if(t!=null){
97                  final Location newLoc=findFarthestLocation(t.getLocation(),
98                  Location nextLocation = newLoc.offset(direction);
99                  Tile tileToBeMerged = nextLocation.isValidFor() ? gameGrid.
100                 if (tileToBeMerged != null && !tileToBeMerged.isMerged() &&
101                     tileToBeMerged.merge(t);
102                     tileToBeMerged.toFront();
103                     gameGrid.put(nextLocation, tileToBeMerged);
104                     gameGrid.replace(t.getLocation(), null);
105                     parallelTransition.getChildren().add(animateExistingTil
106                     parallelTransition.getChildren().add(animateMergedTile(
107                     mergedToBeRemoved.add(t);
108                     board.addPoints(tileToBeMerged.getValue());
109                     return 1;
110                 }
111                 if(!newLoc.equals(t.getLocation())){
112
```

Compilation done 95:0



Live Compiling - The GameManager

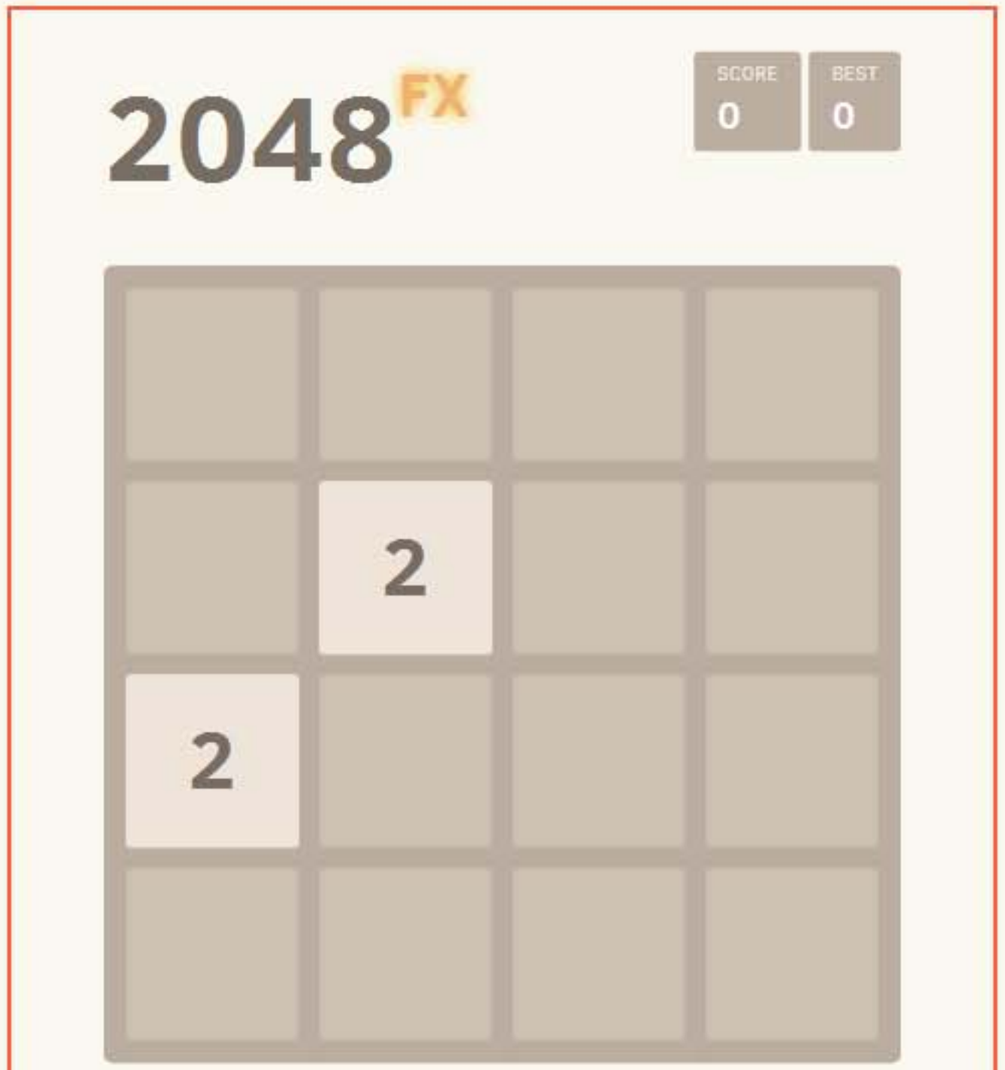
2048FX

TestGameManager (6)

