JAVAONE 2014

Create the Game 2048 with Java 8 and JavaFX [HOL3244]

By developing the famous game 2048 with JavaFX and Java 8 in this hands-on lab session, you will encounter several new language features such as Stream API, lambda expressions, and new util classes and methods.
You will also learn basic concepts of JavaFX 2-D animations, parallel processing, UI design, CSS styling, and more.

Project Base

*Open the repo and have a look at the project:

`https://github.com/jperedadnr/Game2048HOL.git`

* It is a JavaFX application, with the following Java files:

-Game2048

-GameManager

-Board

-Tile

-Location

-Direction

-GridOperator

And resources:

-game.css

-ClearSans-Bold.ttf

Most of them are empty or with empty methods.

The main task is following this guide, to complete the methods and have a fully operative version of the Game 2048^{FX}.

Basically you'll just have to copy and paste the snippets of code available on every step.

Please follow carefully every step. Try to build and run the application every now and then to check you didn't miss any step and everything is in place.

Fill free to ask any questions along the way.

STEP 0. Clone the repository

Open NetBeans 8.0

Under Team->Git select Clone ant type this URL:
https://github.com/jperedadnr/Game2048Empty.git

Accept the default destination folder or choose one.

Select branch and click Next/Finish.

Wait till the local copy is created, and the project is opened in NetBeans.

Select Build and Run to test the (empty) application.

STEP 1

Create instance of GameManager in Game2048 and add it to the root

SOLUTION CODE

```
Game2048.start(){
gameManager=new GameManager();
root.getChildren().add(gameManager);
}
```

Step 2

Create instance of Board in GameManager and add it to it

SOLUTION CODE

```
GameManager(){
board=new Board();
getChildren().add(board);
}
```

STEP 3

Create nodes for hTop in createScore(), called on Board constructor.

In hTop: Labels “2048” and “FX”, empty HBox HGrow always, vScores

In vScores: hScores, VBox vFill, vgrow always

In hScores: vScore, vRecord, separation 5

In VBox vScore: Label “Score”, Label lblScore

In VBox vRecord: Label “Best”, Label lblBest

SOLUTION CODE

```
Board(){
createScore();
}
```

```
createScore() {
        Label lblTitle=new Label("2048");
        Label lblSubtitle=new Label("FX");

        HBox hFill=new HBox();
        HBox.setHgrow(hFill, Priority.ALWAYS);

        VBox vScores = new VBox();
        HBox hScores=new HBox(5);

        Label lblTit = new Label("SCORE");
        vScore.getChildren().addAll(lblTit, lblScore);

        VBox vRecord = new VBox(0);
        Label lblTitBest = new Label("BEST");
        vRecord.getChildren().addAll(lblTitBest, lblBest);

        hScores.getChildren().addAll(vScore,vRecord);
        VBox vFill = new VBox();
        VBox.setVgrow(vFill, Priority.ALWAYS);
        vScores.getChildren().addAll(hScores,vFill);

        hTop.getChildren().addAll(lblTitle, lblSubtitle, hFill,vScores);
}
```

STEP 4

In createCell method create a rectangle in corners i*cell_size, j*cell_size, fill white, border grey

SOLUTION CODE

```
        createCell(){
        cell = new Rectangle(i * CELL_SIZE, j * CELL_SIZE, CELL_SIZE, CELL_SIZE);
        cell.setFill(Color.WHITE);
        cell.setStroke(Color.GREY);
        }
```

STEP 5

In createGrid(), add 4x4 cells to gridGroup, called from constructor

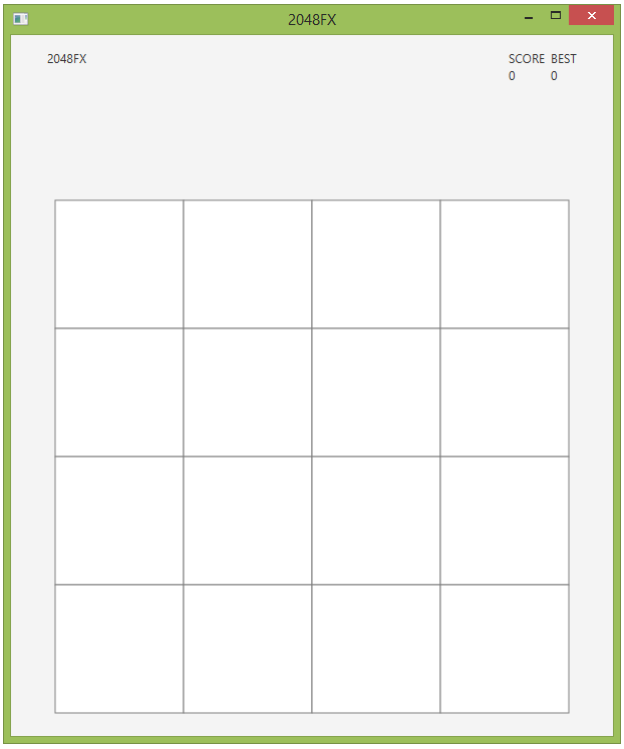
For(i->4),for(j->4), gridGroup add createCell(I,j)

