Exercise 1

1.

CRC Cards before the change in code:

TwentyFourtyGame	
Superclass: Game	
Subclasses: none	
Setting screen	GameScreen
Initialize handler of the assets	AssetHandler
Initialize handler of the buttons	ButtonHandler
Initialize scores from preferences	PreferenceHandler
Disposing of the screen	GameScreen
Disposing of the handler of assets	AssetHandler
Disposing of the game	Game

GameScreen	
Superclass: Screen	
Subclasses: none	
Creating world	GameWorld
Setting up the input processor	InputHandler
Setting up the renderer	GameRenderer
Manipulating the screen	Screen
Pause the game	Button, ProgressHandler

GameWorld	
Superclass: none	
Subclasses: none	
Load saved game	ProgressHandler
Creating new game	AnimatedGrid
Handling the state of the gam	GameState
Handling the highscore	
Handling the highest tile	
Handling the current score	

TileHandler	
Superclass: none	
Subclasses: none	
Determine if a neighbour is free	
Move all the tiles in one direction	Tile
Merge the tiles which have the same value	Tile

Grid	
Superclass: none	
Subclasses: AnimatedGrid	
Initialize the grid with two random tiles	Tile
Setting the tiles in the grid	Tile
Handling the moving of each tile	TileHandler
Checking if there is a move left	

InputHandler	
Superclass: InputProcessor	
Subclasses: none	
Detects key pressed	
Gives the direction of the key	Grid

CRC Cards after the change in code:

TwentyFourtyGame	
Superclass: Game	
Subclasses: none	
Initialize handler of the screens	ScreenHandler
Initialize handler of the assets	AssetHandler
Initialize scores from preferences	PreferenceHandler
Disposing of the screen	GameScreen
Disposing of the handler of assets	AssetHandler
Disposing of the game	Game
Handling the state of the game	GameState
Handling the highscore	
Handling the highest tile	
Handling the current score	

GameScreen	
Superclass: Screen	
Subclasses: none	
Creating the GUI	
Setting up the input processor	InputHandler
Adding actors to the GUI	RestartButton
Manipulating the screen	Screen
Pause the game	MenuButton, ProgressHandler
Setting up the grid	ProgressHandler
Checks if the game is won or lost	Grid

ProgressHandler	
Superclass: none	
Subclass: none	
Loading game from preferences	
Saving game to preferences	

TileHandler	
Superclass: none	
Subclasses: none	
Determine if a neighbour is free	
Move all the tiles in one direction	Tile
Merge the tiles which have the same value	Tile