With parallel streams we traverse the grid (up and left) to find possible mergeable tiles and look for game over if the grid is full and there are no pairs

```
214
215     private SequentialTransition animateMergedTile(Tile tile) {
216         final ScaleTransition scale0 = new ScaleTransition(Duration.millis(
217         scale0.setToX(1.2);
218         scale0.setToY(1.2);
219         scale0.setInterpolator(Interpolator.EASE_IN);
220 
221         final ScaleTransition scale1 = new ScaleTransition(Duration.millis(
222         scale1.setToX(1.0);
223         scale1.setToY(1.0);
224         scale1.setInterpolator(Interpolator.EASE_OUT);
225 
226         return new SequentialTransition(scale0, scale1);
227     }
228 
229     private int mergeMovementsAvailable() {
230         final AtomicInteger numMergeableTile = new AtomicInteger();
231         Stream.of(Direction.UP, Direction.LEFT)
232             .parallel()
233             .forEach(direction -> {
234                 GridOperator.traverseGrid((x, y) -> {
235                     Location thisloc = new Location(x, y);
236                     Tile t1=gameGrid.get(thisloc);
237                     if(t1!=null){
238                         Location nextLoc=thisloc.offset(direction);
239                         if(nextLoc.isValidFor()){
240                             Tile t2=gameGrid.get(nextLoc);
241                             if(t2!=null && t1.isMergeable(t2)){
242                                 numMergeableTile.incrementAndGet();
243
```

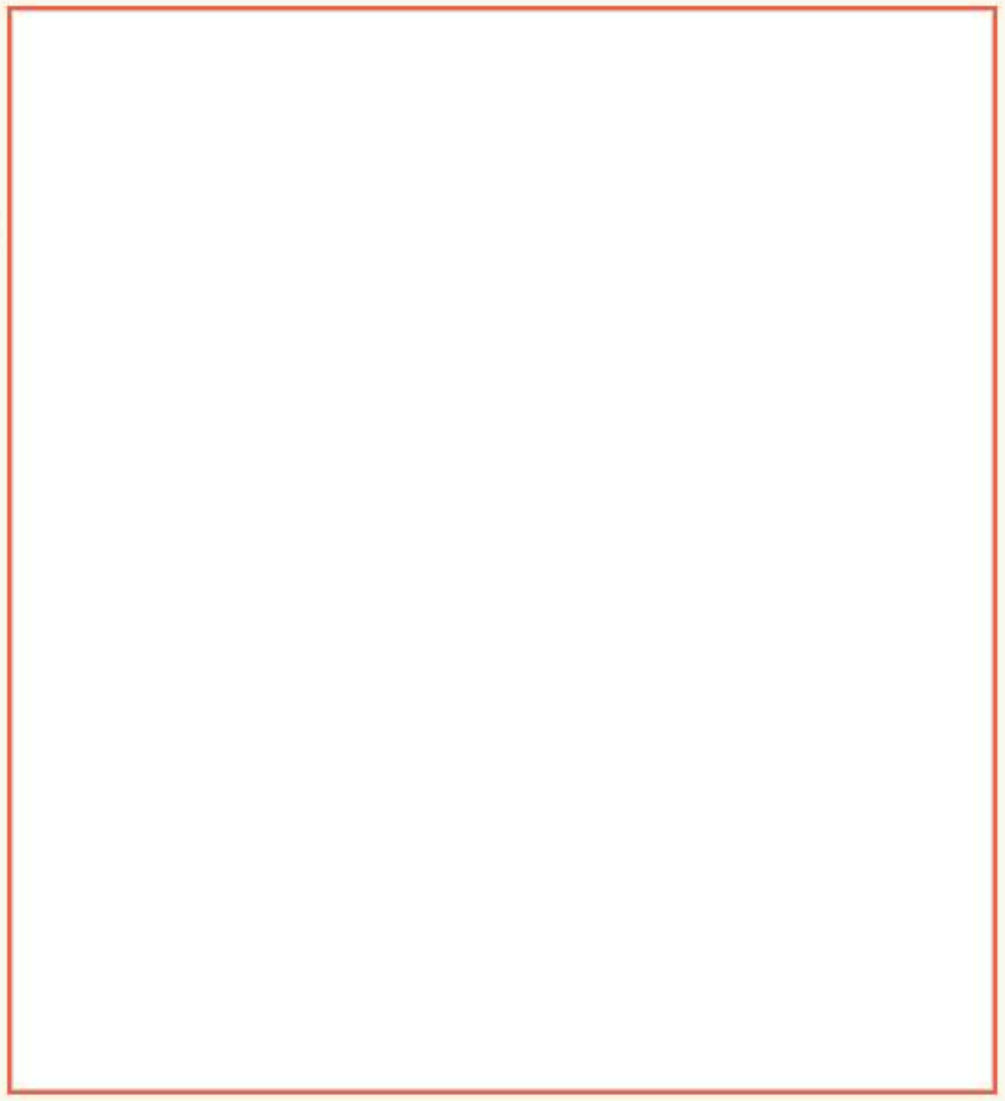
Compilation done 228:0



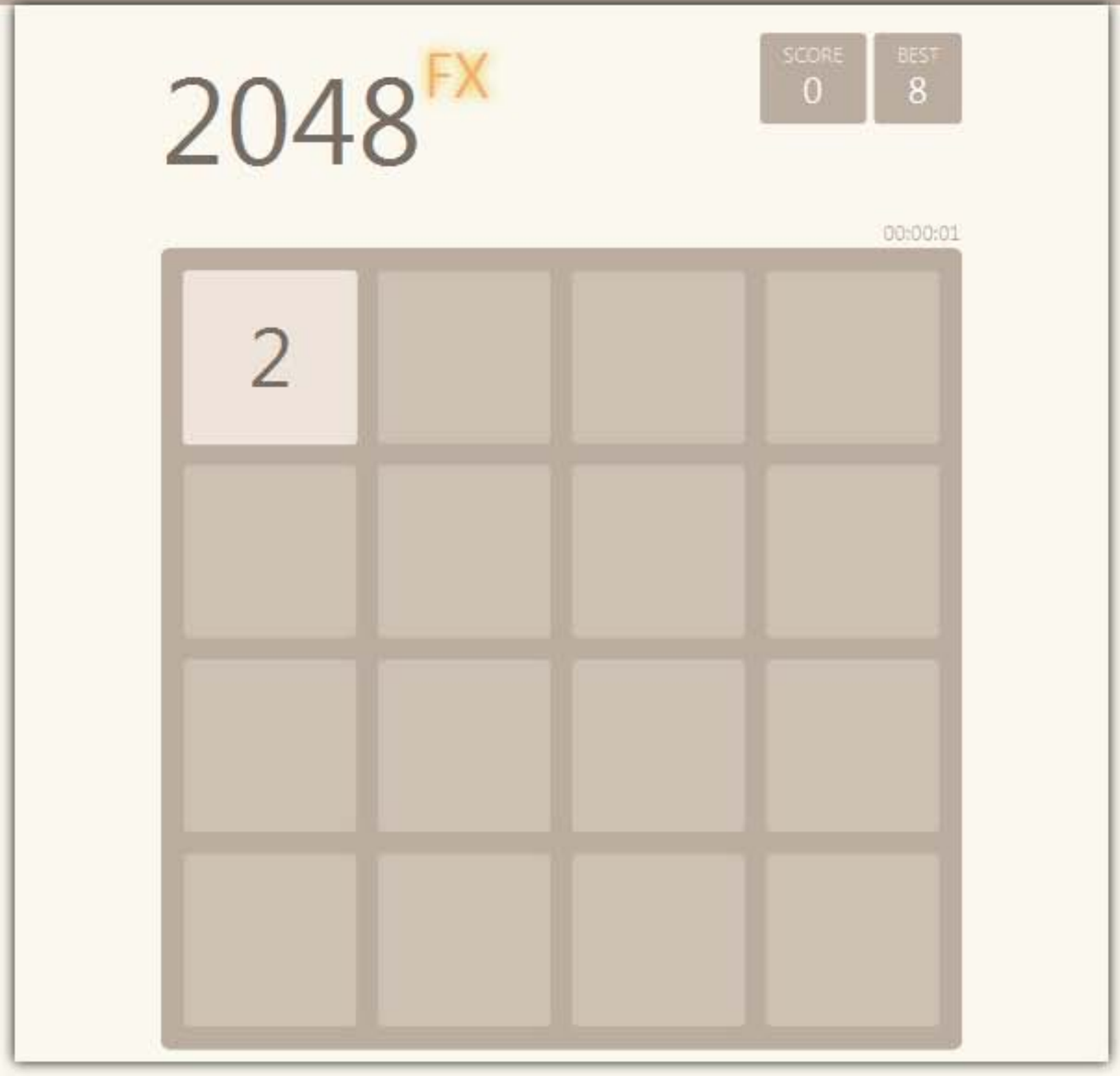
Have you ever won 2048?

- Yes, with more than 1000 points
- Yes, with more than 20000 points
- Yes, with more than 100000 points

Tweet your answer with: #2048FX



Let's play the 2048FX game!



Develping 2048FX

- Java 8 (Lambdas and Streams) are really useful to deal with the grid of tiles and all the operations
- JavaFX animations and styling are suitable for gaming applications
- Out of the box, you can use it on ARM dev
- Let's show how PiDome play with it!

Now?

To learn more:

- Java SE Documentation
- Source code:

http://github.com/brunoborges/fx2048

- Twitter:

@brunoborges @jperedadnr

How to Build the Game 2048 with JavaFX and Java 8

Lessons Learned

JavaOne 2014 - CON2710

Bruno Borges ORACLE

José Pereda UVa Universidad de Valladolid