The Project

When we first started working on our assignment for the class we were told we would be making a game that involved moving tile pieces around to complete a maze pattern. Throughout the process of making this project, we used an Agile Design process. This involved a cycle of receiving requirements, creating a new design for the software, developing that design, creating or updating the documentation for the design, testing the software, and submitting or deploying the software. The entire process was on a deadline just as a job would be.

Each version's UML diagram is displayed under the process description. The first UML shows the ideas we had for the entire project, and each subsequent UML shows what was implemented at the time.

Version 1

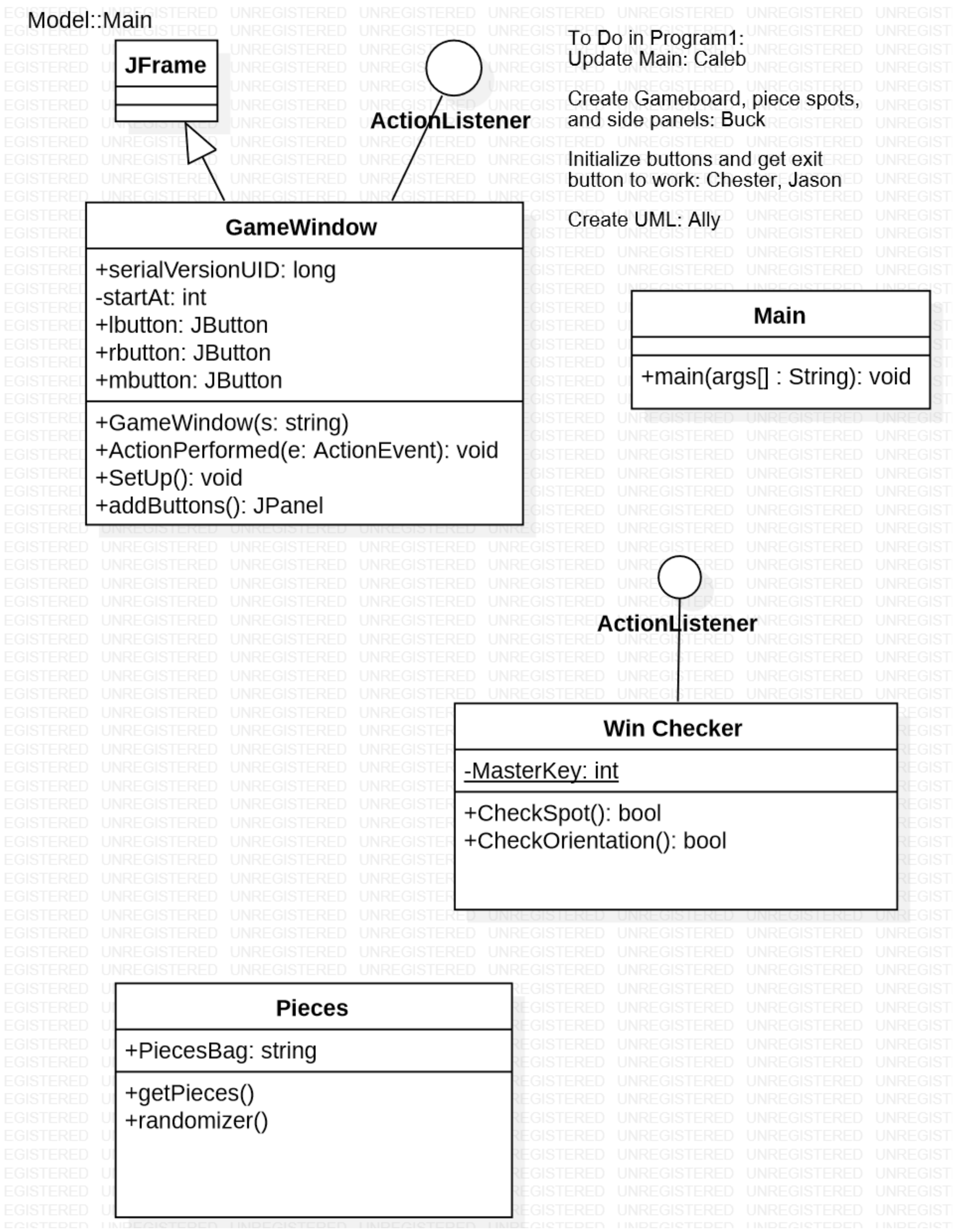
This portion of the project was where the most planning happened. Not only did we have to plan for the current implementation, but we also had to come up with ideas for every future implementation and consider how the new parts may interact with the old.

For this program, we had an initial meeting that was dedicated solely to the design step. This involved drawing diagrams on a board, creating lists of possible features, and refining our ideas into a final concept of what we were making. After this meeting adjourned, we began development on the code we needed for the features we wanted.

During this meeting, we also assigned everyone a role for what we thought everyone would do. These roles would be shifted around later, but that’s not unexpected since plans can change. After a few iterations, however, our roles were fairly well cemented.

Through a few exchanged messages we eventually finished the code and decided to meet once more to test the code and make sure that all the documents were up to date. Upon testing, we made a few changes and prepared everything for submission. After submission, we simply had to wait for the boss, our teacher, to test the software and give us feedback on what they did and didn’t like and what else they wanted to see the program do. This would be considered for future implementations.

The first few meetings were also the only meetings everyone could attend. Given academic and work schedules, people’s calendars can be pretty full. In an actual work environment this wouldn’t be as much of a problem as set meetings would be called every so often to make sure everything is on track and to address any issues that may arise.