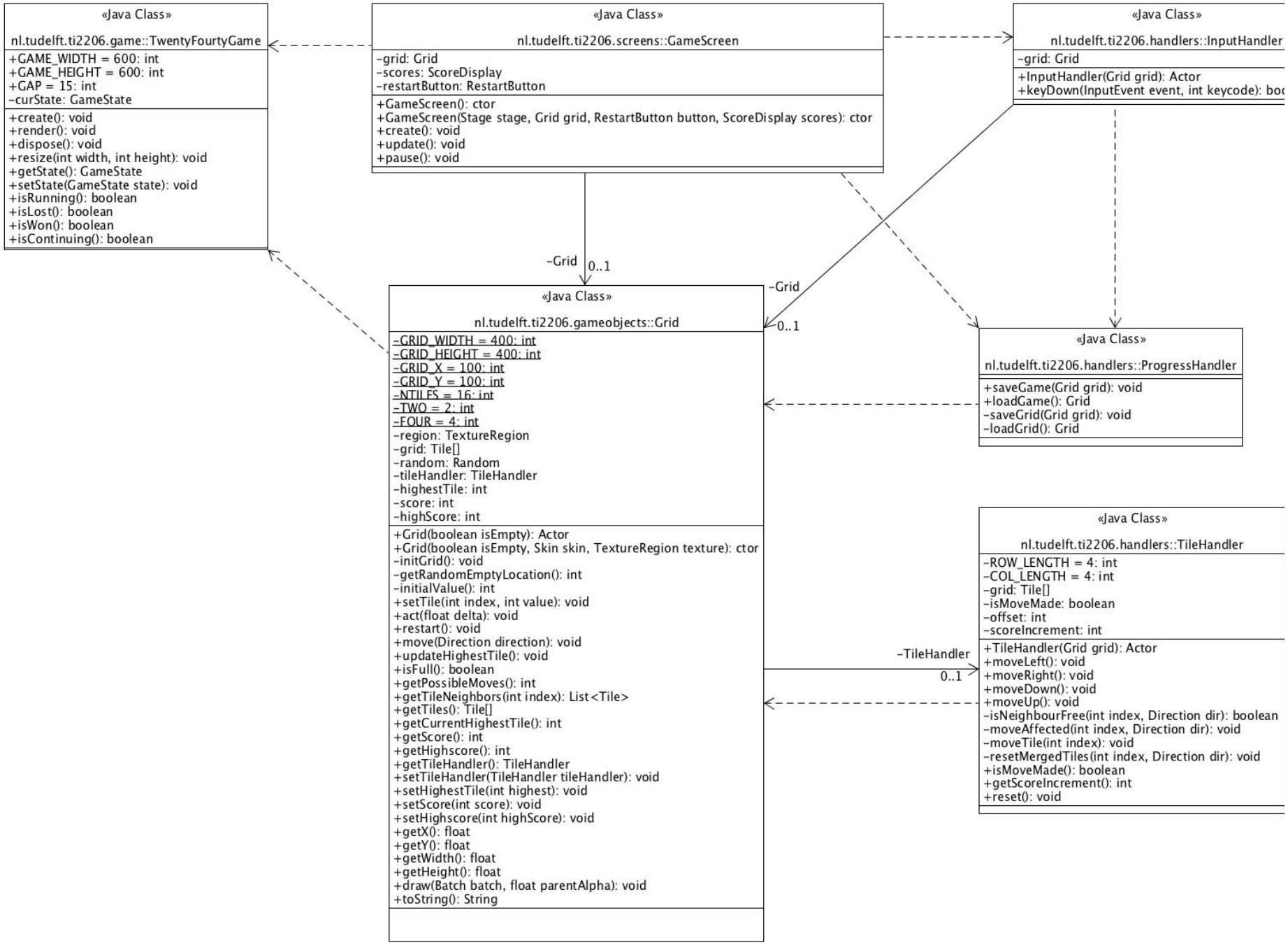
Grid	
Superclass: Actor	
Subclasses: none	
Initialize the grid with two random tiles	Tile
Setting the tiles in the grid	Tile
Handling the direction in which each tile is move	TileHandler
Checking if there is a move left	

InputHandler	
Superclass: InputProcessor	
Subclasses: none	
Detects key pressed	
Gives the direction of the key	Grid

2.

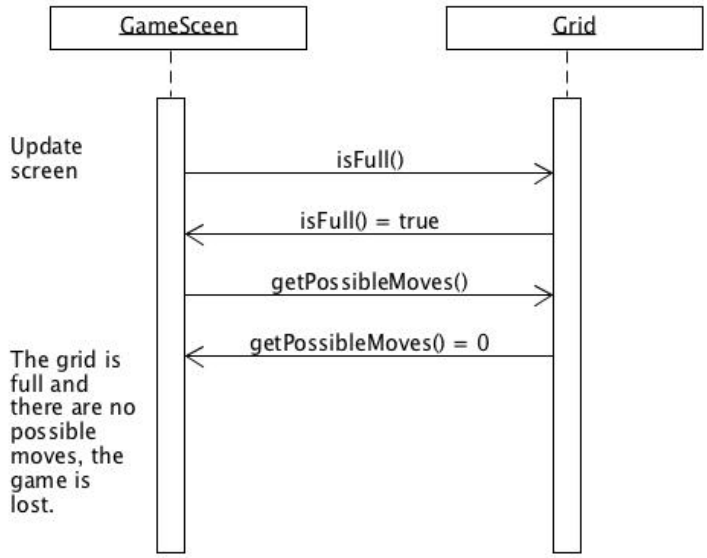
The other classes do have one responsibility and are not necessary for the basics of the game. The classes we will keep are Tile, AssetHandler, ProgressHandler, PreferenceHandler, CoordinateHandler and all the Button classes as they have all a unique responsibility.

