Team 5

Project 4

Software Engineering

Cristian Cedeno
Ethan Hannen
Fahad Dawood
Nicholas Boyum

Use case: User selects AI level 1

Actors: Player 1, AI

Steps:

Actor Action	System Response
1. Player 1 selects AI Level 1	2. Systems sets the AI level to 1
3. Player 1 selects a square	4. the system starts the game, and the AI makes a selection.
5. Player 1 selects a square	6. AI makes a selection.
7. Player 1 selects a square and creates a row	7. System Displays time's up Player wins

Use case: User selects AI level2

Actors: Player 1, AI

Steps:

Actor Action	System Response
1. Player 1 selects AI Level 2	2. Systems sets the AI level to 2
3. Player 1 selects a square	4. the system starts the game, and the AI makes a selection.
5. Player 1 selects a square	6. AI makes a selection then displays player 1 turn
7. Player 1 selects a square	7. AI makes a selection and creates a row then system displays time's up AI wins

Use case: User selects checker theme

Actors: user

Steps:

Actor Action	System Response
1. User selects checker theme	2. Systems changes the GUI to a cheker them

Use case: User selects Winter theme

Actors: User

Steps:

Actor Action	System Response
1. User selects checker theme	2. Systems changes the GUI to a Winter theme

Use case: User selects default them

Actors: User

Steps:

Actor Action	System Response
1. User selects default theme	2. Systems changes the GUI to the default theme

Use case: User selects Two Player

Actors: Player 1, Player 2

Steps:

Actor Action	System Response
1. Player 1 selects Two player	2. System creates a new board game and displays Player 1's turn
3. Player 1 selects a square	4. system checks condition and displays player 2 turn
5. Player 2 selects a square	6. system checks condition and displays player 1 turn
7. Player 1 selects a square	8. system checks condition and displays player 2 turn

9. Player 2 selects a square	10. system checks condition and displays player 1 turn
11. Player 1 selects a square and creates a row	12. system checks condition and displays time is up player1 win

Use case: User selects new game

Actors: Player 1

Steps:

Actor Action	System Response
1. Player 1 selects new game	2. TicTacToe creates a new game and new UI. Game wipes all squares and resets them to an empty string. A countdown is started. Player 1's turn is indicated in the message block.

Use case: Player 1 wins game (AI Player loses game)

Actors: Player 1, Computer Player

Steps:

Actor Action	System Response
1. Player 1 selects new game	2. Use case 'User selects new game' initiated
3. Player 1 selects a square to mark with X	4. Checks if a win condition exists, if not then reset timer. Display Computer's turn in message block. Locks the marked square so no additional marks can be made.
5. Computer Player sets a square to mark with O	6. Same steps as before if not win condition.
7. Player 1 and computer mark back and forth until Player 1 or computer creates a subsequent row, column, or diagonal with X's or O's according to the k's value	8. System checks win condition, verifies a win exists with all X's. Stops timer. Freezes squares. Updates message displaying time's up AI wins

Use case: Board is full and results in a draw

Actors: Player 1, AI

Steps:

Actor Action	System Response
1. Player selects new game	2. Use case 'User selects new game' initiated
3. Player 1 selects a square to mark with X	4. Checks if a win condition exists, if not then reset timer. Display Computer's turn in message block. Locks the marked square so no additional marks can be made.
5. Computer player sets a square to mark with O	6. Same steps as before if not win condition.
7. Player 1 and computer mark back and forth until all squares are populated with X's and O's.	8. System checks win condition according to the k's value. A win condition does not exist. Checks how many marks have been made. Verifies that 9 marks have been entered into the game board. Stops the timer and updates the message to display "Draw!"

Use case: Player loses game – timer runs out

Actors: Player 1, AI

Steps:

Actor Action	System Response
1. Player selects new game	2. Use case 'User selects new game' initiated
3. Player 1 selects a square to mark with X	4. Checks if a win condition exists, if not then reset timer. Display Computer's turn in message block. Locks the marked square so no additional marks can be made.
5. Computer Player sets a square to mark with O	6. Same steps as before if not win condition
7. Player 1 does not enter a mark. Timer runs out.	8. System responds by freezing the game board, stopping the timer, and displays that Player 1 won in the message block.

Use case: Sets board dimension

Actors: Player 1

Steps:

Actor Action	System Response
1. Player 1 selects and enters the values of m and n.	2. Systems capture the new values of M and N

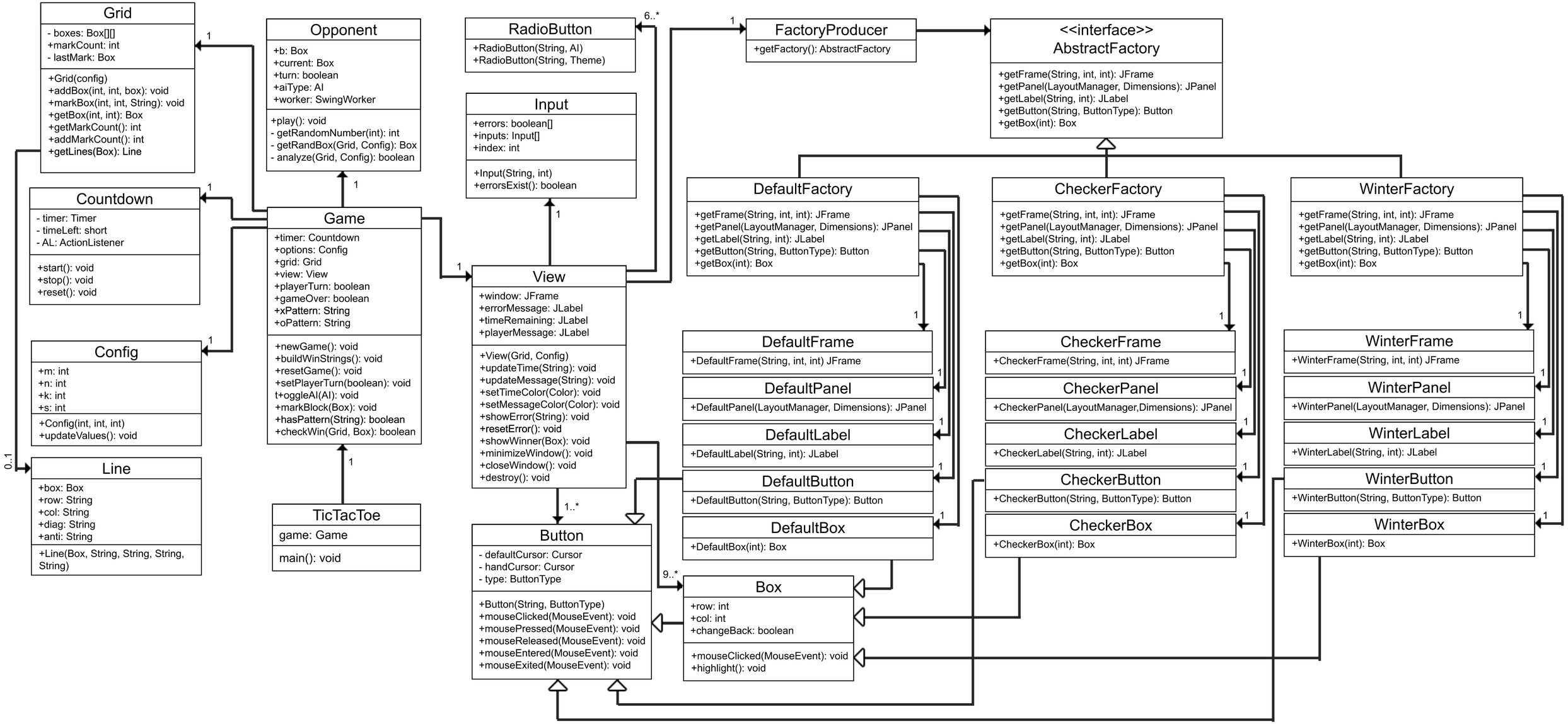
3. Player 1 clicks on the new game button	4. TicTacToe creates a new game and new UI. Game wipes all squares, resets them to an empty string, and sets the dimension of the board according to the M and N values. A countdown is started. Player 1's turn is indicated in the message block.

Use case: Sets k pieces to win

Actors: Player 1

Steps:

Actor Action	**System Response**
1. Player 1 selects and enters the value of K.	2. Systems capture the new value of K
3. Player 1 clicks on the new game button	4. TicTacToe creates a new game and new UI. Game sets new winning condition according to the K value, wipes all squares, resets them to an empty string. A countdown is started. Player 1's turn is indicated in the message block.


Grid
- boxes: Box[][]
+markCount: int
- lastMark: Box
+Grid(config)
+addBox(int, int, box): void
+markBox(int, int, String): void
+getBox(int, int): Box
+getMarkCount(): int
+addMarkCount(): int
+getLines(Box): Line
Opponent
+b: Box
+current: Box
+turn: boolean
+aiType: AI
+worker: SwingWorker
+play(): void
- getRandomNumber(int): int
- getRandBox(Grid, Config): Box
- analyze(Grid, Config): boolean
RadioButton
+RadioButton(String, AI)
+RadioButton(String, Theme)
FactoryProducer
+getFactory(): AbstractFactory
<<interface>>
AbstractFactory
+getFrame(String, int, int): JFrame
+getPanel(LayoutManager, Dimensions): JPanel
+getLabel(String, int): JLabel
+getButton(String, ButtonType): Button
+getBox(int): Box
Input
+errors: boolean[]
+inputs: Input[]
+index: int
+Input(String, int)
+errorsExist(): boolean
Countdown
- timer: Timer
- timeLeft: short
- AL: ActionListener
+start(): void
+stop(): void
+reset(): void
Game
+timer: Countdown
+options: Config
+grid: Grid
+view: View
+playerTurn: boolean
+gameOver: boolean
+xPattern: String
+oPattern: String
+newGame(): void
+buildWinStrings(): void
+resetGame(): void
+setPlayerTurn(boolean): void
t+oggleAI(AI): void
+markBlock(Box): void
+hasPattern(String): boolean
+checkWin(Grid, Box): boolean
View
+window: JFrame
+errorMessage: JLabel
+timeRemaining: JLabel
+playerMessage: JLabel
+View(Grid, Config)
+updateTime(String): void
+updateMessage(String): void
+setTimeColor(Color): void
+setMessageColor(Color): void
+showError(String): void
+resetError(): void
+showWinner(Box): void
+minimizeWindow(): void
+closeWindow(): void
+destroy(): void
DefaultFactory
+getFrame(String, int, int): JFrame
+getPanel(LayoutManager, Dimensions): JPanel
+getLabel(String, int): JLabel
+getButton(String, ButtonType): Button
+getBox(int): Box
CheckerFactory
+getFrame(String, int, int): JFrame
+getPanel(LayoutManager, Dimensions): JPanel
+getLabel(String, int): JLabel
+getButton(String, ButtonType): Button
+getBox(int): Box
WinterFactory
+getFrame(String, int, int): JFrame
+getPanel(LayoutManager, Dimensions): JPanel
+getLabel(String, int): JLabel
+getButton(String, ButtonType): Button
+getBox(int): Box
Config
+m: int
+n: int
+k: int
+s: int
+Config(int, int, int)
+updateValues(): void
DefaultFrame
+DefaultFrame(String, int, int) JFrame
DefaultPanel
+DefaultPanel(LayoutManager, Dimensions): JPanel
DefaultLabel
+DefaultLabel(String, int): JLabel
DefaultButton
+DefaultButton(String, ButtonType): Button
DefaultBox
+DefaultBox(int): Box
CheckerFrame
+CheckerFrame(String, int, int) JFrame
CheckerPanel
+CheckerPanel(LayoutManager,Dimensions): JPanel
CheckerLabel
+CheckerLabel(String, int): JLabel
CheckerButton
+CheckerButton(String, ButtonType): Button
CheckerBox
+CheckerBox(int): Box
WinterFrame
+WinterFrame(String, int, int) JFrame
WinterPanel
+WinterPanel(LayoutManager, Dimensions): JPanel
WinterLabel
+WinterLabel(String, int): JLabel
WinterButton
+WinterButton(String, ButtonType): Button
WinterBox
+WinterBox(int): Box
Line
+box: Box
+row: String
+col: String
+diag: String
+anti: String
+Line(Box, String, String, String, String)
TicTacToe
game: Game
main(): void
Button
- defaultCursor: Cursor
- handCursor: Cursor
- type: ButtonType
+Button(String, ButtonType)
+mouseClicked(MouseEvent): void
+mousePressed(MouseEvent): void
+mouseReleased(MouseEvent): void
+mouseEntered(MouseEvent): void
+mouseExited(MouseEvent): void
Box
+row: int
+col: int
+changeBack: boolean
+mouseClicked(MouseEvent): void
+highlight(): void
1
6..*
1..*
9..*
0..1

Test Case 1

Purpose:

1. To ensure that the 3 themes appear correctly

Requirement Traceability:
Requirement number appears here.(not sure what to put here)
Setup:

Run the java file TicTacToe.java

Test Data:

Action	**Input**	**Expected Output**
Run the java file		The tictactoe GUI appears
Cycle through themes	Click checker theme	GUI update to a Checker board color pattern
	Click winter theme	GUI update to winter color pattern (light blue & white)
	Click default theme	Returns GUI to default (all black)

Notes:

Test Case 2

Purpose:

1. To ensure the board sizes update correctly

Requirement Traceability:
Requirement number appears here.(not sure what to put here)
Setup:

Run the java file TicTacToe.java

Test Data:

Action	**Input**	**Expected Output**
Run the java file		The tictactoe GUI appears
Change board dimmensions	Type a number into M and N text box then click new game Do this for multiple numbers 1-9	The gui should update the height and width of the board accordingly
Test invalid number	Type a number larger than 9 and less than 1 into M, N, or K	“Invalid Range: Must be 3 ~ 9” error should be visible
Test invalid character	Type any non number into M, N, or K	“Invalid Input” Should be visible

Notes:

Test Case 3

Purpose:

1. Test whether running out the timer ends the game , as well as test visual cues of timer running out

Requirement Traceability:
Requirement number appears here.(not sure what to put here)
Setup:

Run the java file TicTacToe.java

Test Data:

Action	**Input**	**Expected Output**
Run the java file		The tictactoe GUI appears
Change board dimmensions	Type a number into M and N text box then click new game Do this for multiple numbers 1-9	The gui should update the height and width of the board accordingly
Start the game	Click any of the squares	The first move is made the timer starts
Let the time run out for player 2	Do nothing for 7 seconds Let last 3 seconds run out	“Time Remaining “ and the number should turn red at “Time Remaining: 3” It should say “Times up! Player 1 wins!”
Let time run out for player 1	Start the game and get to player 1’s turn Do nothing for 7 seconds Let last 3 seconds run out	1 x and 1 O on the board and the GUI should say player 1’s Turn “Time Remaining “ and the number should turn red at “Time Remaining: 3” It should say “Times up! Player 2 wins!”

Notes:

Test Case 3

Purpose:

2. Test whether running out the timer ends the game , as well as test visual cues of timer running out

Requirement Traceability:
Requirement number appears here.(not sure what to put here)
Setup:

Run the java file TicTacToe.java

Test Data:

Action	Input	Expected Output
Run the java file		The tictactoe GUI appears
Start the game	Click any of the squares	The first move is made the timer starts
Let the time run out for player 2	Do nothing for 7 seconds Let last 3 seconds run out	“Time Remaining “ and the number should turn red at “Time Remaining: 3” It should say “Times up! Player 1 wins!”
Let time run out for player 1	Start the game and get to player 1’s turn Do nothing for 7 seconds Let last 3 seconds run out	1 x and 1 O on the board and the GUI should say player 1’s Turn “Time Remaining “ and the number should turn red at “Time Remaining: 3” It should say “Times up! Player 2 wins!”

Notes:

Test Case 4

Purpose:

1. Win a game and lose a game against level 1 AI

Requirement Traceability:
Requirement number appears here.(not sure what to put here)
Setup:

Run the java file TicTacToe.java

Test Data:

Action	**Input**	**Expected Output**
Run the java file		The tictactoe GUI appears
Set game to ai mode	Click AI level 1	Should fill in the level 1 ai bubble
Start the game	Click any of the squares	The first move is made the timer starts
Win or lose	Win the tic tac toe game	After winning the game the bottom of the board Should say “Player 1 Wins!” And the winning line should be green
	Lose the tic tac toe game	After losing the game the bottom of the board Should say “Computer Wins!”And the winning line should be green

Notes:

Software Engineering Team 5

Cristian Cedeno
Ethan Hannen
Fahad Dawood
Nicholas Boyum

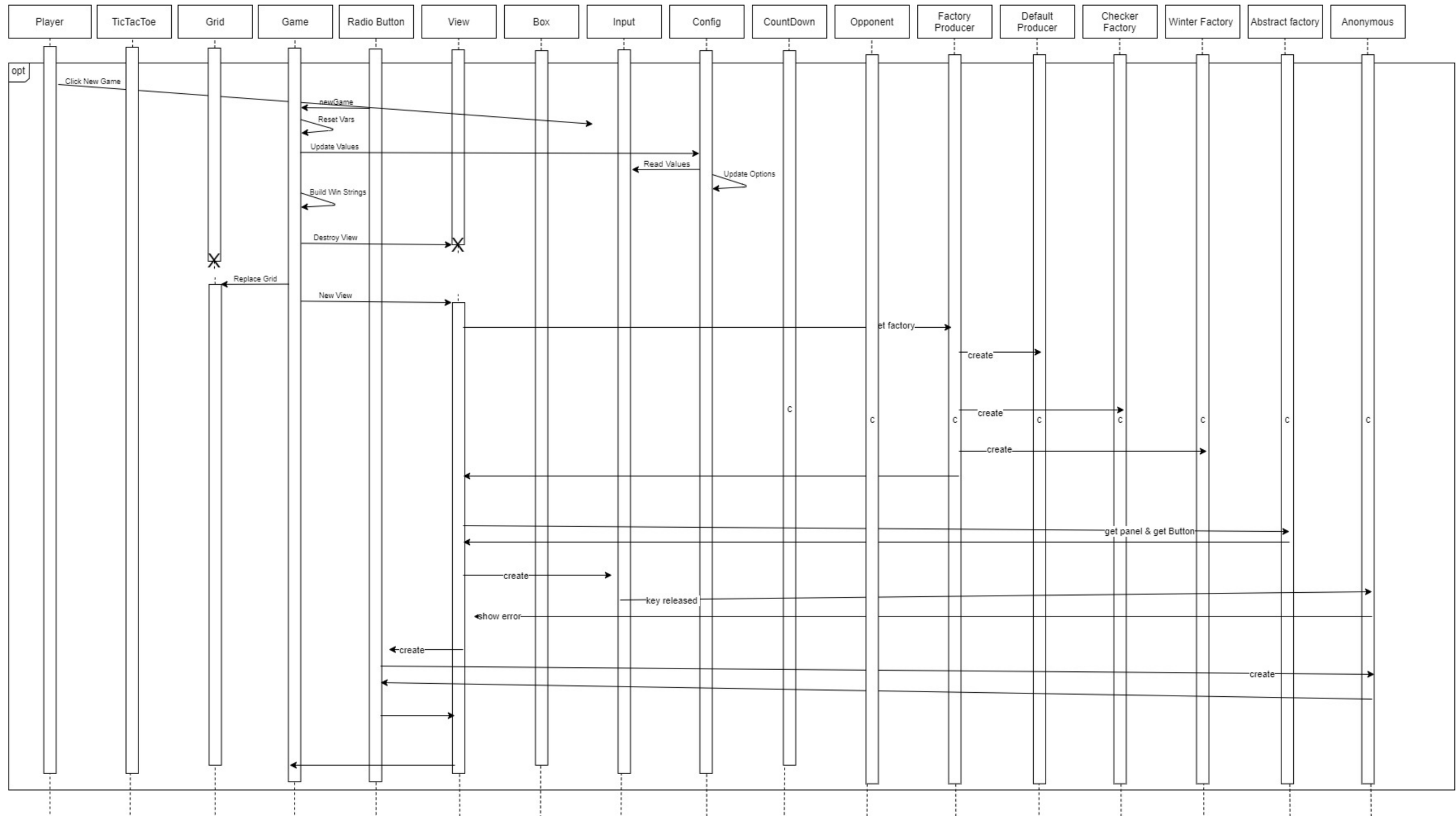
Unit Test Coverage Report

Team 5 wrote 18 test cases to cover various aspects of our program functionality. Classes such as View and Game are easy to test because they are driver files that perform very specific functions. Other classes, such as Opponent, are hard to test with unit tests due to interacting with an AI and the unpredictability of its moves. Testing Level 1 (basic) AI is easy to accomplish. Level 2, however, is not as easy to model. Regardless, we achieved an overall coverage rate of 71.2%.

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
SoftwareEngineering	71.2 %	2,444	990	3,434
src	64.4 %	1,790	988	2,778
cen4010.pa4	64.4 %	1,790	988	2,778
Opponent.java	28.4 %	160	404	564
Themes.java	40.4 %	115	170	285
Game.java	68.7 %	244	111	355
AbstractFactory.java	32.0 %	40	85	125
Input.java	66.0 %	130	67	197
Config.java	76.7 %	122	37	159
Box.java	62.3 %	48	29	77
CountDown.java	66.7 %	50	25	75
View.java	96.2 %	528	21	549
Button.java	81.0 %	85	20	105
Grid.java	96.1 %	268	11	279
TicTacToe.java	0.0 %	0	8	8
test	99.7 %	654	2	656

A majority of the code not covered has to do with AI actions and theme updates. The theme updates use listeners to react when a radio button is clicked. We could cover most of the other functionality through live Test Scripts.

Sequence 2
Player clicks 'New Game'
Team 5
Cristian Cedeno, Ethan Hannen,
Fahad Dawood, Nicholas Boyum



Sun Apr 18 23:13:36 2021 -0400 - Ethan Hannen: Update Opponent

Sun Apr 18 20:48:25 2021 -0400 - Ethan Hannen: Update AI

Sun Apr 18 17:36:46 2021 -0400 - Ethan Hannen: Add PDF

Sun Apr 18 17:35:15 2021 -0400 - Ethan Hannen: Update Opponent, update Class Diagram

Sun Apr 18 17:08:31 2021 -0400 - Ethan Hannen: Fix mixup in grid traversal

Sun Apr 18 16:47:13 2021 -0400 - Ethan Hannen: Fix some stuff

Sun Apr 18 16:43:00 2021 -0400 - NIckyptb: Fixed some of the broken tests

Sun Apr 18 15:53:45 2021 -0400 - NIckyptb: added more tests

Sun Apr 18 15:40:58 2021 -0400 - NIckyptb: Merge branch 'master' of https://github.com/UNF-CEN4010/T5.git

Sun Apr 18 15:05:54 2021 -0400 - NIckyptb: test coverage 69.1 > 70.7%

Sun Apr 18 15:04:24 2021 -0400 - Ethan Hannen: Add overloaded markBox method; rearrange Opponent

Sun Apr 18 14:50:42 2021 -0400 - Ethan Hannen: Merge branch 'master' of https://github.com/UNF-CEN4010/T5

Sun Apr 18 14:49:43 2021 -0400 - Ethan Hannen: Stage for advanced AI. Put Grid in its own file. Updated Winter theme slightly

Sun Apr 18 14:44:12 2021 -0400 - NIckyptb: added test cases for PA4

Sat Apr 17 16:23:46 2021 -0400 - Ethan Hannen: Changed summer theme to checker theme

Sat Apr 17 15:50:40 2021 -0400 - Ethan Hannen: Reworked abstract factory. Consolidated into Themes.java

Fri Apr 16 22:57:27 2021 -0400 - NIckyptb: created all the needed extra boxes and button classes for abstract factory

Fri Apr 16 13:36:42 2021 -0400 - NIckyptb: can't figure out how to access the input value from the text field any ideas?

Fri Apr 16 11:47:09 2021 -0400 - NIckyptb: I created the theme input box

Fri Apr 16 11:19:08 2021 -0400 - NIckyptb: figured out how to make changes to theme easily

Mon Apr 5 20:56:51 2021 -0400 - NIckyptb: continued last push

Tue Mar 30 16:20:26 2021 -0400 - Ethan Hannen: Add pdf versions of sequence diagrams for easier viewing

Mon Mar 29 12:50:07 2021 -0400 - Ethan Hannen: Add option for player vs player or player vs AI

Tue Mar 23 00:13:40 2021 -0400 - Ethan Hannen: Add project files

Mon Mar 22 23:50:49 2021 -0400 - crissmith222: updated sequence and class digram

Mon Mar 22 23:38:12 2021 -0400 - crissmith222: Use Case class diagrams

Mon Mar 22 21:34:00 2021 -0400 - crissmith222: Merge branch 'master' of https://github.com/UNF-CEN4010/T5

Mon Mar 22 21:33:38 2021 -0400 - crissmith222: fixed time remaining bug

Mon Mar 22 21:16:18 2021 -0400 - Ethan Hannen: Tabs to spaces and add headers

Mon Mar 22 21:06:43 2021 -0400 - Ethan Hannen: Add instructions

Mon Mar 22 19:50:54 2021 -0400 - Ethan Hannen: View fixes for M/N/K Inputs

Mon Mar 22 19:12:32 2021 -0400 - Ethan Hannen: Test unit fixes

Mon Mar 22 18:19:41 2021 -0400 - Ethan Hannen: Change view of Grid boxes and added access methods instead of direct static access

Mon Mar 22 17:05:17 2021 -0400 - Ethan Hannen: Remove unused code and general cleanup

Mon Mar 22 16:56:17 2021 -0400 - Ethan Hannen: Fix for crashes on m != n grids and better view for boxes

Mon Mar 22 14:42:36 2021 -0400 - n01425457: Fixed

Mon Mar 22 14:42:09 2021 -0400 - n01425457: Fixed

Mon Mar 22 14:41:37 2021 -0400 - n01425457: Fixed

Mon Mar 22 14:41:04 2021 -0400 - n01425457: Fixed

Mon Mar 22 14:40:39 2021 -0400 - n01425457: Fixed

Mon Mar 22 14:39:50 2021 -0400 - n01425457: Fixed

Mon Mar 22 14:39:20 2021 -0400 - n01425457: Fixed

Mon Mar 22 14:38:30 2021 -0400 - n01425457: Fixed

Mon Mar 22 14:37:20 2021 -0400 - n01425457: Fixed

Sun Mar 21 01:10:31 2021 -0400 - Ethan Hannen: Add button highlight animations and clean up view.java

Sun Mar 21 00:23:58 2021 -0400 - Ethan Hannen: Add opponent randomness

Sun Mar 21 00:09:37 2021 -0400 - Ethan Hannen: Add inherited classes Box and Input. Rename Grid to View and add Grid as a different local class to Game. View now represent the actual global program view whereas Grid represents just the player field. Added error handling for user input. Began working on rebuild logic for new game parameters.

Fri Mar 19 18:40:58 2021 -0400 - Ethan Hannen: Remove unnecessary code and add to do list

Fri Mar 19 18:23:52 2021 -0400 - Ethan Hannen: Added AI Opponent

Fri Mar 19 13:58:11 2021 -0400 - Ethan Hannen: Updates to game and grid

Thu Mar 18 18:21:11 2021 -0400 - Ethan Hannen: First pass at PA3 implementation

Wed Mar 17 22:34:13 2021 -0400 - Ethan Hannen: Update package identifier and headers

Wed Mar 17 22:32:17 2021 -0400 - Ethan Hannen: Add initial PA3 structure with copy of PA2 code