To Do in Program1:
Update Main: Caleb

Create Gameboard, piece spots, and side panels: Buck

Initialize buttons and get exit button to work: Chester, Jason

Create UML: Ally

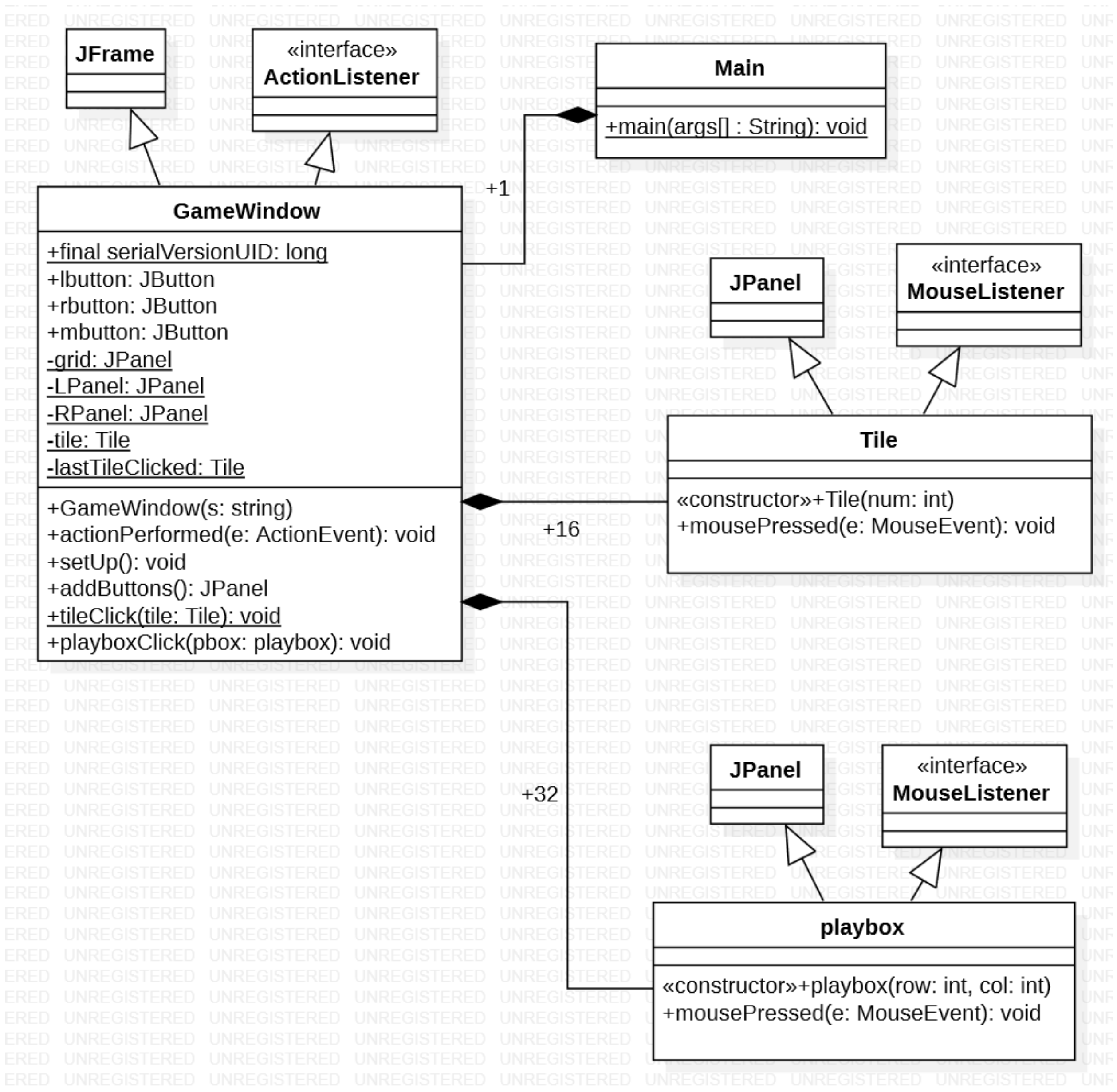
Version 2

For this program implementation there were a few subtle changes in the overall process because we now had to handle client feedback, but all the basic steps still remained.

At the start of the project we were given a feature that the client wanted. We now had to add tiles to the window we had created in the previous program. After having an initial meeting to go over the new requirements, we all agreed to try to make an implementation that met the requirements and reconvene to see which one we liked the best. At the second meeting, we decided that Buck's was the best and we would continue to build on his edition. We also received feedback from our boss and had to adjust what we had created during the first version accordingly.

After the meeting adjourned, we created all the necessary documentation and agreed to meet one more time to test the code and finalize the documents. At this point, we started to encounter issues with schedules, but everyone still finished their respective parts.

There was one more meeting for this version that included a rundown of the new requirements and some corrections for the old. Shortly after this meeting, the new version was once again given to the boss to review and comment on. Our team continued to think of what might be in future implementations as we awaited our feedback.

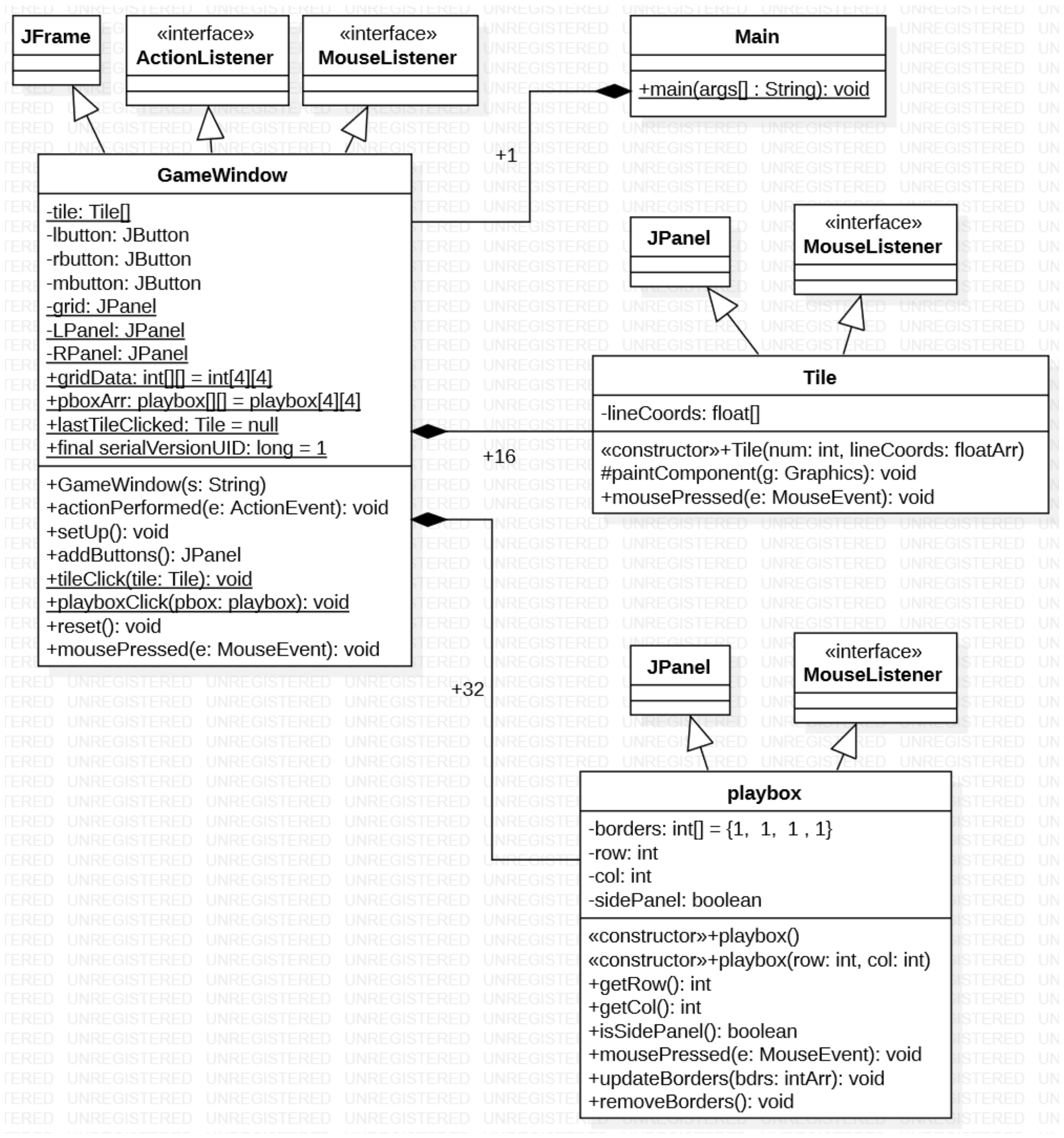

JFrame
«interface»
ActionListener
Main
+main(args[] : String): void
+1
GameWindow
+final serialVersionUID: long
+lbutton: JButton
+rbutton: JButton
+mbutton: JButton
-grid: JPanel
-LPanel: JPanel
-RPanel: JPanel
-tile: Tile
-lastTileClicked: Tile
+GameWindow(s: string)
+actionPerformed(e: ActionEvent): void
+setUp(): void
+addButtons(): JPanel
+tileClick(tile: Tile): void
+playboxClick(pbox: playbox): void
JPanel
«interface»
MouseListener
Tile
«constructor»+Tile(num: int)
+mousePressed(e: MouseEvent): void
+16
JPanel
«interface»
MouseListener
+32
playbox
«constructor»+playbox(row: int, col: int)
+mousePressed(e: MouseEvent): void

Version 3
By this version, the roles had been firmly cemented. Buck was the main code developer, Ally and Caleb were the UML leads, Jason was in charge of the minutes and helping Chester with the version document, and Chester was in charge of making sure everything was uploaded on time and submitted.

This version of the program focused on making moveable tiles. Our boss gave us two options to either have the tiles be drag and drop or click to select and move. We also received feedback earlier for this implementation, so we were able to design the code with the new requirements and feedback in mind.

Our initial meeting for this program involved redesigning the code and implementing some of the new features. We also decided to update all of the documentation that we could during this meeting and later finished the rest when the code was done.

The second meeting was focused on testing the code before deploying it. Any bugs that were found were fixed, and the new code documentation guidelines our boss wanted us to follow were implemented. After a brief review of everything, we again submitted the program and waited for it to be evaluated by the boss.



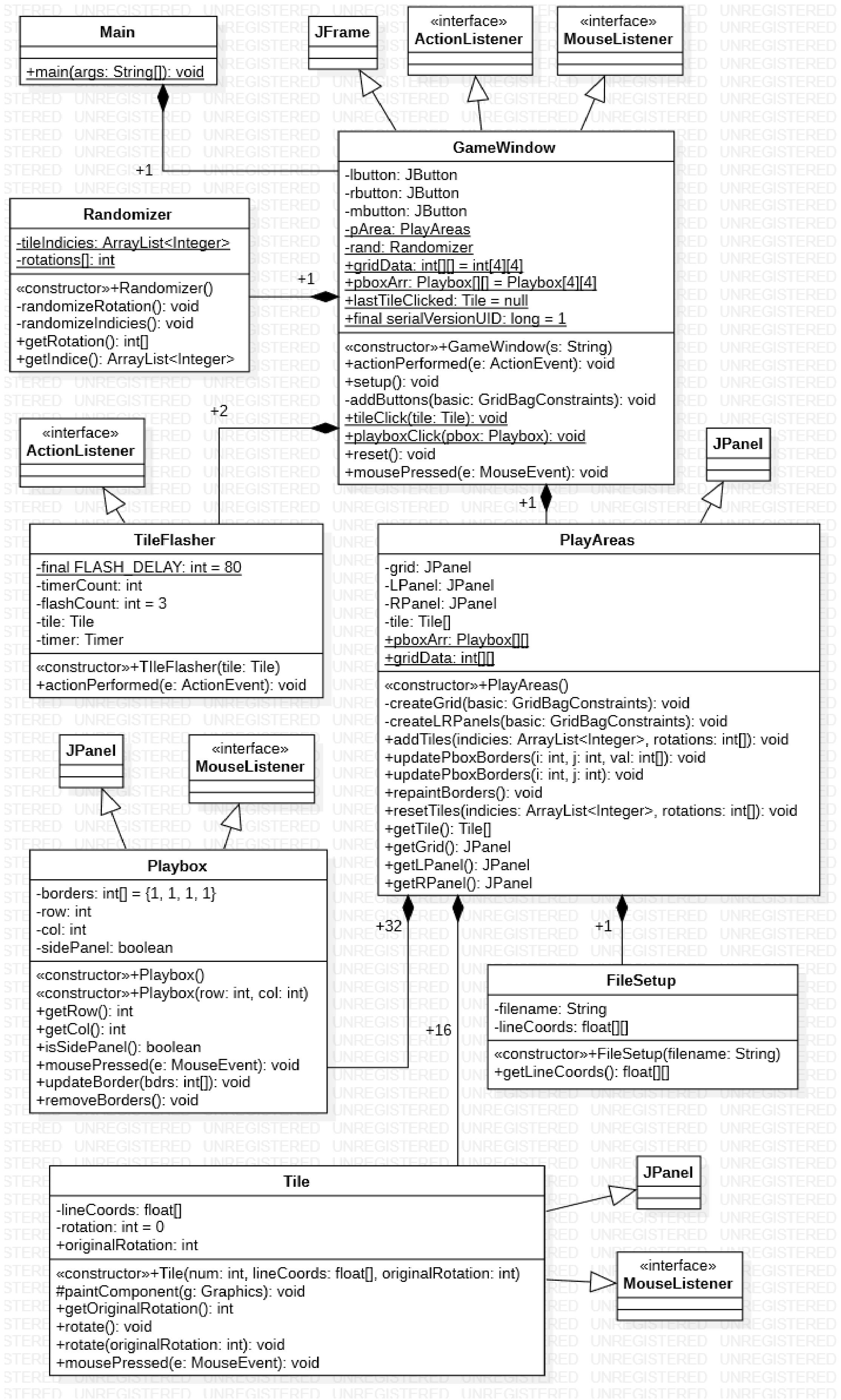
JFrame
«interface» ActionListener
«interface» MouseListener
Main
+main(args[] : String): void
+1
GameWindow
-tile: Tile[]
-lbutton: JButton
-rbutton: JButton
-mbutton: JButton
-grid: JPanel
-LPanel: JPanel
-RPanel: JPanel
+gridData: int[][] = int[4][4]
+pboxArr: playbox[][] = playbox[4][4]
+lastTileClicked: Tile = null
+final serialVersionUID: long = 1
+GameWindow(s: String)
+actionPerformed(e: ActionEvent): void
+setUp(): void
+addButtons(): JPanel
+tileClick(tile: Tile): void
+playboxClick(pbox: playbox): void
+reset(): void
+mousePressed(e: MouseEvent): void
JPanel
«interface» MouseListener
Tile
-lineCoords: float[]
«constructor»+Tile(num: int, lineCoords: floatArr)
#paintComponent(g: Graphics): void
+mousePressed(e: MouseEvent): void
+16
+32
JPanel
«interface» MouseListener
playbox
-borders: int[] = {1, 1, 1 , 1}
-row: int
-col: int
-sidePanel: boolean
«constructor»+playbox()
«constructor»+playbox(row: int, col: int)
+getRow(): int
+getCol(): int
+isSidePanel(): boolean
+mousePressed(e: MouseEvent): void
+updateBorders(bdrs: intArr): void
+removeBorders(): void

Version 4
Version 4 continued to add to the design portion of the process. Not only did we have to implement new changes and accommodate what the client wanted from the last iteration, we also had to reconsider our design going forward. In other words, we needed to refactor parts of the code. Given our limited time window, we only got a few items compartmentalized properly and would continue the process in Version 5.

The focus of this version was to randomize the starting tile placement. To do this, we added a class to handle that task. After generating the tiles, the randomizer would give each a random starting position and rotation. We spent most of the first meeting addressing this issue and creating documentation for it.

After receiving feedback from our boss, we realized our design needed some work. After some discussion through messages, we came up with a new plan. Going forward, we would make GameWindow a display for the pieces and board, have PlayAreas (board and starting positions) made of PlayBoxes, have the PlayBoxes hold the tiles, and have the Tiles contain all the tile data. To start making GameWindow and Tile align with this new design, we made a class to handle input file reading and removed that functionality from the Tile class. We also added a class to handle the animation that plays when an improper move is made.

All of these changes were designed close to the deadline and had to be implemented on a crunch. Because of this, the team had to collaborate heavily over a few days to ensure everything was implemented and tested.


Main
+main(args: String[]): void
JFrame
«interface»
ActionListener
«interface»
MouseListener
+1
GameWindow
-lbutton: JButton
-rbutton: JButton
-mbutton: JButton
-pArea: PlayAreas
-rand: Randomizer
+gridData: int[][] = int[4][4]
+pboxArr: Playbox[][] = Playbox[4][4]
+lastTileClicked: Tile = null
+final serialVersionUID: long = 1
«constructor»+GameWindow(s: String)
+actionPerformed(e: ActionEvent): void
+setup(): void
-addButtons(basic: GridBagConstraints): void
+tileClick(tile: Tile): void
+playboxClick(pbox: Playbox): void
+reset(): void
+mousePressed(e: MouseEvent): void
Randomizer
-tileIndicies: ArrayList<Integer>
-rotations[]: int
«constructor»+Randomizer()
-randomizeRotation(): void
-randomizeIndicies(): void
+getRotation(): int[]
+getIndice(): ArrayList<Integer>
+1
+2
«interface»
ActionListener
JPanel
+1
TileFlasher
-final FLASH_DELAY: int = 80
-timerCount: int
-flashCount: int = 3
-tile: Tile
-timer: Timer
«constructor»+TIleFlasher(tile: Tile)
+actionPerformed(e: ActionEvent): void
PlayAreas
-grid: JPanel
-LPanel: JPanel
-RPanel: JPanel
-tile: Tile[]
+pboxArr: Playbox[][]
+gridData: int[][]
«constructor»+PlayAreas()
-createGrid(basic: GridBagConstraints): void
-createLRPanels(basic: GridBagConstraints): void
+addTiles(indicies: ArrayList<Integer>, rotations: int[]): void
+updatePboxBorders(i: int, j: int, val: int[]): void
+updatePboxBorders(i: int, j: int): void
+repaintBorders(): void
+resetTiles(indicies: ArrayList<Integer>, rotations: int[]): void
+getTile(): Tile[]
+getGrid(): JPanel
+getLPanel(): JPanel
+getRPanel(): JPanel
JPanel
«interface»
MouseListener
Playbox
-borders: int[] = {1, 1, 1, 1}
-row: int
-col: int
-sidePanel: boolean
«constructor»+Playbox()
«constructor»+Playbox(row: int, col: int)
+getRow(): int
+getCol(): int
+isSidePanel(): boolean
+mousePressed(e: MouseEvent): void
+updateBorder(bdrs: int[]): void
+removeBorders(): void
+32
+1
+16
FileSetup
-filename: String
-lineCoords: float[][]
«constructor»+FileSetup(filename: String)
+getLineCoords(): float[][]
Tile
-lineCoords: float[]
-rotation: int = 0
+originalRotation: int
«constructor»+Tile(num: int, lineCoords: float[], originalRotation: int)
#paintComponent(g: Graphics): void
+getOriginalRotation(): int
+rotate(): void
+rotate(originalRotation: int): void
+mousePressed(e: MouseEvent): void
JPanel
«interface»
MouseListener

Version 5
Version 5's task flow was very similar to Version 4's including the same crunch at the end. The only difference was we had a few more days than the previous program to work on problems. We also had to do some heavy debugging for this iteration.

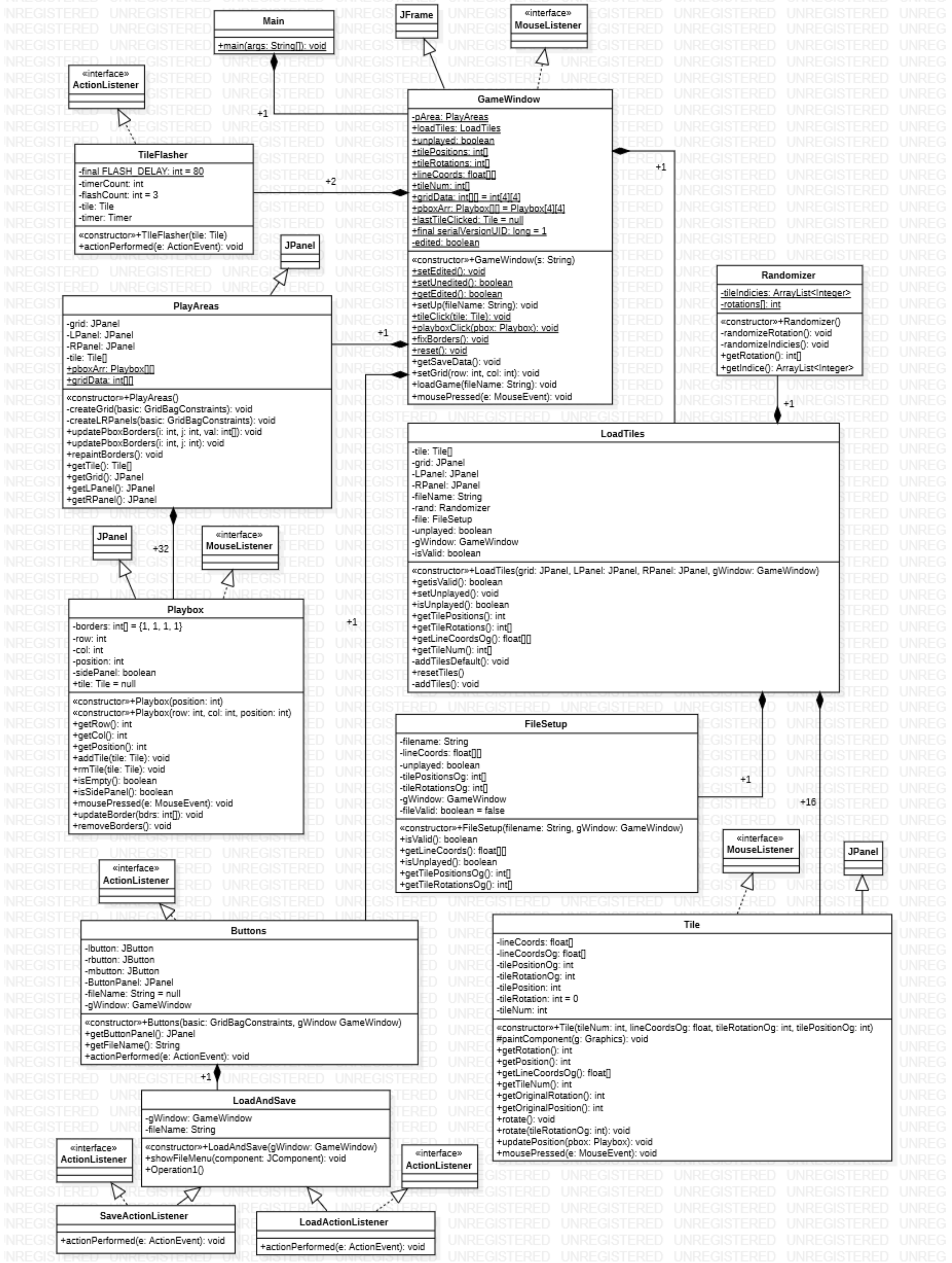
The focus for this version was loading and saving game files. This change was unexpected from our original prompt, but the team adapted well. Throughout this version, we continued to refactor the code to meet our new structure from Version 4, and we continued to check the new code for anything we may need to refactor. There were very few comments from our boss that needed to be addressed in this version, and most were related to documentation.

To finish the compartmentalizing process, we moved the buttons displayed in the GameWindow to their own class. This allowed us to sequester their design and functionality, finally making GameWindow only handle the graphical elements of the game.

The first meeting for this project was fairly extensive. Buck presented the code to the group and explained what was left to do on the project. Chester agreed to help with what was left and noticed a few things that needed to be changed to loosen the coupling. Ally and Buck also found an issue with deep copies. The documents were updated based on the code that was currently available and were later updated with the final code.

Version 5 built on our communication from Version 4. The group heavily relied on discord messages and meetings to notify other members when something was pushed to GitHub so the documents could be updated. After several days of communicating and updating both the code and documentation, a final meeting was called.

During the meeting, Buck covered the code requirements and went over what was completed. While going through the requirements, he noticed a bug that needed to be fixed. Each member then took a look at the code in the hopes that someone could figure out the problem. During the debugging process, several group members had to leave due to prior arrangements, but some progress was made. The group knew the problem arose from loading. This left Chester and Buck to work on the bug. After several hours of adding print statements to the code and testing different portions of the code, they finally worked out a solution. When the bug-free code was complete, the two pushed it, Ally updated the UML, and the team submitted the project for review.


Main
+main(args: String[]): void
JFrame
«interface» MouseListener
«interface» ActionListener
GameWindow
-pArea: PlayAreas
+loadTiles: LoadTiles
+unplayed: boolean
+tilePositions: int[]
+tileRotations: int[]
+lineCoords: float[][]
+tileNum: int[]
+gridData: int[][] = int[4][4]
+pboxArr: Playbox[][] = Playbox[4][4]
+lastTileClicked: Tile = null
+final serialVersionUID: long = 1
-edited: boolean
«constructor»+GameWindow(s: String)
+setEdited(): void
+setUnedited(): boolean
+getEdited(): boolean
+setUp(fileName: String): void
+tileClick(tile: Tile): void
+playboxClick(pbox: Playbox): void
+fixBorders(): void
+reset(): void
+getSaveData(): void
+setGrid(row: int, col: int): void
+loadGame(fileName: String): void
+mousePressed(e: MouseEvent): void
TileFlasher
-final FLASH_DELAY: int = 80
-timerCount: int
-flashCount: int = 3
-tile: Tile
-timer: Timer
«constructor»+TileFlasher(tile: Tile)
+actionPerformed(e: ActionEvent): void
+1
+2
+1
JPanel
PlayAreas
-grid: JPanel
-LPanel: JPanel
-RPanel: JPanel
-tile: Tile[]
+pboxArr: Playbox[][]
+gridData: int[][]
«constructor»+PlayAreas()
-createGrid(basic: GridBagConstraints): void
-createLRPanels(basic: GridBagConstraints): void
+updatePboxBorders(i: int, j: int, val: int[]): void
+updatePboxBorders(i: int, j: int): void
+repaintBorders(): void
+getTile(): Tile[]
+getGrid(): JPanel
+getLPanel(): JPanel
+getRPanel(): JPanel
+1
Randomizer
-tileIndicies: ArrayList<Integer>
-rotations[]: int
«constructor»+Randomizer()
-randomizeRotation(): void
-randomizeIndicies(): void
+getRotation(): int[]
+getIndice(): ArrayList<Integer>
+1
LoadTiles
-tile: Tile[]
-grid: JPanel
-LPanel: JPanel
-RPanel: JPanel
-fileName: String
-rand: Randomizer
-file: FileSetup
-unplayed: boolean
-gWindow: GameWindow
-isValid: boolean
«constructor»+LoadTiles(grid: JPanel, LPanel: JPanel, RPanel: JPanel, gWindow: GameWindow)
+getisValid(): boolean
+setUnplayed(): void
+isUnplayed(): boolean
+getTilePositions(): int
+getTileRotations(): int[]
+getLineCoordsOg(): float[][]
+getTileNum(): int[]
-addTilesDefault(): void
+resetTiles()
-addTiles(): void
JPanel
+32
«interface» MouseListener
+1
Playbox
-borders: int[] = {1, 1, 1, 1}
-row: int
-col: int
-position: int
-sidePanel: boolean
+tile: Tile = null
«constructor»+Playbox(position: int)
«constructor»+Playbox(row: int, col: int, position: int)
+getRow(): int
+getCol(): int
+getPosition(): int
+addTile(tile: Tile): void
+rmTile(tile: Tile): void
+isEmpty(): boolean
+isSidePanel(): boolean
+mousePressed(e: MouseEvent): void
+updateBorder(bdrs: int[]): void
+removeBorders(): void
FileSetup
-filename: String
-lineCoords: float[][]
-unplayed: boolean
-tilePositionsOg: int[]
-tileRotationsOg: int[]
-gWindow: GameWindow
-fileValid: boolean = false
«constructor»+FileSetup(filename: String, gWindow: GameWindow)
+isValid(): boolean
+getLineCoords(): float[][]
+isUnplayed(): boolean
+getTilePositionsOg(): int[]
+getTileRotationsOg(): int[]
+1
+16
«interface» MouseListener
JPanel
«interface» ActionListener
Buttons
-lbutton: JButton
-rbutton: JButton
-mbutton: JButton
-ButtonPanel: JPanel
-fileName: String = null
-gWindow: GameWindow
«constructor»+Buttons(basic: GridBagConstraints, gWindow GameWindow)
+getButtonPanel(): JPanel
+getFileName(): String
+actionPerformed(e: ActionEvent): void
Tile
-lineCoords: float[]
-lineCoordsOg: float[]
-tilePositionOg: int
-tileRotationOg: int
-tilePosition: int
-tileRotation: int = 0
-tileNum: int
«constructor»+Tile(tileNum: int, lineCoordsOg: float, tileRotationOg: int, tilePositionOg: int)
#paintComponent(g: Graphics): void
+getRotation(): int
+getPosition(): int
+getLineCoordsOg(): float[]
+getTileNum(): int
+getOriginalRotation(): int
+getOriginalPosition(): int
+rotate(): void
+rotate(tileRotationOg: int): void
+updatePosition(pbox: Playbox): void
+mousePressed(e: MouseEvent): void
+1
LoadAndSave
-gWindow: GameWindow
-fileName: String
«constructor»+LoadAndSave(gWindow: GameWindow)
+showFileMenu(component: JComponent): void
+Operation1()
«interface» ActionListener
«interface» ActionListener
SaveActionListener
+actionPerformed(e: ActionEvent): void
LoadActionListener
+actionPerformed(e: ActionEvent): void

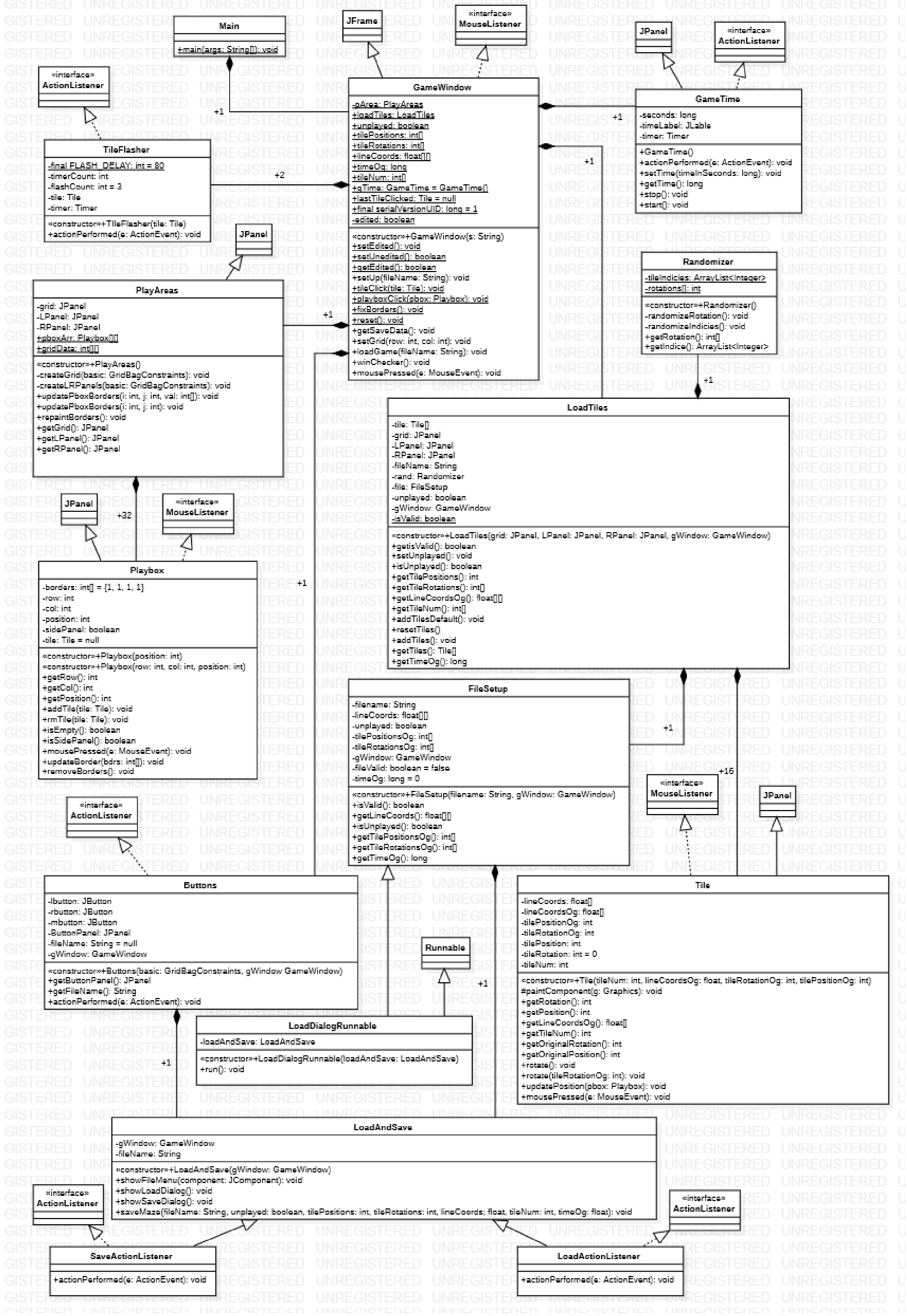
Version 6
Version 6, the final implementation of the project. From Version 5, we had a mostly functional game. The player could load and save files, reset their game, have fully randomized tile placement when starting a new game, and move and rotate the tiles on the board. The only thing that was left was telling the player when they had won. Our boss also added that we should let the player know how long it took them to win.

In all the previous versions we had discussed briefly how this may be implemented in the future since it seemed to be one of the more complex integrations. Given the early planning and everything else that had already been implemented, the task became significantly easier. Buck got an early start on the code and showed the finalized product at the first meeting. Chester asked about a few test cases, and a bug was found. Buck fixed the bug shortly after the meeting.

The meeting was rather sparse due to finals being just around the corner. Buck and Chester both made it, but the rest of the group was busy. Each member still ensured they'd finish their portion once the final code was pushed.

A few days after the meeting, the boss' suggested revisions were given to the group, and the group made changes accordingly. The group then had a final meeting to reassure everything was up to standards and submit the final version. The team discussed what else they imagined the program could do in the future and adjourned their meeting.

The Future of the Project
Any further changes or refactoring would need to be done by an external group. This would mean figuring out the code and making changes as needed. The process of figuring out another group's system is made simple with good documentation. Including Javadox, comments in the code, UML, and explanations of what the program should do.


Main
+main(args: String[]): void
JFrame
«interface» MouseListener
GameWindow
-pArea: PlayAreas
+loadTiles: LoadTiles
+unplayed: boolean
+tilePositions: int[]
+tileRotations: int[]
+lineCoords: float[][]
+timeOg: long
+tileNum: int[]
+gTime: GameTime = GameTime()
+lastTileClicked: Tile = null
+final serialVersionUID: long = 1
-edited: boolean
«constructor»+GameWindow(s: String)
+setEdited(): void
+setUnedited(): boolean
+getEdited(): boolean
+setUp(fileName: String): void
+tileClick(tile: Tile): void
+playboxClick(pbox: Playbox): void
+fixBorders(): void
+reset(): void
+getSaveData(): void
+setGrid(row: int, col: int): void
+loadGame(fileName: String): void
+winChecker(): void
+mousePressed(e: MouseEvent): void
JPanel
«interface» ActionListener
GameTime
-seconds: long
-timeLabel: JLable
-timer: Timer
+GameTime()
+actionPerformed(e: ActionEvent): void
+setTime(timeInSeconds: long): void
+getTime(): long
+stop(): void
+start(): void
«interface» ActionListener
TileFlasher
-final FLASH_DELAY: int = 80
-timerCount: int
-flashCount: int = 3
-tile: Tile
-timer: Timer
«constructor»+TileFlasher(tile: Tile)
+actionPerformed(e: ActionEvent): void
+1
+2
+1
+1
+1
JPanel
PlayAreas
-grid: JPanel
-LPanel: JPanel
-RPanel: JPanel
+pboxArr: Playbox[][]
+gridData: int[][]
«constructor»+PlayAreas()
-createGrid(basic: GridBagConstraints): void
-createLRPanels(basic: GridBagConstraints): void
+updatePboxBorders(i: int, j: int, val: int[]): void
+updatePboxBorders(i: int, j: int): void
+repaintBorders(): void
+getGrid(): JPanel
+getLPanel(): JPanel
+getRPanel(): JPanel
Randomizer
-tileIndicies: ArrayList<Integer>
-rotations[]: int
«constructor»+Randomizer()
-randomizeRotation(): void
-randomizeIndicies(): void
+getRotation(): int[]
+getIndice(): ArrayList<Integer>
+1
LoadTiles
-tile: Tile[]
-grid: JPanel
-LPanel: JPanel
-RPanel: JPanel
-fileName: String
-rand: Randomizer
-file: FileSetup
-unplayed: boolean
-gWindow: GameWindow
-isValid: boolean
«constructor»+LoadTiles(grid: JPanel, LPanel: JPanel, RPanel: JPanel, gWindow: GameWindow)
+getisValid(): boolean
+setUnplayed(): void
+isUnplayed(): boolean
+getTilePositions(): int
+getTileRotations(): int[]
+getLineCoordsOg(): float[][]
+getTileNum(): int[]
+addTilesDefault(): void
+resetTiles()
+addTiles(): void
+getTiles(): Tile[]
+getTimeOg(): long
JPanel
+32
«interface» MouseListener
+1
Playbox
-borders: int[] = {1, 1, 1, 1}
-row: int
-col: int
-position: int
-sidePanel: boolean
-tile: Tile = null
«constructor»+Playbox(position: int)
«constructor»+Playbox(row: int, col: int, position: int)
+getRow(): int
+getCol(): int
+getPosition(): int
+addTile(tile: Tile): void
+rmTile(tile: Tile): void
+isEmpty(): boolean
+isSidePanel(): boolean
+mousePressed(e: MouseEvent): void
+updateBorder(bdrs: int[]): void
+removeBorders(): void
FileSetup
-filename: String
-lineCoords: float[][]
-unplayed: boolean
-tilePositionsOg: int[]
-tileRotationsOg: int[]
-gWindow: GameWindow
-fileValid: boolean = false
-timeOg: long = 0
«constructor»+FileSetup(filename: String, gWindow: GameWindow)
+isValid(): boolean
+getLineCoords(): float[][]
+isUnplayed(): boolean
+getTilePositionsOg(): int[]
+getTileRotationsOg(): int[]
+getTimeOg(): long
+1
+16
«interface» MouseListener
JPanel
«interface» ActionListener
Buttons
-lbutton: JButton
-rbutton: JButton
-mbutton: JButton
-ButtonPanel: JPanel
-fileName: String = null
-gWindow: GameWindow
«constructor»+Buttons(basic: GridBagConstraints, gWindow GameWindow)
+getButtonPanel(): JPanel
+getFileName(): String
+actionPerformed(e: ActionEvent): void
Runnable
+1
Tile
-lineCoords: float[]
-lineCoordsOg: float[]
-tilePositionOg: int
-tileRotationOg: int
-tilePosition: int
-tileRotation: int = 0
-tileNum: int
«constructor»+Tile(tileNum: int, lineCoordsOg: float, tileRotationOg: int, tilePositionOg: int)
#paintComponent(g: Graphics): void
+getRotation(): int
+getPosition(): int
+getLineCoordsOg(): float[]
+getTileNum(): int
+getOriginalRotation(): int
+getOriginalPosition(): int
+rotate(): void
+rotate(tileRotationOg: int): void
+updatePosition(pbox: Playbox): void
+mousePressed(e: MouseEvent): void
LoadDialogRunnable
-loadAndSave: LoadAndSave
«constructor»+LoadDialogRunnable(loadAndSave: LoadAndSave)
+run(): void
+1
LoadAndSave
-gWindow: GameWindow
-fileName: String
«constructor»+LoadAndSave(gWindow: GameWindow)
+showFileMenu(component: JComponent): void
+showLoadDialog(): void
+showSaveDialog(): void
+saveMaze(fileName: String, unplayed: boolean, tilePositions: int, tileRotations: int, lineCoords; float, tileNum: int, timeOg: float): void
«interface» ActionListener
«interface» ActionListener
SaveActionListener
+actionPerformed(e: ActionEvent): void
LoadActionListener
+actionPerformed(e: ActionEvent): void