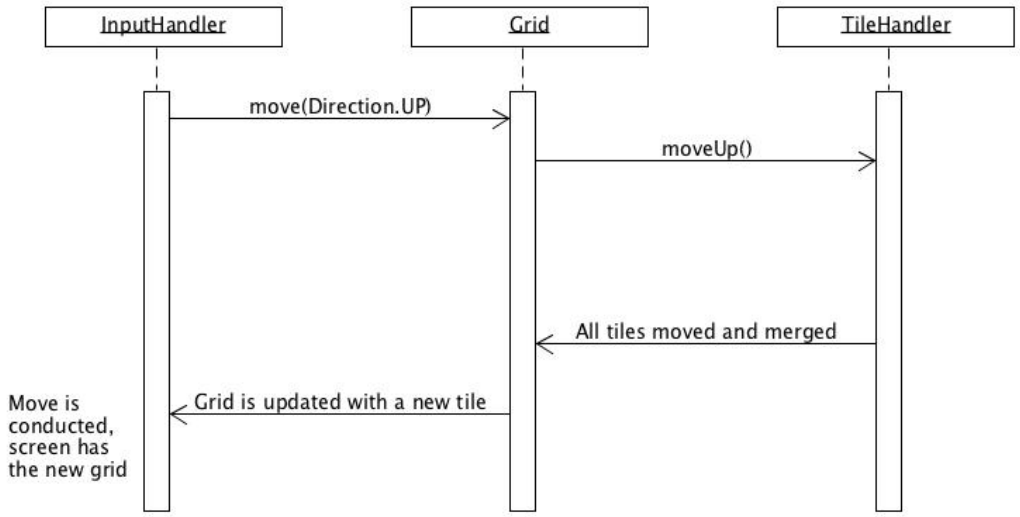
We are going to delete the GameRenderer as we have found a better way to render using stages and actors from the library GDX.
After reflection it was clear that we could merge TwentyFourtyGame with GameWorld, as they would have similar responsibility after we started using stages and actors. The class ProgressHandler will get the responsibility from GameWorld to make initialize a Grid, as it already made a Grid when a game was saved. The class ButtonHandler became also obsolete after using stages and actors, as the buttons could become an actor.
We are going to merge the AnimatedGrid with Grid and AnimatedTile with Tile; these were separate so it conformed to the model view controller pattern. These are now also actors.