SOLUTION CODE

```
Board(){
createGrid();
```

```
}
createGrid(){
        for(int i=0; i<4; i++){
                for(int j=0; j<4; j++){
                        gridGroup.getChildren().add(createCell(i, j));
                }
        }
}
```

Screenshot after #5



STEP 6

Create Tile in private constructor

Label size cell_size-13; align center, setText(value)

Setstyle background color #c9c9c9;

SOLUTION CODE

```
Tile(value){
        final int squareSize = Board.CELL_SIZE - 13;
        setMinSize(squareSize, squareSize);
        setMaxSize(squareSize, squareSize);
        setPrefSize(squareSize, squareSize);
        setStyle("-fx-background-color: #c9c9c9;");
        setAlignment(Pos.CENTER);
```

```
        this.value = value;
        this.merged = false;
        setText(Integer.toString(value));
}
```

STEP 7

Assign 90% possibilities to random tiles

SOLUTION CODE

```
newRandomTile{
        return newTile(new Random().nextDouble() < 0.9 ? 2 : 4);
}
```

STEP 8

In Board.moveTile method, set tile layout x,y by location.layout x,y – tile minW,h/2. From addTile, call moveTile.

SOLUTION CODE

```
addTile(tile){
        moveTile(tile, tile.getLocation());
}
moveTile(tile){
        double layoutX = tile.getLocation().getLayoutX(CELL_SIZE) - (tile.getMinWidth() /
2);
        double layoutY = tile.getLocation().getLayoutY(CELL_SIZE) - (tile.getMinHeight()
/ 2);

        tile.setLayoutX(layoutX);
        tile.setLayoutY(layoutY);
        }
```

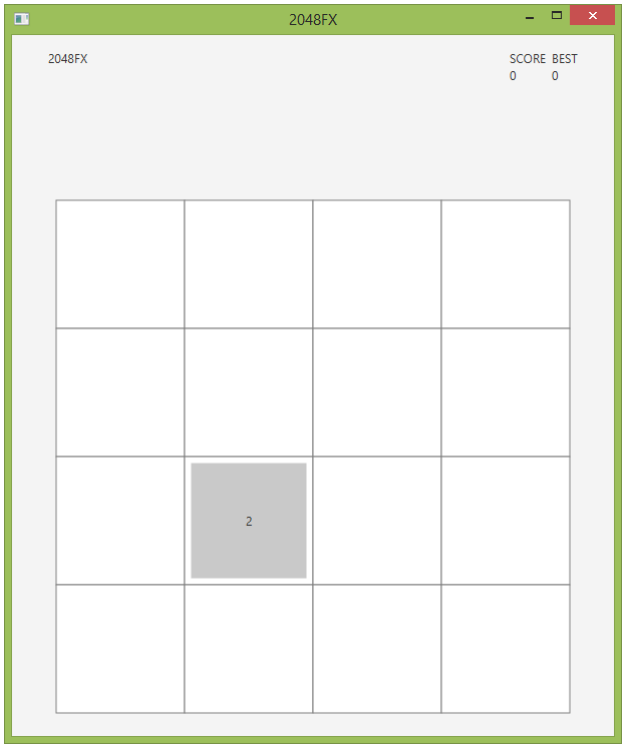
STEP 9

In GameManager.startGame, add random tile at random location

SOLUTION CODE

```
GameManager (){
        startGame();
}
startGame(){
        Tile tile0 = Tile.newRandomTile();
        tile0.setLocation(new Location(1,2));
        board.addTile(tile0);
}
```

Screenshot after #9



STEP 10

In Game2048, load the custom font, enable css styling

Apply all the styles to:

*Game2048->root, game-root

*Tile, game-label, ‘game-tile’+value, and remove hardcode style.

*Board:

-lblTitle, game-label, game-title

-lblSubtitle, game-label, game-subtitle

-vScore y vRecord, game-vbox

-lblScore, game-label, game-score

-lblTit, game-label, game-titScore

-lblTitBest, game-label, game-titScore

-lblBest, game-label, game-score

-Rectangle, game-grid-cell, Adjust arcsize in cell to Cell_size/6;

-GridGroup, game-grid

-hBottom, game-backGrid

SOLUTION CODE

```
init (){
        Font.loadFont(Game2048.class.getResource("ClearSans-Bold.ttf").toExternalForm(),
10.0);
}
start(){
        scene.getStylesheets().add(Game2048.class.getResource("game.css").toExternalForm(
));
        root.getStyleClass().addAll("game-root");
}
createScore(){
        lblTitle.getStyleClass().addAll("game-label","game-title");
        lblSubtitle.getStyleClass().addAll("game-label","game-subtitle");
        vScore.getStyleClass().add("game-vbox");
        lblTit.getStyleClass().addAll("game-label","game-titScore");
        lblScore.getStyleClass().addAll("game-label","game-score");
        vRecord.getStyleClass().add("game-vbox");
        lblTitBest.getStyleClass().addAll("game-label","game-titScore");
        lblBest.getStyleClass().addAll("game-label","game-score");
}
createCell(){
        cell.setArcHeight(CELL_SIZE/6d);
        cell.setArcWidth(CELL_SIZE/6d);
        cell.getStyleClass().add("game-grid-cell");
}
createGrid(){
        gridGroup.getStyleClass().add("game-grid");
        hBottom.getStyleClass().add("game-backGrid");
}
Tile(){
        getStyleClass().addAll("game-label", "game-tile-" + value);
}
```

Screenshot after #10

STEP 11

In Direction, add valueFor static method, from KeyCode get valueOf as Direction

SOLUTION CODE

```
valueFor(){
	return valueOf(keyCode.name());
}
```

STEP 12

In Location, add offset method creating a new Location based on the actual and a direction

SOLUTION CODE

```
offset(direction){
	return new Location(x + direction.getX(), y + direction.getY());
}
```

STEP 13

In GameManager.move, get a list List<Tile> of tiles in the gridGroup, remove all the list from the gridGroup, and create new tiles based in the old ones, with offset(dir) in their location, add them to gridGroup. Check if newLoc isValidFor and if in new location there is no another tile, else keep tile location.

SOLUTION CODE

```
move(direction){
	List<Tile> tiles=board.getGridGroup().getChildren().stream()
		.filter(g->g instanceof Tile).map(t->(Tile)t)
		.collect(Collectors.toList());
```

```
        board.getGridGroup().getChildren().removeAll(tiles);
        tiles.forEach(t->{
                Tile newTile = Tile.newTile(t.getValue());
                final Location newLoc=t.getLocation().offset(dir);
                if(newLoc.isValidFor() && !tiles.stream().filter(t2-
        >t2.getLocation().equals(newLoc)).findAny().isPresent()){
                        newTile.setLocation(newLoc);
                } else {
                        newTile.setLocation(t.getLocation());
                }
                board.addTile(newTile);
        });
}
```

STEP 14

In Game2048, add listener to scene on Key Pressed, get keyCode, check is arrowkey, get Direction and move tile

SOLUTION CODE

```
valueFor(){
        scene.setOnKeyPressed(ke -> {
                KeyCode keyCode = ke.getCode();
                if(keyCode.isArrowKey()){
                        Direction dir = Direction.valueFor(keyCode);
                        gameManager.move(dir);
                }
        });
}
```

Screenshot after #14 and Right arrow pressed.

STEP 15

In GameManager.initializeGameGrid, clear the list and add all valid locations, call it before startGame.

SOLUTION CODE

```
GameManager(){
        if(Game2048.STEP>=15){
                initializeGameGrid();
        }
}
initializeGameGrid(){
        gameGrid.clear();
        locations.clear();
        for(int i=0; i<4; i++){
                for(int j=0; j<4; j++){
                Location location = new Location(i,j);
                locations.add(location);
                gameGrid.put(location, null);
                }
        }
}
```

STEP 16

Modify GameManager.startGame to get a random location for the tile shuffling a copy of locations, then pick two random tiles and add them to the board at random locations.

SOLUTION CODE

```
startGame (){
        List<Location> locCopy=locations.stream().collect(Collectors.toList());
        Collections.shuffle(locCopy);
        tile0.setLocation(locCopy.get(0));
        gameGrid.put(tile0.getLocation(), tile0);
        Tile tile1 = Tile.newRandomTile();
        tile1.setLocation(locCopy.get(1));
        gameGrid.put(tile1.getLocation(), tile1);

        redrawTilesInGameGrid();
}
redrawTilesInGameGrid(){
        gameGrid.values().stream().filter(Objects::nonNull).forEach(board::addTile);
}
```

STEP 17

In GameManager.findFarthestLocation, search for the farthest location in the direction of the movement without any tiles and inside the grid.

SOLUTION CODE

```
findFarthestLocation(){
        do {
                farthest = location;
                location = farthest.offset(direction);
        } while (location.isValidFor() && gameGrid.get(location)==null);
}
```

STEP 18

In GameManager.move, replace board.gridGroup with gameGrid, using a double IntStream to traverse the grid, moving the tiles to the farthest location possible (location & layout).

SOLUTION CODE

```
move(){
    IntStream.range(0, 4).boxed().forEach(i->{
        IntStream.range(0, 4).boxed().forEach(j->{
        Tile t=gameGrid.get(new Location(i,j));
        if(t!=null){
                final Location newLoc=findFarthestLocation(t.getLocation(),direction);
                if(!newLoc.equals(t.getLocation())){
                        board.moveTile(t, newLoc);
                        gameGrid.put(newLoc, t);
```

```
                        gameGrid.replace(t.getLocation(),null);
                        t.setLocation(newLoc);
                        }
                }
        });
    });
}
```

STEP 19

In GameManager.animateExistingTile, animate the tile translation from its actual location to the new one, in 65 ms.

SOLUTION CODE

```
animateExistingTile(){
        KeyValue kvX = new KeyValue(tile.layoutXProperty(),
        newLocation.getLayoutX(Board.CELL_SIZE) - (tile.getMinHeight() / 2),
Interpolator.EASE_OUT);
        KeyValue kvY = new KeyValue(tile.layoutYProperty(),
        newLocation.getLayoutY(Board.CELL_SIZE) - (tile.getMinHeight() / 2),
Interpolator.EASE_OUT);

        KeyFrame kfX = new KeyFrame(Duration.millis(65), kvX);
        KeyFrame kfY = new KeyFrame(Duration.millis(65), kvY);

        timeline.getKeyFrames().add(kfX);
        timeline.getKeyFrames().add(kfY);
}
```

STEP 20

In GameManager.move, use a parallelTransition to play the tiles animation. Use a volatile variable to avoid input while moving.

SOLUTION CODE

```
move(){
        synchronized (gameGrid) {
                if (movingTiles) {
                        return;
                }
        }
        …
        parallelTransition.getChildren().add(animateExistingTile(t, newLoc));
        …
        parallelTransition.setOnFinished(e -> {
```

```
                synchronized (gameGrid) {
                        movingTiles = false;
                }
        });
        synchronized (gameGrid) {
                movingTiles = true;
        }
        parallelTransition.play();
        parallelTransition.getChildren().clear();
}
```

STEP 21

In GameManager.findRandomAvailableLocation, from the remaining locations with no tile, shuffle the collection to get a random position, if any.

SOLUTION CODE

```
findFarthestLocation(){
        List<Location> availableLocations = locations.stream().filter(l ->
gameGrid.get(l) == null)
        .collect(Collectors.toList());

        if (availableLocations.isEmpty()) {
                return null;
        }

        Collections.shuffle(availableLocations);
        location = availableLocations.get(0);
}
```

STEP 22

In GameManager. addAndAnimateRandomTile, Set the scale to 0 of the new tile, and create scaleTransition to scale it to 1, in 125 ms, easy_out.

SOLUTION CODE

```
addAndAnimateRandomTile (){
        Tile tile = Tile.newRandomTile();
        tile.setLocation(randomLocation);
        tile.setScaleX(0);
        tile.setScaleY(0);
        board.addTile(tile);
        gameGrid.put(tile.getLocation(), tile);
```

```
        final ScaleTransition scaleTransition = new ScaleTransition(Duration.millis(125),
tile);
        scaleTransition.setToX(1.0);
        scaleTransition.setToY(1.0);
        scaleTransition.setInterpolator(Interpolator.EASE_OUT);
        scaleTransition.play();
}
```

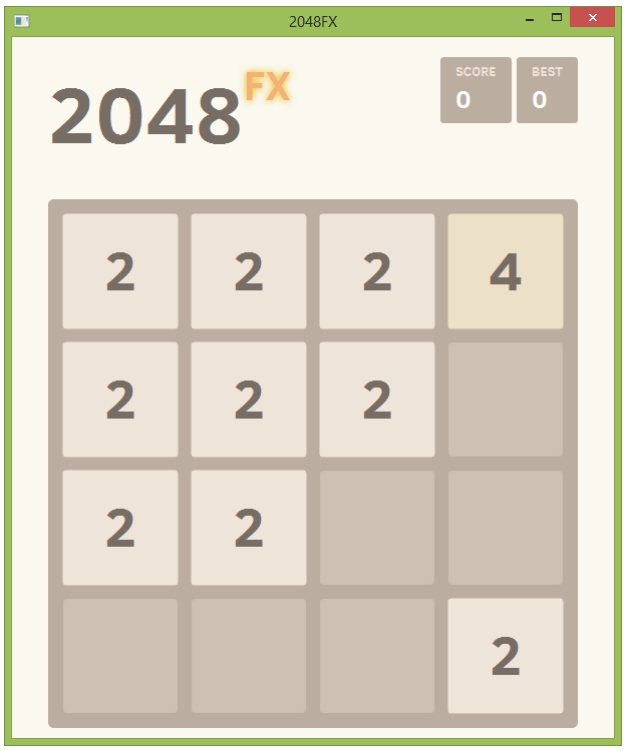
STEP 23

In GameManager.move, in parallel.OnFinished, get a randomLocation, check not null, create random tile, add to board and to map. Else print Game Over

SOLUTION CODE

```
move(){
parallelTransition.setOnFinished(){
        Location randomAvailableLocation = findRandomAvailableLocation();
                if (randomAvailableLocation != null){
                        addAndAnimateRandomTile(randomAvailableLocation);
                } else {
                        System.out.println("Game Over");
                }
}
}
```

Screenshot after #23



STEP 24

In GridOperator, traverseGrid method, apply the application of a functional to every cell of the grid, returning an int with the sum of the results of this functional.

SOLUTION CODE

```
traverseGrid(){
	traversalX.forEach(x -> {
		traversalY.forEach(y -> {
			at.addAndGet(func.applyAsInt(x, y));
		});
	});
}
```

STEP 25

In GameManager. initializeGameGrid method, replace the double for with the traverseGrid method.

In GameManager.move method, replace the IntStreams with the traverseGrid method. Assign to `tilesWereMoved` to account for the tiles moved.

In GameManager.move.setOnFinished, before adding a tile, check there were some movements done.

In Board.createGrid, replace the double for with the traverseGrid

SOLUTION CODE

```
initializeGameGrid(){
	GridOperator.traverseGrid((i, j) -> {
		Location location = new Location(i,j);
		locations.add(location);
		gameGrid.put(location, null);
		return 0;
	});
}
move(){
	tilesWereMoved = GridOperator.traverseGrid((i,j)->{
		Tile t=gameGrid.get(new Location(i,j));
		if(t!=null){
			final Location
		newLoc=findFarthestLocation(t.getLocation(),direction);
			if(!newLoc.equals(t.getLocation())){
				parallelTransition.getChildren().add(animateExistingTile(t,
			newLoc));
				gameGrid.put(newLoc, t);
				gameGrid.replace(t.getLocation(),null);
```

```
                        t.setLocation(newLoc);
                        return 1;
                    }
                }
                return 0;
        });
}

move(){
        setOnFinished(){
                if(tilesWereMoved>0){
                        addAndAnimateRandomTile(randomAvailableLocation);
                }
        }
}
createGrid(){
        GridOperator.traverseGrid((i, j) -> {
                gridGroup.getChildren().add(createCell(i, j));
                return 0;
        });
}
```

STEP 26

In GridOperator, sort the traverse X,Y list, so for Right or Down directions traverseX,Y are taken in reverse order. In move, first of all call sortGrid before traverseGrid.

SOLUTION CODE

```
sortGrid(){
        Collections.sort(traversalX, direction.equals(Direction.RIGHT) ?
Collections.reverseOrder() : Integer::compareTo);
        Collections.sort(traversalY, direction.equals(Direction.DOWN)?
Collections.reverseOrder() : Integer::compareTo);
}
move(){
        GridOperator.sortGrid(direction);
}
```

STEP 27

In Tile. merge method, add to tile's value the value of the tile to be merged to, set the text of the label with the new value and replace the old style 'game-title-"-value with the new one. In Tile.isMergeable Check it this.tile can be merged with anotherTile

SOLUTION CODE

```
merge(){
```

```
        getStyleClass().remove("game-tile-" + value);
        this.value += another.getValue();
        setText(Integer.toString(value));
        merged = true;
        getStyleClass().add("game-tile-" + value);
}
isMergeable(){
        return anotherTile != null && getValue()==anotherTile.getValue();
}
```

STEP 28

In GameManager. animateMergedTile method, add a sequential animation, with two scale animations, from 1 to 1.2, ease_in, and from 1.2 to 1 ease_out, in 80 ms each

SOLUTION CODE

```
animateMergedTile(){
        final ScaleTransition scale0 = new ScaleTransition(Duration.millis(80), tile);
        scale0.setToX(1.2);
        scale0.setToY(1.2);
        scale0.setInterpolator(Interpolator.EASE_IN);

        final ScaleTransition scale1 = new ScaleTransition(Duration.millis(80), tile);
        scale1.setToX(1.0);
        scale1.setToY(1.0);
        scale1.setInterpolator(Interpolator.EASE_OUT);

        return new SequentialTransition(scale0, scale1);
}
```

STEP 29

In GameManager.move method, get tile for an offset, check if it's a valid tile, not merged, and check if tiles can be merged. Then merge this new tile with the old one, move to front the new tile, put the nextLocation on the map, with the new tile and replace last location with null. Add old tile to animateExistingTile, and new one to animateMerdedTile. Add old tile to mergedToBeRemoved, and return 1.

SOLUTION CODE

```
move(){
        Location nextLocation = newLoc.offset(direction);
        Tile tileToBeMerged = nextLocation.isValidFor() ? gameGrid.get(nextLocation) :
null;
        if (tileToBeMerged != null && !tileToBeMerged.isMerged() &&
t.isMergeable(tileToBeMerged)) {
                tileToBeMerged.merge(t);
```

```
                tileToBeMerged.toFront();

                gameGrid.put(nextLocation, tileToBeMerged);

                gameGrid.replace(t.getLocation(), null);

                parallelTransition.getChildren().add(animateExistingTile(t,
        nextLocation));

                parallelTransition.getChildren().add(animateMergedTile(tileToBeMerged));

                mergedToBeRemoved.add(t);

                return 1;

        }

}
```

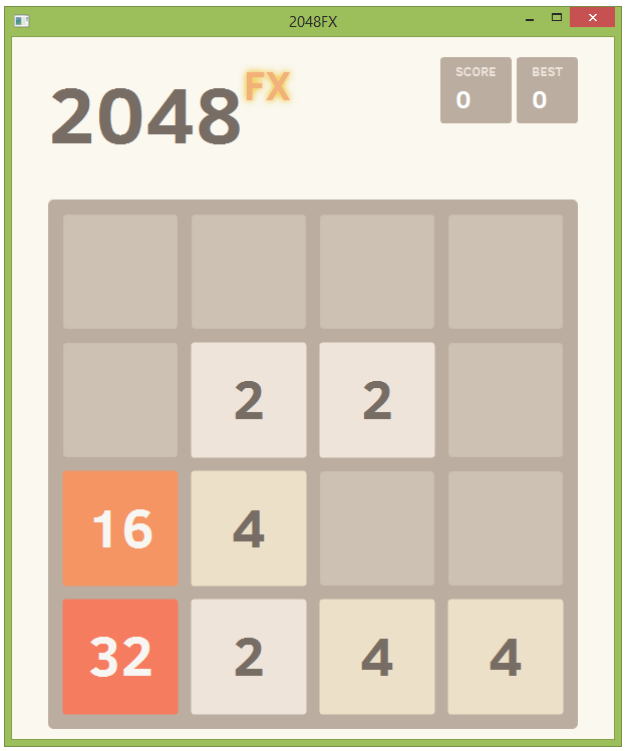
STEP 30

In GameManager.move, on finished method, remove the tiles in the set from the gridGroup and clear the set. For all the tiles on the board: set to false their merged value.

SOLUTION CODE

```
move(){

setOnFinished(){

        board.getGridGroup().getChildren().removeAll(mergedToBeRemoved);

        mergedToBeRemoved.clear();

        gameGrid.values().stream().filter(Objects::nonNull).forEach(t-
>t.setMerged(false));

        }

}
```

Screenshot after #30



STEP 31

In Board.createGrid, to avoid dropshadow effect of higher tiles move the grid, add a rectangle to clip hBottom.

SOLUTION CODE

```
createGrid(){
        Rectangle rect = new Rectangle(GRID_WIDTH, GRID_WIDTH);
        hBottom.setClip(rect);
}
```

STEP 32

In Board.createScore, add style for lblPoints, add to board. Add a listener to text changes, so it gets centered right below the center of vScore.

SOLUTION CODE

```
createScore(){
        lblPoints.getStyleClass().addAll("game-label","game-points");
        lblPoints.setAlignment(Pos.CENTER);
        lblPoints.setMinWidth(100);
        getChildren().add(lblPoints);
}
```

STEP 33

In Board.createScore, bind lblPoints to gameMovePoints with a “+” prefix, if points>0, and bind the lblScore text property with the gameScore property.

In Board.addPoints, add the points to gameMove and gameScore property.

In GameManager.move, reset the points before traverseGrid, and add the points for every merged cell

SOLUTION CODE

```
createScore(){
        lblPoints.textProperty().bind(Bindings.createStringBinding(()->
                (gameMovePoints.get()>0)?"+".concat(Integer.toString(gameMovePoints.get()
        )):"", gameMovePoints.asObject()));
        lblScore.textProperty().bind(gameScoreProperty.asString());
}
addPoints(){
        gameMovePoints.set(gameMovePoints.get() + points);
        gameScoreProperty.set(gameScoreProperty.get() + points);
}
move(){
        board.setPoints(0);
}
```

```
move(){
        board.addPoints(tileToBeMerged.getValue());
}
```

STEP 34

In Board.createScore, add a listener to lblPoints text changes, so it gets centered right below the center of vScore.

SOLUTION CODE

```
createScore(){
        lblPoints.textProperty().addListener((ov,s,s1)->{
                lblPoints.setLayoutX(0);
                double midScoreX=vScore.localToScene(vScore.getWidth()/2d,0).getX();
                lblPoints.setLayoutX(lblPoints.sceneToLocal(midScoreX, 0).getX()-
        lblPoints.getWidth()/2d);
        });
}
```

STEP 35

In Board.createScore, create the timeline to translate the lblPoints in Y from 20 to 100 and reduce its opacity from 1 to 0 in 600 ms.

In Board.animateScore, start the animation.

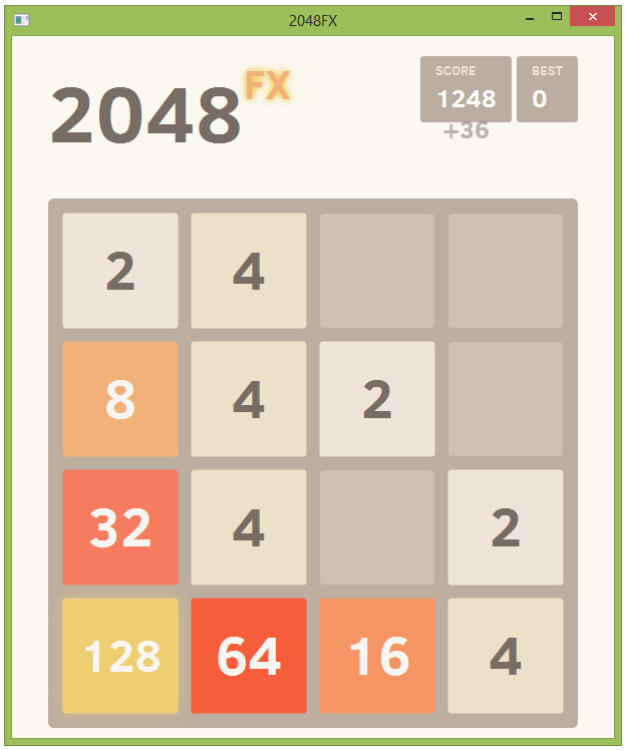
In GameManager.move call board.animateScore after traverseGrid.

SOLUTION CODE

```
createScore(){
        final KeyValue kvO0 = new KeyValue(lblPoints.opacityProperty(), 1);
        final KeyValue kvY0 = new KeyValue(lblPoints.layoutYProperty(), 20);
        final KeyValue kvO1 = new KeyValue(lblPoints.opacityProperty(), 0);
        final KeyValue kvY1 = new KeyValue(lblPoints.layoutYProperty(), 100);
        final KeyFrame kfO0 = new KeyFrame(Duration.ZERO, kvO0);
        final KeyFrame kfY0 = new KeyFrame(Duration.ZERO, kvY0);

        Duration animationDuration = Duration.millis(600);
        final KeyFrame kfO1 = new KeyFrame(animationDuration, kvO1);
        final KeyFrame kfY1 = new KeyFrame(animationDuration, kvY1);

        animateAddedPoints.getKeyFrames().addAll(kfO0,kfY0,kfO1,kfY1);
}
animateScore(){
        animateAddedPoints.playFromStart();
}
```

```
move(){
        board.animateScore();
}
```

Screenshot after #35



STEP 36

In GameManager.mergeMovementsAvailable, traverse the grid in two directions (Up, Left) and for every tile look if the offset tile is mergeable.

SOLUTION CODE

```
mergeMovementsAvailable(){
        Stream.of(Direction.UP, Direction.LEFT).parallel().forEach(direction -> {
                GridOperator.traverseGrid((x, y) -> {
                        Location thisloc = new Location(x, y);
                        Tile t1=gameGrid.get(thisloc);
                        if(t1!=null){
                                Location nextLoc=thisloc.offset(direction);
                                if(nextLoc.isValidFor()){
                                        Tile t2=gameGrid.get(nextLoc);
                                        if(t2!=null && t1.isMergeable(t2)){
                                                numMergeableTile.incrementAndGet();
                                        }
                                }
                        }
                        return 0;
```

```
                });
        });
}
```

STEP 37

In GameManager.move.setOnFinished, call mergeMovementsAvailable if randomAvailableLocation == null and print Game Over, else try to add a new tile if tiles were moved

In GameManager.addAndAnimateRandomTile, after last movement on full grid, check if there are movements available

SOLUTION CODE

```
setOnFinished(){
        if(mergeMovementsAvailable()==0){
                System.out.println("Game Over");
        };
}
addAndAnimateRandomTile(){
        scaleTransition.setOnFinished(e -> {
                if (gameGrid.values().parallelStream().noneMatch(Objects::isNull) &&
        mergeMovementsAvailable()==0 ) {
                        System.out.println("Game Over");
                }
        });
}
```

STEP 38

In GameManager.move, check if the merged tile is 2048, and print win.

SOLUTION CODE

```
move(){
        if(tileToBeMerged.getValue()==2048){
                System.out.println("You win!");
        }
}
```

STEP 39

In Board.initGameProperties, style buttons, set listeners to click. In both, remove overlay. In bTry also remove tiles and reset all game properties.

SOLUTION CODE

```
initGameProperties(){
        bTry.getStyleClass().add("game-button");
        bTry.setOnAction(e->{
```

```
                getChildren().removeAll(overlay, buttonsOverlay);
                gridGroup.getChildren().removeIf(c->c instanceof Tile);
                resetGame.set(false);
                gameScoreProperty.set(0);
                gameWonProperty.set(false);
                gameOverProperty.set(false);
                resetGame.set(true);
        });
        bContinue.getStyleClass().add("game-button");
        bContinue.setOnAction(e->getChildren().removeAll(overlay, buttonsOverlay));
}
```

STEP 40

In Board.initGameProperties, add listeners to game over, won properties. Set style to overlay, set text and its style, add buttons, and add overlay to board.

SOLUTION CODE

```
initGameProperties(){
        gameOverProperty.addListener((observable, oldValue, newValue) -> {
                if (newValue) {
                        overlay.getStyleClass().setAll("game-overlay","game-overlay-
        over");
                        lOvrText.setText("Game over!");
                        lOvrText.getStyleClass().setAll("game-label","game-lblOver");
                        buttonsOverlay.getChildren().setAll(bTry);
                        this.getChildren().addAll(overlay,buttonsOverlay);
                }
        });

        gameWonProperty.addListener((observable, oldValue, newValue) -> {
                if (newValue) {
                        overlay.getStyleClass().setAll("game-overlay","game-overlay-won");
                        lOvrText.setText("You win!");
                        lOvrText.getStyleClass().setAll("game-label","game-lblWon");
                        buttonsOverlay.getChildren().setAll(bContinue, bTry);
                        this.getChildren().addAll(overlay,buttonsOverlay);
                }
        });
}
```

STEP 41

In Board constructor call initGameProperties.

In GameManager.move, add set win with setGameWin, and setGameOver in onFinished

In GameManager.addAndAnimateRandomTile, add setGameOver.

SOLUTION CODE

```
Board(){
	initGameProperties();
}
move(){
	board.setGameWin(true);
}


move(){
	setOnFinished(){
		board.setGameOver(true);
	}
}
addAndAnimateRandomTile(){
	board.setGameOver(true);
}
```

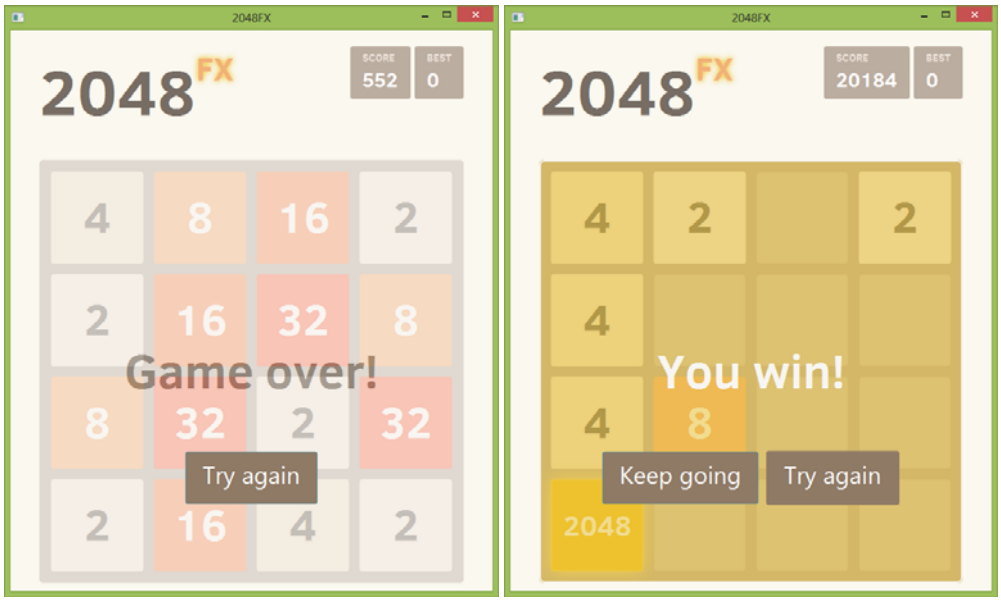
STEP 42

In GameManager constructor add a listener to reset game property to start the game again with a clear grid.

SOLUTION CODE

```
GameManager(){
	board.resetGameProperty().addListener((ov, b, b1) -> {
		if (b1) {
			initializeGameGrid();
			startGame();
		}
	});
}
```

Screenshots after #42



STEP 43

In GameManager.optionalTile, return an Optional of nullable from a given location on the map gameGrid.

SOLUTION CODE

```
optionalTile(){
        return Optional.ofNullable(gameGrid.get(loc));
}
```

STEP 44

In GameManager.mergeMovementsAvailable, use optionalTile to find pairs of mergeable tiles

SOLUTION CODE

```
mergeMovementsAvailable (){
        optionalTile(thisloc).ifPresent(t1->{
                optionalTile(thisloc.offset(direction)).filter(t2->t1.isMergeable(t2))
                        .ifPresent(t2->numMergeableTile.incrementAndGet());
                });
}
```

STEP 45

In GameManager.move, use optionalTile to traverse the grid, with an atomicInteger to return the results

SOLUTION CODE

```
move (){
        GridOperator.sortGrid(direction);
        board.setPoints(0);
        tilesWereMoved = GridOperator.traverseGrid((i, j) -> {
                AtomicInteger result=new AtomicInteger();
```

```
                optionalTile(new Location(i,j)).ifPresent(t1->{

                        final Location newLoc=findFarthestLocation(t1.getLocation(),
                direction);

                        Location nextLocation = newLoc.offset(direction); // calculates to
                a possible merge

                        optionalTile(nextLocation).filter(t2->t1.isMergeable(t2) &&
                !t2.isMerged()).ifPresent(t2->{

                                t2.merge(t1);

                                t2.toFront();

                                gameGrid.put(nextLocation, t2);

                                gameGrid.replace(t1.getLocation(), null);

                                board.addPoints(t2.getValue());

                                if(t2.getValue()==2048){

                                        board.setGameWin(true);

                                }

                                parallelTransition.getChildren().add(animateExistingTile(t1
                        , nextLocation));

                                parallelTransition.getChildren().add(animateMergedTile(t2))
                        ;

                                mergedToBeRemoved.add(t1);



                                result.set(1);

                        });



                        if(result.get()==0 && !newLoc.equals(t1.getLocation())){

                                parallelTransition.getChildren().add(animateExistingTile(t1
                        , newLoc));

                                gameGrid.put(newLoc, t1);

                                gameGrid.replace(t1.getLocation(),null);

                                t1.setLocation(newLoc);

                                result.set(1);

                        }

                });

                return result.get();

        });

}
```