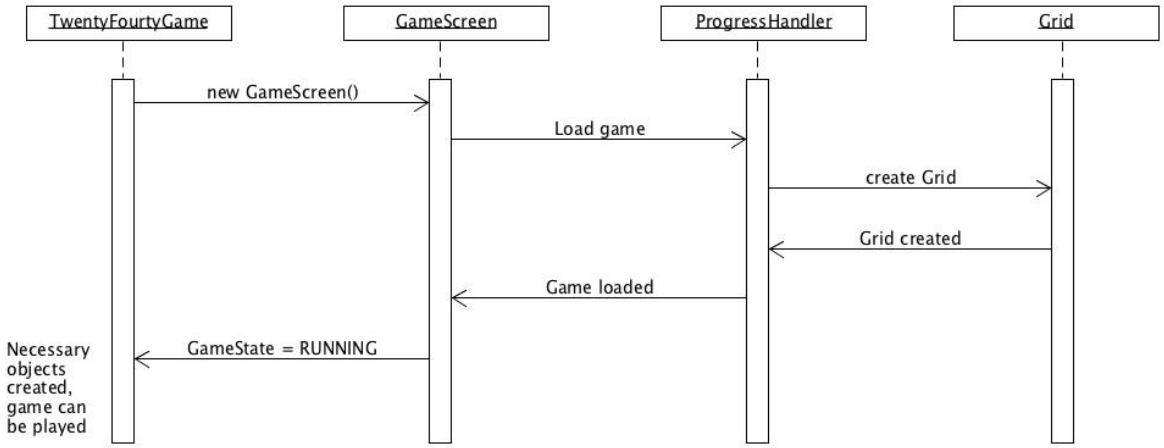

«Java Class»
nl.tudelft.ti2206.game::TwentyFourtyGame
+GAME_WIDTH = 600: int
+GAME_HEIGHT = 600: int
+GAP = 15: int
-curState: GameState
+create(): void
+render(): void
+dispose(): void
+resize(int width, int height): void
+getState(): GameState
+setState(GameState state): void
+isRunning(): boolean
+isLost(): boolean
+isWon(): boolean
+isContinuing(): boolean
«Java Class»
nl.tudelft.ti2206.screens::GameScreen
-grid: Grid
-scores: ScoreDisplay
-restartButton: RestartButton
+GameScreen(): ctor
+GameScreen(Stage stage, Grid grid, RestartButton button, ScoreDisplay scores): ctor
+create(): void
+update(): void
+pause(): void
«Java Class»
nl.tudelft.ti2206.handlers::InputHandler
-grid: Grid
+InputHandler(Grid grid): Actor
+keyDown(InputEvent event, int keycode): boo
-Grid
0..1
-Grid
0..1
«Java Class»
nl.tudelft.ti2206.gameobjects::Grid
-GRID_WIDTH = 400: int
-GRID_HEIGHT = 400: int
-GRID_X = 100: int
-GRID_Y = 100: int
-NTILES = 16: int
-TWO = 2: int
-FOUR = 4: int
-region: TextureRegion
-grid: Tile[]
-random: Random
-tileHandler: TileHandler
-highestTile: int
-score: int
-highScore: int
+Grid(boolean isEmpty): Actor
+Grid(boolean isEmpty, Skin skin, TextureRegion texture): ctor
-initGrid(): void
-getRandomEmptyLocation(): int
-initialValue(): int
+setTile(int index, int value): void
+act(float delta): void
+restart(): void
+move(Direction direction): void
+updateHighestTile(): void
+isFull(): boolean
+getPossibleMoves(): int
+getTileNeighbors(int index): List<Tile>
+getTiles(): Tile[]
+getCurrentHighestTile(): int
+getScore(): int
+getHighscore(): int
+getTileHandler(): TileHandler
+setTileHandler(TileHandler tileHandler): void
+setHighestTile(int highest): void
+setScore(int score): void
+setHighscore(int highScore): void
+getX(): float
+getY(): float
+getWidth(): float
+getHeight(): float
+draw(Batch batch, float parentAlpha): void
+toString(): String
«Java Class»
nl.tudelft.ti2206.handlers::ProgressHandler
+saveGame(Grid grid): void
+loadGame(): Grid
-saveGrid(Grid grid): void
-loadGrid(): Grid
-TileHandler
0..1
«Java Class»
nl.tudelft.ti2206.handlers::TileHandler
-ROW_LENGTH = 4: int
-COL_LENGTH = 4: int
-grid: Tile[]
-isMoveMade: boolean
-offset: int
-scoreIncrement: int
+TileHandler(Grid grid): Actor
+moveLeft(): void
+moveRight(): void
+moveDown(): void
+moveUp(): void
-isNeighbourFree(int index, Direction dir): boolean
-moveAffected(int index, Direction dir): void
-moveTile(int index): void
-resetMergedTiles(int index, Direction dir): void
+isMoveMade(): boolean
+getScoreIncrement(): int
+reset(): void

3.

4.


GameSceen
Grid
Update screen
isFull()
isFull() = true
getPossibleMoves()
getPossibleMoves() = 0
The grid is full and there are no possible moves, the game is lost.


InputHandler
Grid
TileHandler
move(Direction.UP)
moveUp()
All tiles moved and merged
Grid is updated with a new tile
Move is conducted, screen has the new grid


TwentyFourtyGame
GameScreen
ProgressHandler
Grid
new GameScreen()
Load game
create Grid
Grid created
Game loaded
GameState = RUNNING
Necessary objects created, game can be played

UML of the final code: