COMP 302

Chewy Lokum Legend - Phase 2

Design Discussion

Open IT

In the first phase of the project, we aimed to design the game as modular as possible so that further changes and extensions would be easily made. In order to build a modular design, we utilised design patterns such as strategy and subcriber/publisher so for each type of swap, match, explosion and generation, we created specialised classes as we call “handlers” in general. Moreover, we listed the names of those handler classes in a XML file within an order to be able to easily adjust the rules to be applied and their priority. Through this design, we do not need to change the existing classes and codes to remove or add new rules to handle new occasions, we just need to add a new class and state the name and the priority in the XML file.

For the second phase of the project, we are asked to bring a new type of level to the game which has a time limit for the player to reach the target score. Additionally, we are asked to introduce a new type of lokum to this type of levels which gives additional time when exploded. In our design discussions, we have decided to obtain this functionality by adding a new class called “TimeLokum” as a subtype of RegularLokum of our design so that each instance of TimeLokum can store the amount of additional time in its own field as well as it can be easily distinguished by new handler, TimeLokumExploder, which is going to be responsible for explosion of TimeLokum. For the new type of level, we extended our ScoreGoal class with another subclass called TimeScoreGoal. This type of goal states the amount of time for the particular level to be completed. As in our design in phase 1, the information about each level, the type, the target score, the number etc., is stored in another XML file. So for TimeScoreLevels, we are going to extend the structure of the XML file a little and adjust the XMLParser accordingly and that is all.

As the second requirement of the second phase, we are asked to bring special swaps. This is another easy task to be done with our design. To fullfil this requirement, we are going to add a few fields and methods to our GameEngine to fetch the allowed number of special swaps for each level and keep the record of them and allow the user to choose non-consecutive lokums and swap them when special swap option is selected. Allowing special swaps is trivial since GameEngine will just ignore the Board class' response to the swappability of two lokums because Board checks if they are consecutive and the swap results in a match.

To conclude, our design for the phase 1 was so well-thought that we will not have to make a change in the structure of the design. We are going to make only some additions and small changes. Furthermore, we are planning to polish the flaws of our implementation for phase 1 by revising the code and make the current version more coherent.

Use Case UC1: StartGame

Primary Actor: User

Stakeholders and Interests:

- User: Wants fast and accurate start of the game.

Preconditions: The game is executed by the user.

Success Guarantee (Postconditions): The game is started from a level according to the choice of the user. The game screen is displayed and the game begins.

Main Success Scenario (Basic Flow):

1- User selects “Play Game” button.

2- The System asks for the user’s name.

3- User enters a player name.

4- The system presents a dialog with two choices to the user which are start a new game or continue one of the previously saved games.

5- The user chooses to start a new game.

6- The system loads from XML and presents available levels for the chosen player to start.

7- The user selects a level.

8- The game is started from the selected level.

Extensions (Alternative Flows):

5-8a – The player chooses to continue one of the previously saved game sessions:

1- The user chooses to continue one of the previously saved game sessions.
2- The system presents available saved game states for the chosen player.
3- The player selects a game session.
4- The selected game session is opened and the user continues from where s/he left the game.

5-8b - Return to menu

1 – The user chooses to return to menu

2 – System closes the start game dialog and returns to the menu

*a - At any time, System fails

1- User restarts the game.

Frequency of Occurrence: Could be one for each game playıng action.

Use Case UC2: SwapLokums

Primary Actor: User

Stakeholders and Interests:

- User: Wants to swap the two selected Lokums as fast and accurate as possible.

Preconditions: The game is started by using the PlayGame use case.

Success Guarantee (Postconditions): The selected Lokums are swapped and the locations of the Lokums are deleted if necessary. The special Lokum is formed after the swapping if the swapping corresponds one of the special Lokum formation combinations.

Main Success Scenario (Basic Flow):

1- User selects the first lokum to be swapped.

2- User selects the second lokum to be swapped.

3- The system checks if the swapping two lokum is legal.

4- The swapping is done.

5- The line of three consecutive Lokums formed after the swapping is removed.

6- Score is calculated accordingly and updated.

7- Lokums above the removed line comes down to fill created space of the removed line, necessary number of new Lokums dropped to the empty squares.

8- System presents the new state of the board with new Lokums.

System repeats steps 5-8 until there is no Lokums needed to be removed on the new state of the board.

Extensions (Alternative Flows):

2-8a- Deselection of the first Lokum:

1- The first lokum is get deselected.

2-8b- User chooses two non-consecutive Lokums

1- System signals an warning to indicate that two lokums are not swapable.
2- Lokums are get deselecected.

5a- Line of four consecutive Lokums of the same color:

1- The line of four consecutive Lokums formed after the swapping is removed.

2- Striped Lokum is generated and put to correct position.

5b- Line of five consecutive Lokums of the same color:

1- The line of five consecutive Lokums formed after the swapping is removed.

2- Color Bomb Lokum is generated and put to correct position.

5c- Five Lokums of the same color forming a “T” shape:

1- Five Lokums of the same color forming a “T” shape generated after the swapping are removed.

2- Wrapped Lokum is generated and put to correct position.

5d- Only one of the selected Lokums is Striped:

1- Either an entire row or column is cleared.

5e- Only one of the selected Lokums is Wrapped:

1- The 3x3 area of surrounding Lokums are cleared, the Wrapped lokum falls, and explodes one more time.

5f- Only one of the selected Lokums is Color Bomb:

1- All lokums with the same color of the one it is matched with the swapped Lokum.

5g- Both of selected Lokums are Striped:

1- Both entire row and column are cleared.

5h- One of the selected Lokums is Striped, other one is Wrapped:

1- Both entire 3 consecutive rows and 3 consecutive columns are cleared.

5i- Both of selected Lokums are Wrapped:

1- The 6x5 area of surrounding Lokums are cleared, the Wrapped lokums fall, and explode one more time.

5j- One of the selected Lokums is Striped, other one is Color Bomb:

1- Lokums matching with the color of Striped Lokum and their row or column cleared.

5k- One of the selected Lokums is Wrapped, other one is Color Bomb:

1- Lokums matching with the color of Wrapped Lokum and their 3x3 area of surrounding Lokums are cleared , lokums fall, and explode one more time.

5l- Both of selected Lokums are Color Bomb:

1- All lokums are cleared.

6a – If the current level is a time based level and one of the selected lokums is a Time Lokum, then the system adds an additional time to the remaining time.

*a- At any time, System fails

1- User restarts the game.

Special Requirements:

- The colors and the shapes of the Lokums must be distinguished easily.

Frequency of Occurrence: Could be almost continuous.

Use Case UC3: SpecialSwapLokums

Primary Actor: User

Stakeholders and Interests:

- User: Wants to be able to select and swap two non-consecutive Lokums for the next swap

Preconditions: The game is started by using the PlayGame use case. The user have at least one special swaps left for the level.

Success Guarantee (Postconditions): The next swap can be done between any two available lokums on the board.

Main Success Scenario (Basic Flow):

1- User selects the special swap mode.

2- The system allows the user to select any two available lokums on the board.

3- User selects the first lokum to be swapped.

4- User selects the second lokum to be swapped.

5- The system checks if the swapping two lokum is legal.

6- The swapping is done.

7- Any possible explosion is exploded after the swapping is done.

8- Score is calculated accordingly and updated.

9- Lokums above the removed line comes down to fill created space of the removed line, necessary number of new Lokums dropped to the empty squares.

10- System presents the new state of the board with new Lokums.

System repeats steps 7-10 until there is no Lokums needed to be removed on the new state of the board.

Extensions (Alternative Flows):

4-10a- Deselection of the first Lokum:

1- The first lokum is get deselected.

7a- Line of three consecutive Lokums of the same color:

1- The line of three consecutive Lokums formed after the swapping is removed

7b- Line of four consecutive Lokums of the same color:

1- The line of four consecutive Lokums formed after the swapping is removed.

2- Striped Lokum is generated and put to correct position.

7c- Line of five consecutive Lokums of the same color:

1- The line of five consecutive Lokums formed after the swapping is removed.

2- Color Bomb Lokum is generated and put to correct position.

7d- Five Lokums of the same color forming a “T” shape:

1- Five Lokums of the same color forming a “T” shape generated after the swapping are removed.

2- Wrapped Lokum is generated and put to correct position.

7e- Only one of the selected Lokums is Striped:

1- Either an entire row or column is cleared.

7f- Only one of the selected Lokums is Wrapped:

1- The 3x3 area of surrounding Lokums are cleared, the Wrapped lokum falls, and explodes one more time.

7g- Only one of the selected Lokums is Color Bomb:

1- All lokums with the same color of the one it is matched with the swapped Lokum.

7h- Both of selected Lokums are Striped:

1- Both entire row and column are cleared.

7i- One of the selected Lokums is Striped, other one is Wrapped:

1- Both entire 3 consecutive rows and 3 consecutive columns are cleared.

7j- Both of selected Lokums are Wrapped:

1- The 6x5 area of surrounding Lokums are cleared, the Wrapped lokums fall, and explode one more time.

7k- One of the selected Lokums is Striped, other one is Color Bomb:

1- Lokums matching with the color of Striped Lokum and their row or column cleared.

7l- One of the selected Lokums is Wrapped, other one is Color Bomb:

1- Lokums matching with the color of Wrapped Lokum and their 3x3 area of surrounding Lokums are cleared , lokums fall, and explode one more time.

7m- Both of selected Lokums are Color Bomb:

2- All lokums are cleared.

8a – If the current level is a time based level and one of the selected lokums is a Time Lokum, then the system adds an additional time to the remaining time.

*a- At any time, System fails

1- User restarts the game.

Special Requirements:

- The colors and the shapes of the Lokums must be distinguished easily.

Frequency of Occurrence: Could be almost continuous.

Use Case UC4: SaveGame

Primary Actor: User

Stakeholders and Interests:

- User: Wants to save the current game session at the current state in order to be able continue from that state again at any time.

Preconditions: The game is started by using the PlayGame use case.

Success Guarantee (Postconditions): The game state is saved under the given player name at the beginning of the game.

Main Success Scenario (Basic Flow):

1- User presses the “Save” button.

2- The system saves the current game at the current state.

Extensions (Alternative Flows):

*a- At any time, System fails

1- User restarts the game.

Special Requirements:

- There should be enough place in the memory to save the game state.

Frequency of Occurrence: Could be few times for each game session.

Use Case UC5: GameOver

Primary Actor: User

Stakeholders and Interests:

- User: Wants to start the next level or to return to the game menu for starting another game or to exit the game completely

Preconditions: The game is over with either winning of the player or the losing.

Success Guarantee (Postconditions): The game state is saved under the given player name at the beginning of the game.

Main Success Scenario (Basic Flow):

1-The system saves the current session with player name, last level and total score.

2-The system presents the game over dialog with next level and return to menu options.

3-The user selects to continue to play the next level.

4-The system presents the next level.

Extensions (Alternative Flows):

1a – Game is over with a lose:

1-The system presents the game over dialog with return to menu.

3-4a – Return menu:

1- The user selects to return to menu
2- The system closes all dialogs and window and presents the main menu

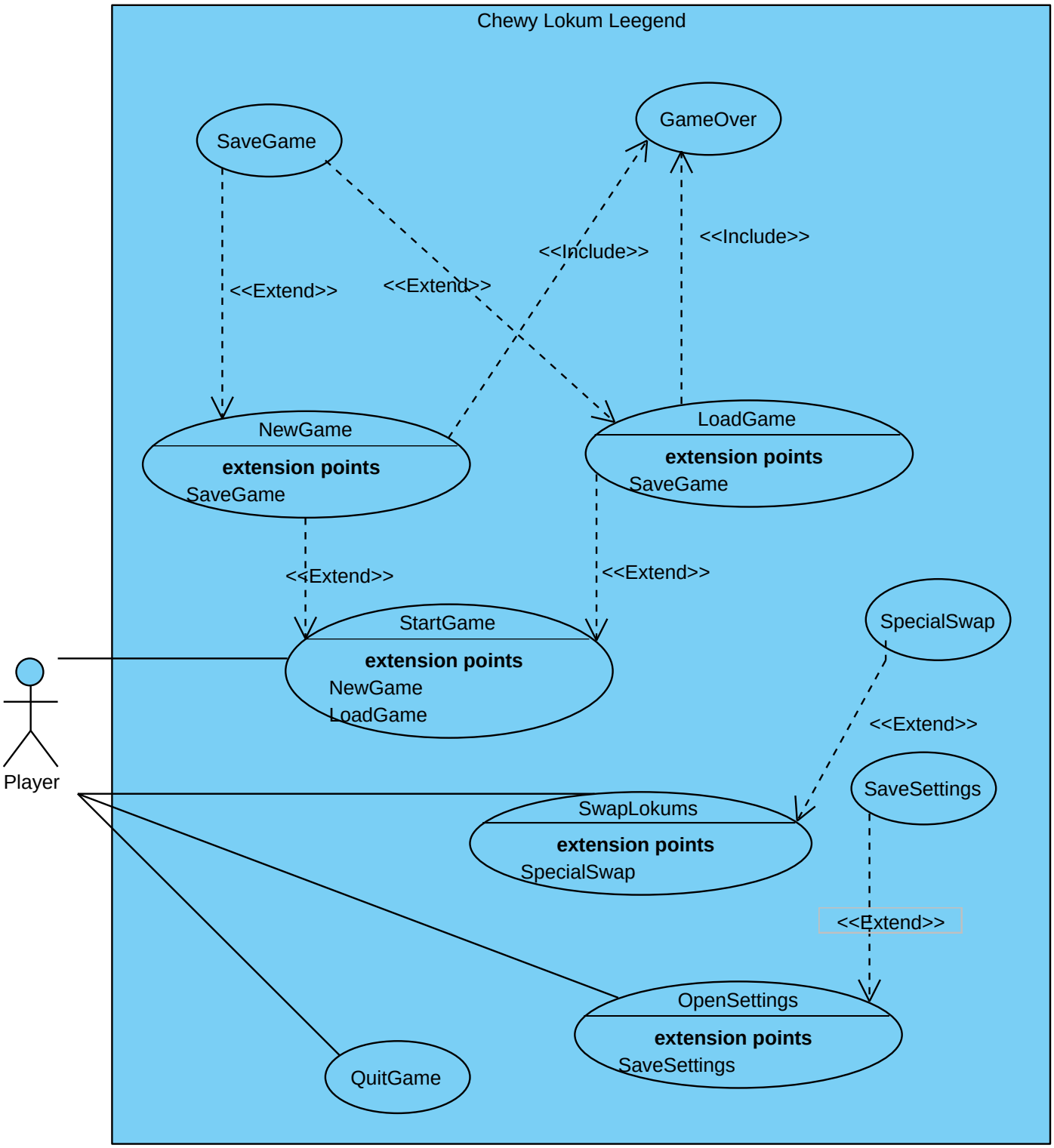
4a- No next level:

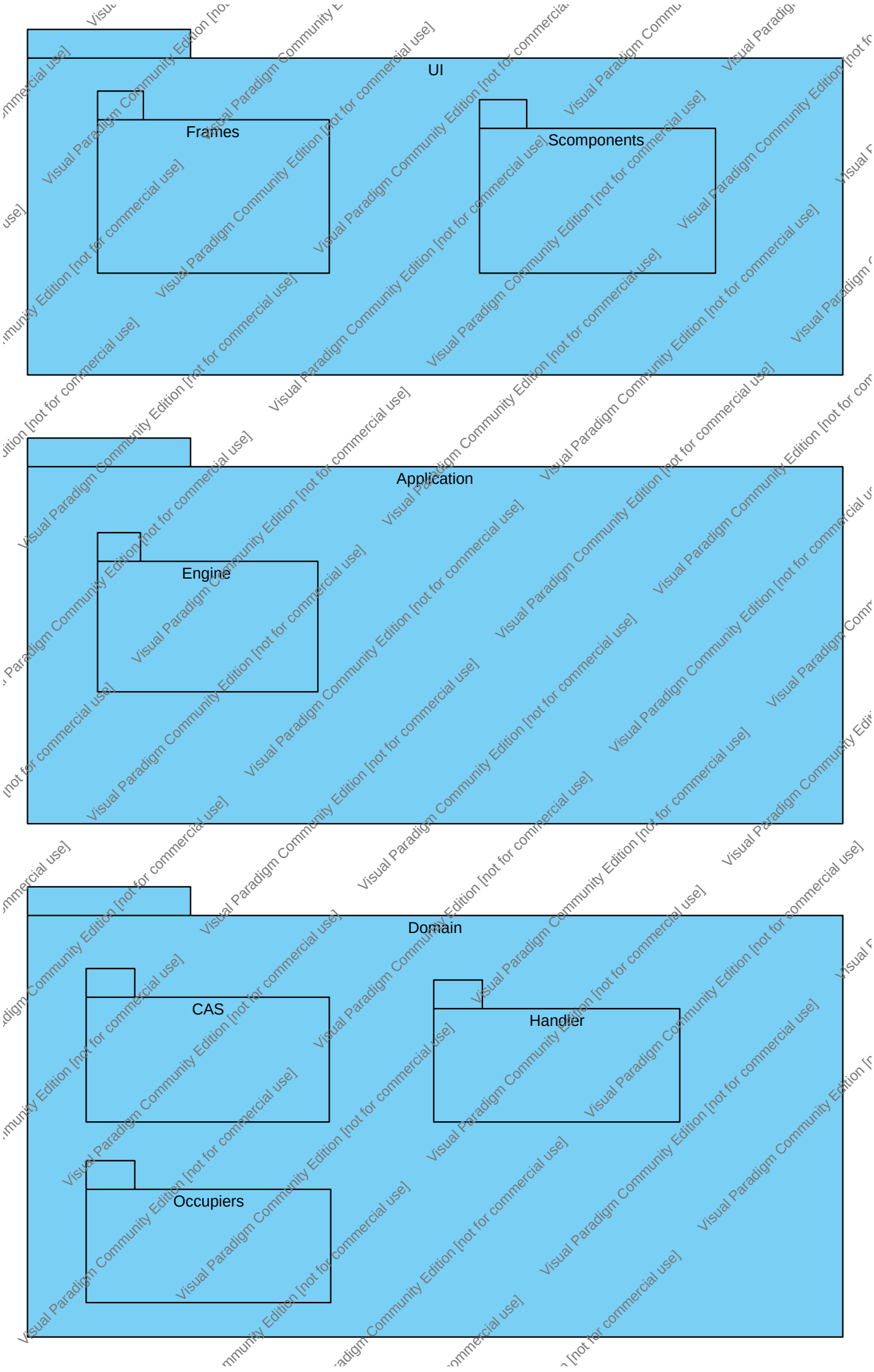
1- The system presents the previously won level again.

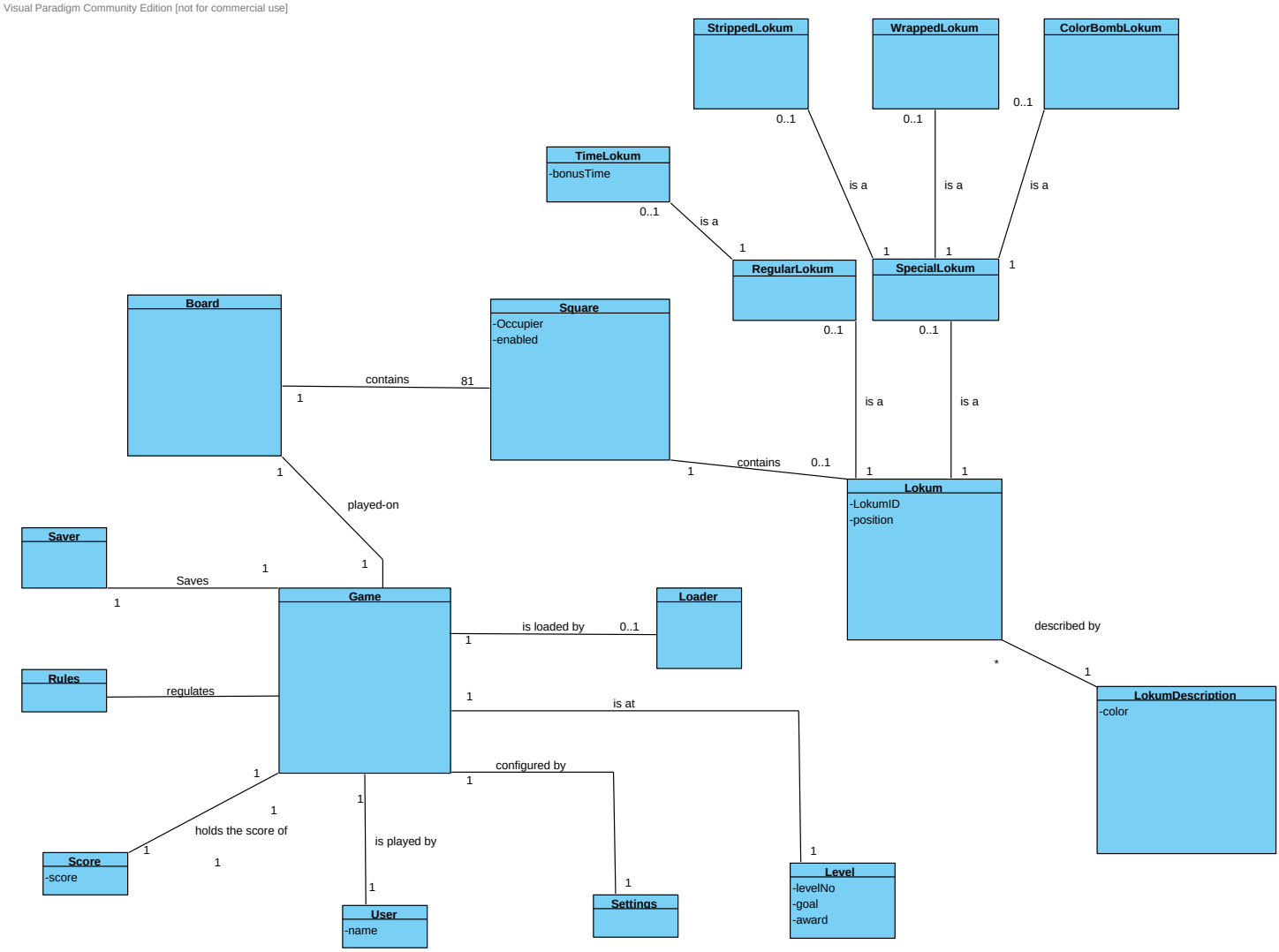
*a- At any time, System fails

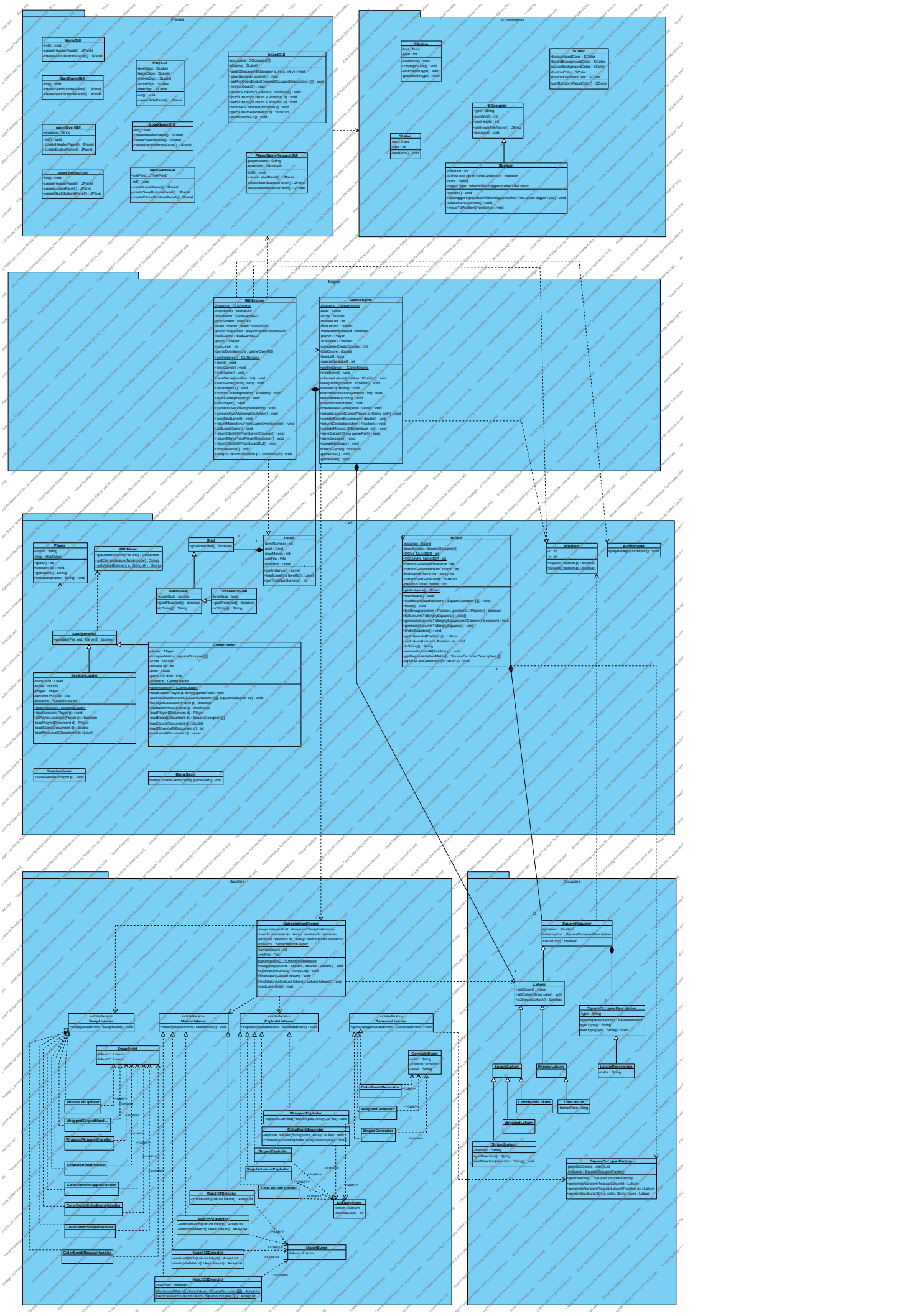
1- User restarts the game.

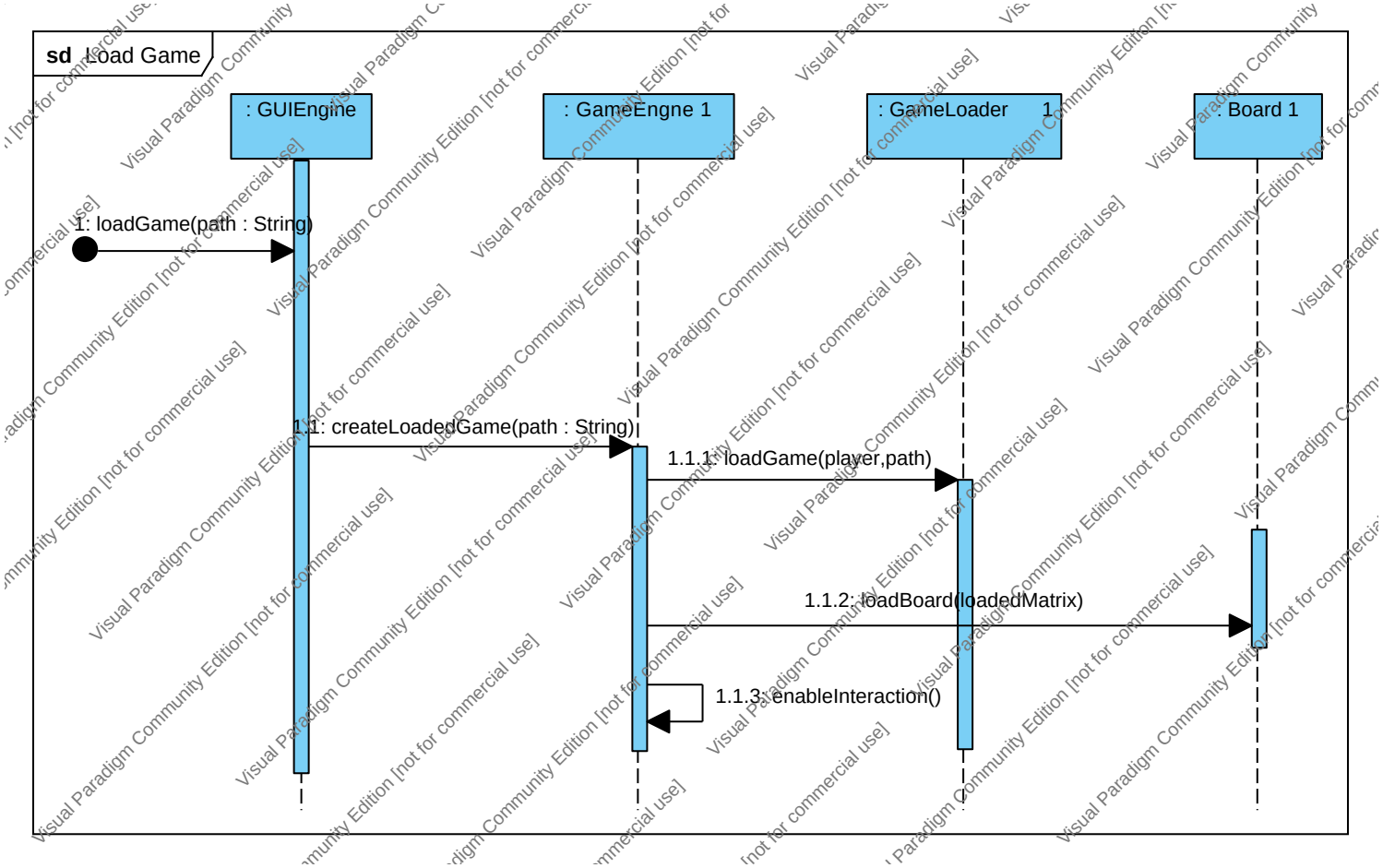
Frequency of Occurrence: For one for each game session.

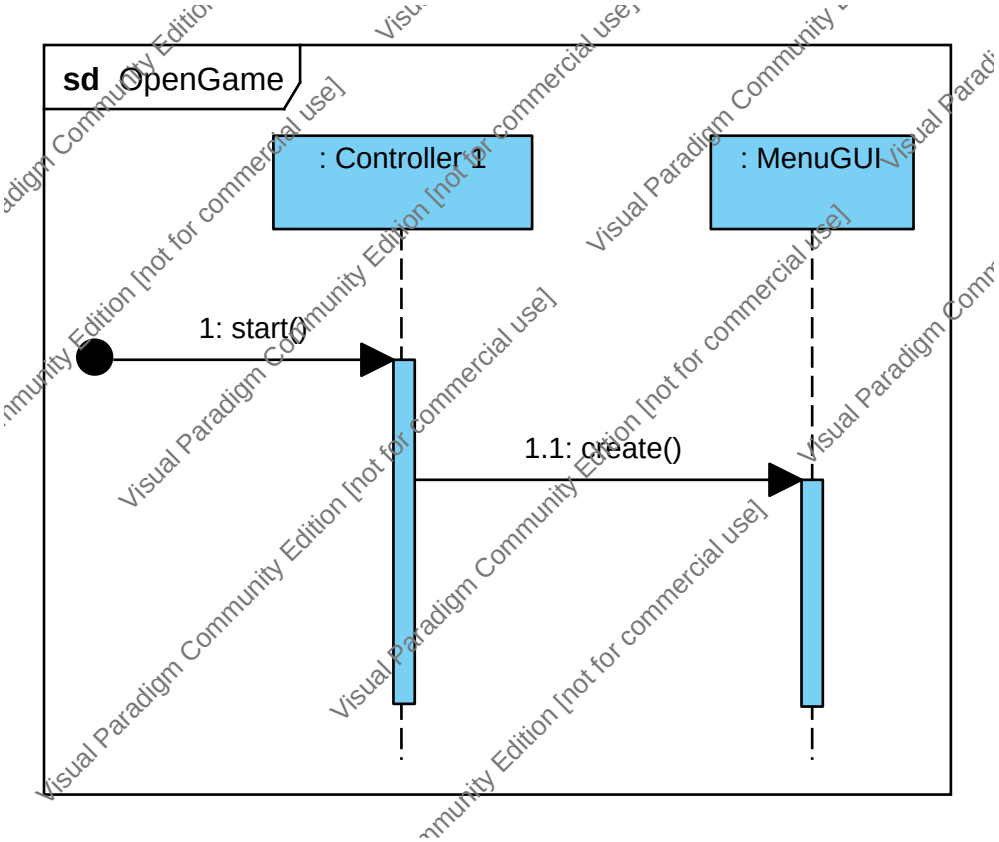


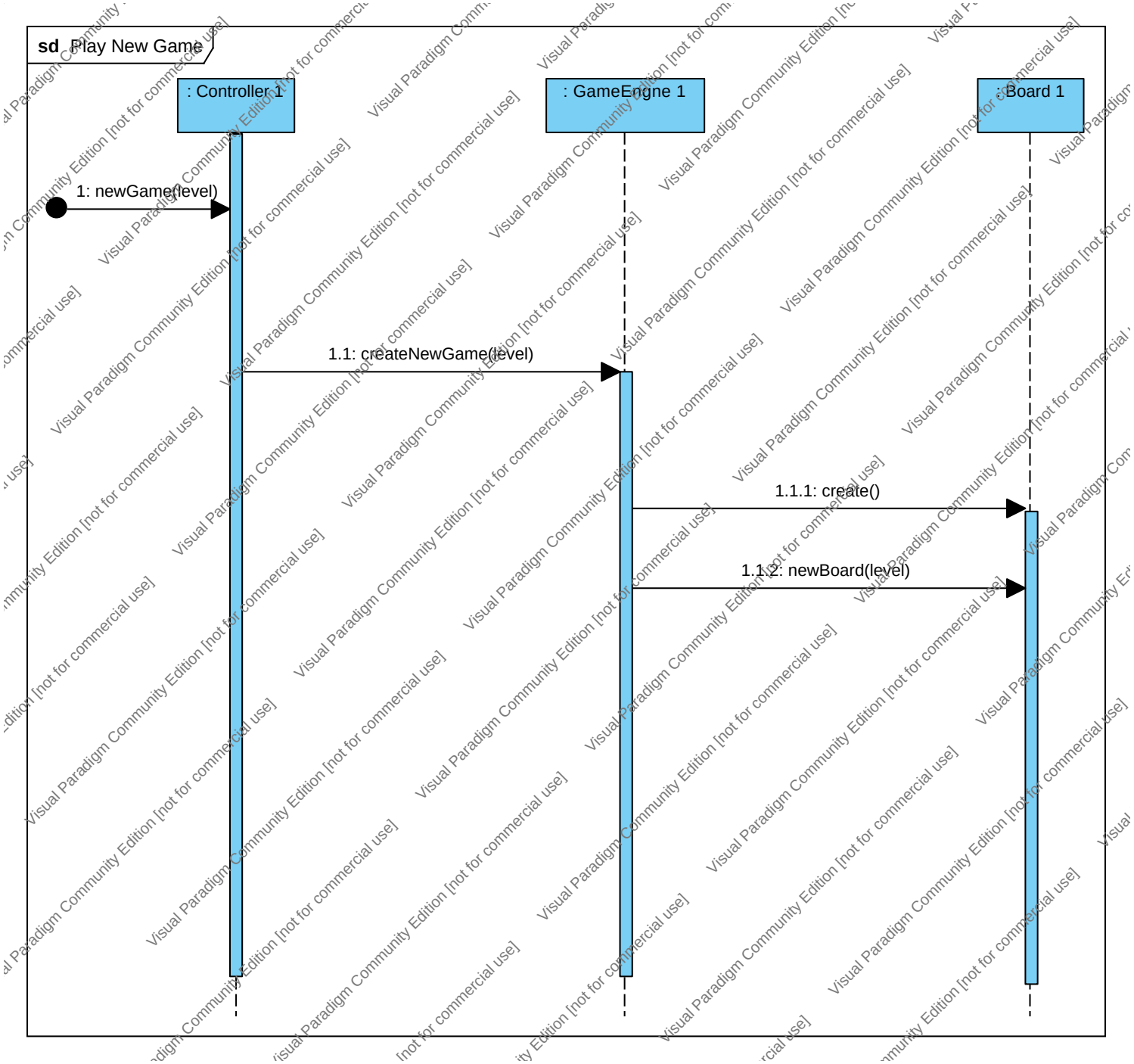

UI
Frames
Scomponents
Application
Engine
Domain
CAS
Handler
Occupiers

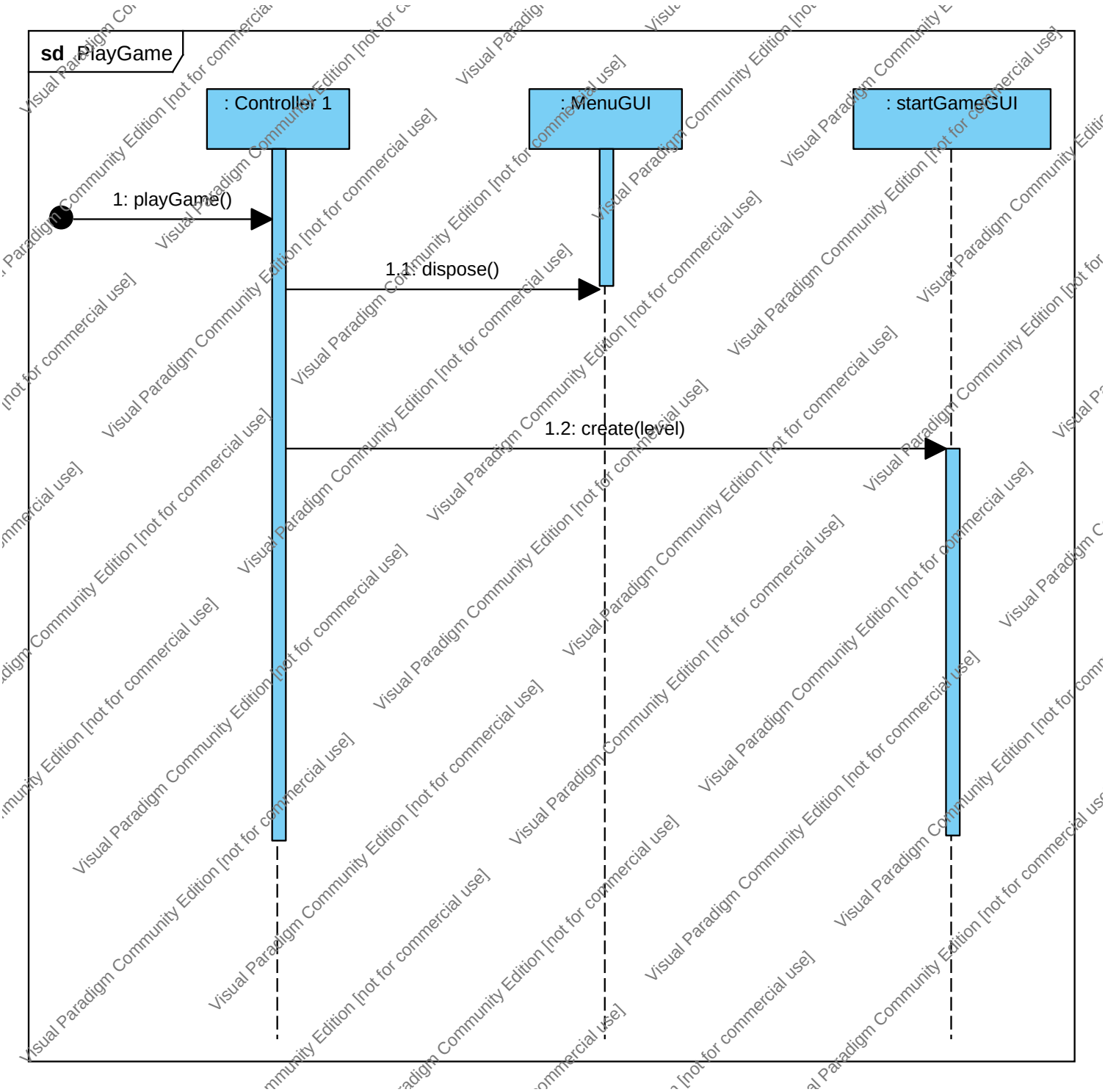


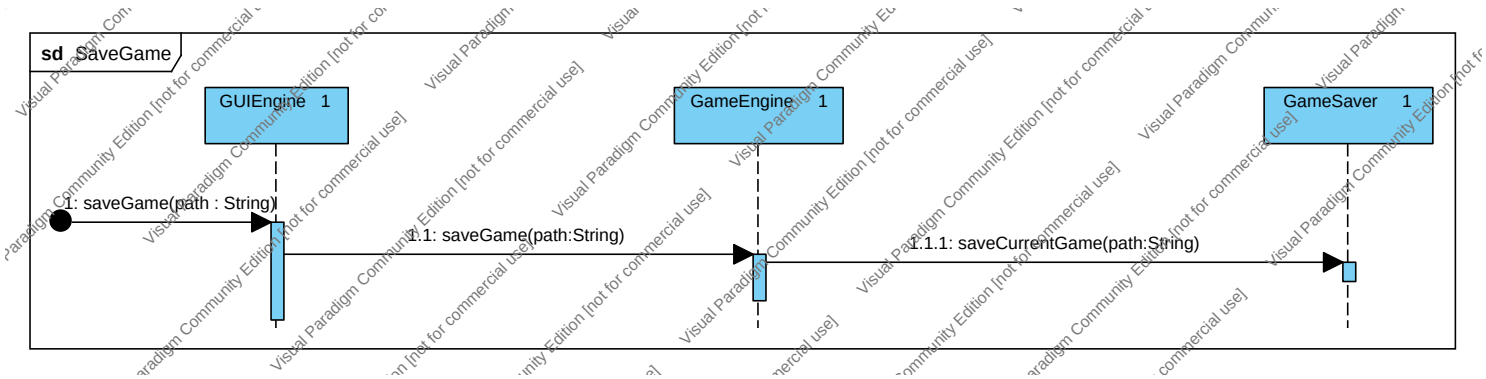

Frames
MenuGUI
-init() : void
-createHeaderPanel() : JPanel
-createMenuButtonsPanel() : JPanel
PlayGUI
-levelSign : SLabel
-targetSign : SLabel
-movesSign : SLabel
-scoreSign : SLabel
-timeSign : SLabel
-init() : void
-createSidePanel() : JPanel
boardGUI
-occupiers : SOccupier[][]
-glowing : SLabel
+addSOccupier(SOccupier s, int x, int y) : void
+glow(boolean visibility) : void
+setAndDrawBoard(SquareOccupierDescription [][]) : void
+refreshBoard() : void
+insertSLokum(SLokum s, Position p) : void
+putSLokum(SLokum s, Position p) : void
+setSLokum(SLokum s, Position p) : void
+removeSLokumAt(Position p) : void
+getSLokumAt(Position p) : SLokum
+printBoardGUI() : void
StartGameGUI
-init() : void
-createStartButtonsPanel() : JPanel
-createBackButtonsPanel() : JPanel
gameOverGUI
-situation : String
+init() : void
+createHeaderPanel() : JPanel
-createButtonsPanel() : JPanel
levelChoosenGUI
-init() : void
-createHeaderPanel() : JPanel
-createLevelsPanel() : JPanel
-createBackButtonsPanel() : JPanel
LoadGameGUI
-init() : void
-createHeaderPanel() : JPanel
-createSavesPanel() : JPanel
-createBackButtonsPanel() : JPanel
saveGameGUI
-textField : JTextField
-init() : void
-createLabelPanel() : JPanel
-createSaveButtonPanel() : JPanel
-createCancelButtonsPanel() : JPanel
PlayerNameRequestGUI
-playerName : String
-textField : JTextField
-init() : void
-createLabelPanel() : JPanel
-createStartButtonsPanel() : JPanel
-createBackButtonsPanel() : JPanel
SComponents
SButton
-font : Font
-type : int
-loadFont() : void
+changeSlide() : void
+setIcon(int type) : void
-getIcon(int type) : Icon
SColor
+backgroundColor : SColor
+boardBackgroundColor : SColor
+planetBackgroundColor : SColor
+buttonColor : SColor
+buttonDisabledColor : SColor
+getRandomPieceColor() : SColor
SLabel
-font : Font
-type : int
-loadFont() : void
SOccupier
-type : String
-iconWidth : int
-iconHeight : int
-getImageFileName() : String
+repaint() : void
SLokum
-distance : int
-isThisLastLokumToBeGenerated : boolean
-color : String
-triggerType : whatWillBeTriggeredAfterThisLokum.triggerType()
+init() : void
+setTriggerType(whatWillBeTriggeredAfterThisLokum.triggerType()) : void
+addLokumListeners() : void
+moveToPosition(Position p) : void
Engine
GUIEngine
-instance : GUIEngine
-mainMenu : MenuGUI
-startMenu : StartGameGUI
-playScreen : playGUI
-levelChooser : levelChooserGUI
-playerRequester : playerNameRequestGUI
-loadGame : loadGameGUI
-player : Player
-mainLevel : int
-gameOverWindow : gameOverGUI
+getInstance() : GUIEngine
+start() : void
+playGame() : void
+exitGame() : void
+newGame(levelNo : int) : void
+loadGame(String path) : void
+returnMenu() : void
+buttonClicked(position : Position) : void
+startGame(Player p) : void
+nextPage() : void
+gameIsOverLosingSituation() : void
+gameIsOverWinningSituation() : void
+loadNextLevel() : void
+returnMainMenuFromGameOverScreen() : void
+stillLoadGame() : void
+returnStartGUIFromLevelChoosen() : void
+returnMenuFromPlayerRequester() : void
+returnStartGUIFromLoadGUI() : void
+chooseLevel() : void
+swapSLokums(Position p1, Position p2) : void
GameEngine
-instance : GameEngine
-level : Level
-score : double
-movesLeft : int
-firstLokum : Lokum
-interactionEnabled : boolean
-player : Player
-sPosition : Position
-completedSwapCounter : int
-totalScore : double
-timeLeft : long
-specialSwapsLeft : int
+getInstance() : GameEngine
+reset(level) : void
+chooseLokum(position : Position) : void
+swapWith(position : Position) : void
+deselectLokum() : void
+decrementMoves(amount : int) : void
+enrollInInteraction() : void
+disableInteraction() : void
+createNewGame(level : Level) : void
+createLoadedGame(Player p, String path) : void
+updateScoreBy(amount : double) : void
+lokumClicked(position : Position) : void
+updateMovesLeftBy(amount : int) : void
+saveGame(String gamePath) : void
+saveSession() : void
+completeSwap() : void
+checkIsGame() : boolean
-gameLost() : void
-gameWon() : void
CAS
Player
-name : String
-map : HashMap
+getId() : int
+setId(int id) : void
+getName() : String
+setName(name : String) : void
XMLParser
+getDomElement(File xml) : Document
+getElement(Document node, String tagName) : String
+getValue(Element e, String str) : String
Goal
+goalReached() : boolean
Level
-levelNumber : int
-goal : Goal
-maxMoves : int
-xmlFile : File
-instance : Level
+getInstance() : Level
+hasLevel(int levelNo) : Level
+getTotalNumLevels() : int
ScoreGoal
-scoreGoal : double
+goalReached() : boolean
+toString() : String
TimeScoreGoal
-timeGoal : long
+goalReached() : boolean
+toString() : String
Board
-instance : Board
-boardMatrix : SquareOccupier[][]
-ROW_NUMBER : int
-COLUMN_NUMBER : int
-currentGenerationForRow : int
-currentGenerationForColumn : int
-findMatchCheckList : ArrayList
-currentLastGenerated : SLokum
-previousTotalCounter : int
+getInstance() : Board
+newBoard() : void
+loadBoard(loadedMatrix : SquareOccupier [][]) : void
+reset() : void
+fastSwap(position : Position, position2 : Position) : boolean
+fillLokumsToEmptySquares() : void()
+generateLokumsToEmptySquaresInAColumn(int column) : void
+generateLokumsToEmptySquares() : void
+findAllMatches() : void
+getLokumAt(Position p) : Lokum
+setLokum(Lokum l, Position p) : void
+toString() : String
+removeLokumAt(Position p) : void
+getRepresentationMatrix() : SquareOccupierDescription [][]
+explodeLastGenerated(SLokum s) : void
Position
-x : int
-y : int
+equals(Position p) : boolean
+isWith(Position p) : boolean
AudioPlayer
+playBackgroundMusic() : void
ConfigurationXML
+populateFile xml, File xml) : boolean
GameLoader
-player : Player
-occupierMatrix : SquareOccupier[][]
-score : double
-movesLeft : int
-level : Level
-gameXSDFile : File
-instance : GameLoader
+getInstance() : GameLoader
+loadGame(Player p, String gamePath) : void
+putToOccupierMatrix(SquareOccupier [][] SquareOccupier so) : void
+isPlayerLoadable(Player p) : boolean
+loadableXML s(Player p) : HashMap
-loadPlayer(Document d) : Player
-loadBoard(Document d) : SquareOccupier [][]
-loadScore(Document d) : double
-loadMovesLeft(Document d) : int
-loadLevel(Document d) : Level
SessionLoader
-maxLevel : Level
-score : double
-player : Player
-sessionXSDFile : File
-instance : SessionLoader
+getInstance() : SessionLoader
+hasSession(Player p) : void
+isPlayerLoadable(Player p) : boolean
-loadPlayer(Document d) : Player
-loadScore(Document d) : double
-loadMaxLevel(Document d) : Level
SessionSaver
+saveSession(Player p) : void
GameSaver
+saveCurrentGame(String gamePath) : void
Handlers
SubscriptionKeeper
-swapListenersList : ArrayList<SwapListeners>
-matchListenersList : ArrayList<MatchListeners>
-explodeListenersList : ArrayList<ExplodeListeners>
-instance : SubscriptionKeeper
-comboCount : int
-xmlFile : File
+getInstance() : SubscriptionKeeper
+swap(lokum1 : Lokum , lokum2 : Lokum) : void
+explode(lokumList : ArrayList) : void
+findMatch(Lokum lokum) : void
+findMatches(Lokum lokum1, Lokum lokum2) : void
-loadListeners() : void
<<Interface>>
SwapListener
+swap(swapEvent : SwapEvent) : void
<<Interface>>
MatchListener
+match(matchEvent MatchEvent) :void
<<Interface>>
ExplodeListener
+explode(explodeEvent ExplodeEvent) :void
<<Interface>>
GenerateListener
+generate(generateEvent : GenerateEvent) :void
SwapEvent
-lokum1 : Lokum
-lokum2 : Lokum
GenerateEvent
-color : String
-position : Position
-detail : String
MovesLeftUpdater
WrappedStripedHandler
WrappedWrappedHandler
StripedStripedHandler
ColorBombWrappedHandler
ColorBombColorBombHandler
ColorBombStripedHandler
ColorBombRegularHandler
ColorBombGenerator
WrappedGenerator
StripedGenerator
WrappedExploder
-explodeListFilter(Position pos, ArrayList list) : void
ColorBombExploder
-explodeListFilter(String color, ArrayList list) : void
-chooseRandomExplodeColor(Position pos)" String
StripedExploder
RegularLokumExploder
TimeLokumExploder
Match4TDetector
-colisaMatch(Lokum lokum) : ArrayList
Match5TDetector
-colisaMatch(Lokum lokum) : ArrayList
Match5SDetector
-verticalMatch(Lokum lokum) : ArrayList
-horizontalMatch(Lokum lokum) : ArrayList
Match3SDetector
-verticalMatch(Lokum lokum) : ArrayList
-horizontalMatch(Lokum lokum) : ArrayList
Match3SDetector
-matched : boolean
-horizontalMatch(Lokum lokum, SquareOccupier [][]) : ArrayList
-verticalMatch(Lokum lokum, SquareOccupier [][]) : ArrayList
ExplodeEvent
-lokum : Lokum
-comboCount : int
MatchEvent
-lokum : Lokum
Occupiers
SquareOccupier
-position : Position
-description : SquareOccupierDescription
+isLokum() : boolean
Lokum
+getColor() :Color
+setColor(String color) : void
+isSpecialLokum() : boolean
SquareOccupierDescription
-type : String
+getRepresentation() : Representation
+getType() : String
+setType(type : String) : void
SpecialLokum
RegularLokum
LokumDescription
-color : String
ColorBombLokum
TimeLokum
-bonusTime : long
WrappedLokum
StripedLokum
-direction : String
+getDirection() : String
+setDirection(direction : String) : void
SquareOccupierFactory
-possibleColors : ArrayList
-instance : SquareOccupierFactory
+getInstance() : SquareOccupierFactory
+generateRandomRegularLokum() : Lokum
+generateRandomRegularLokum(Position p) : Lokum
+generateLokum(String color, String type) : Lokum

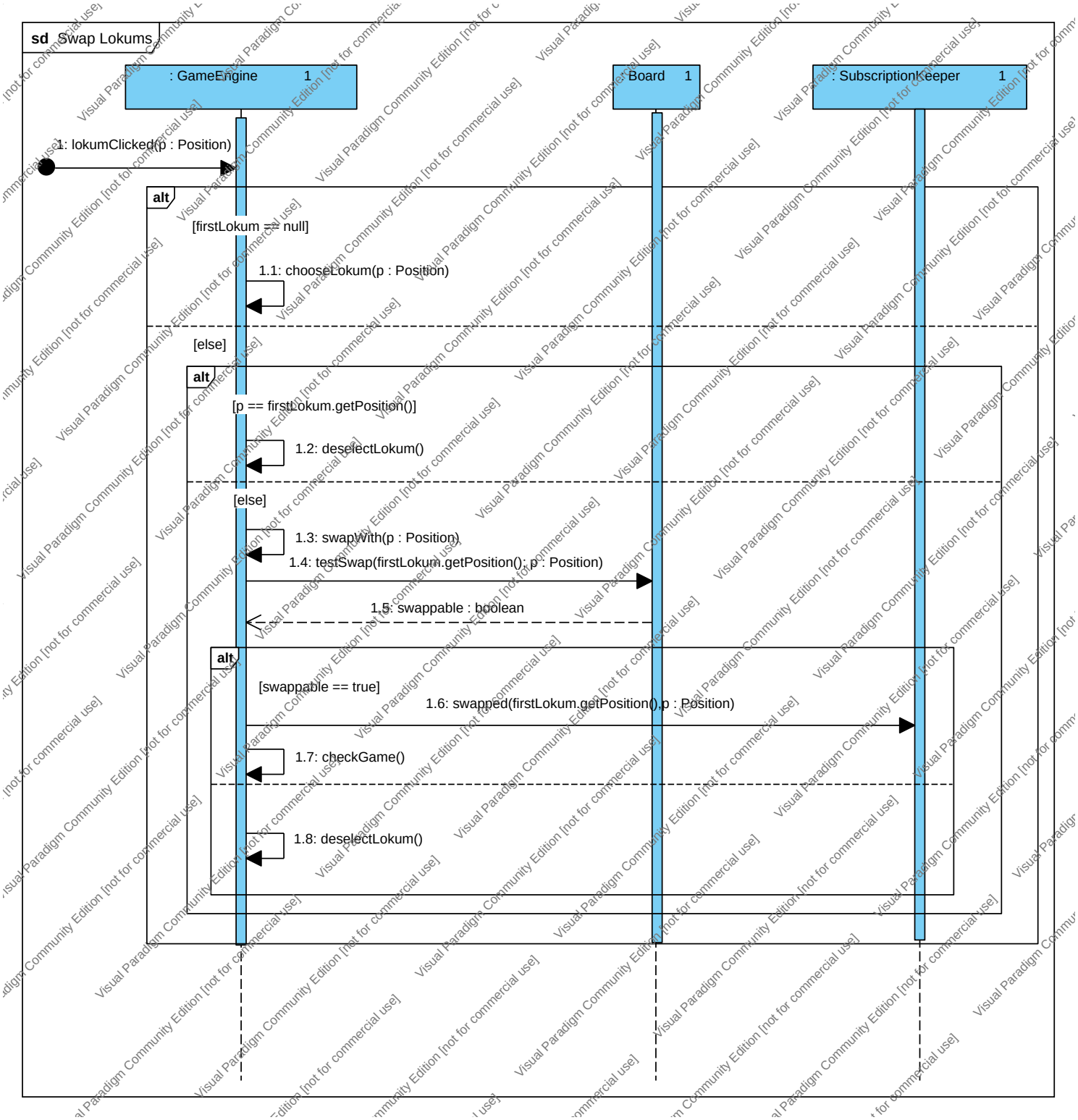

sd load Game
: GUIEngine
: GameEngne 1
: GameLoader 1
: Board 1
1: loadGame(path : String)
1.1: createLoadedGame(path : String)
1.1.1: loadGame(player,path)
1.1.2: loadBoard(loadedMatrix)
1.1.3: enableInteraction()

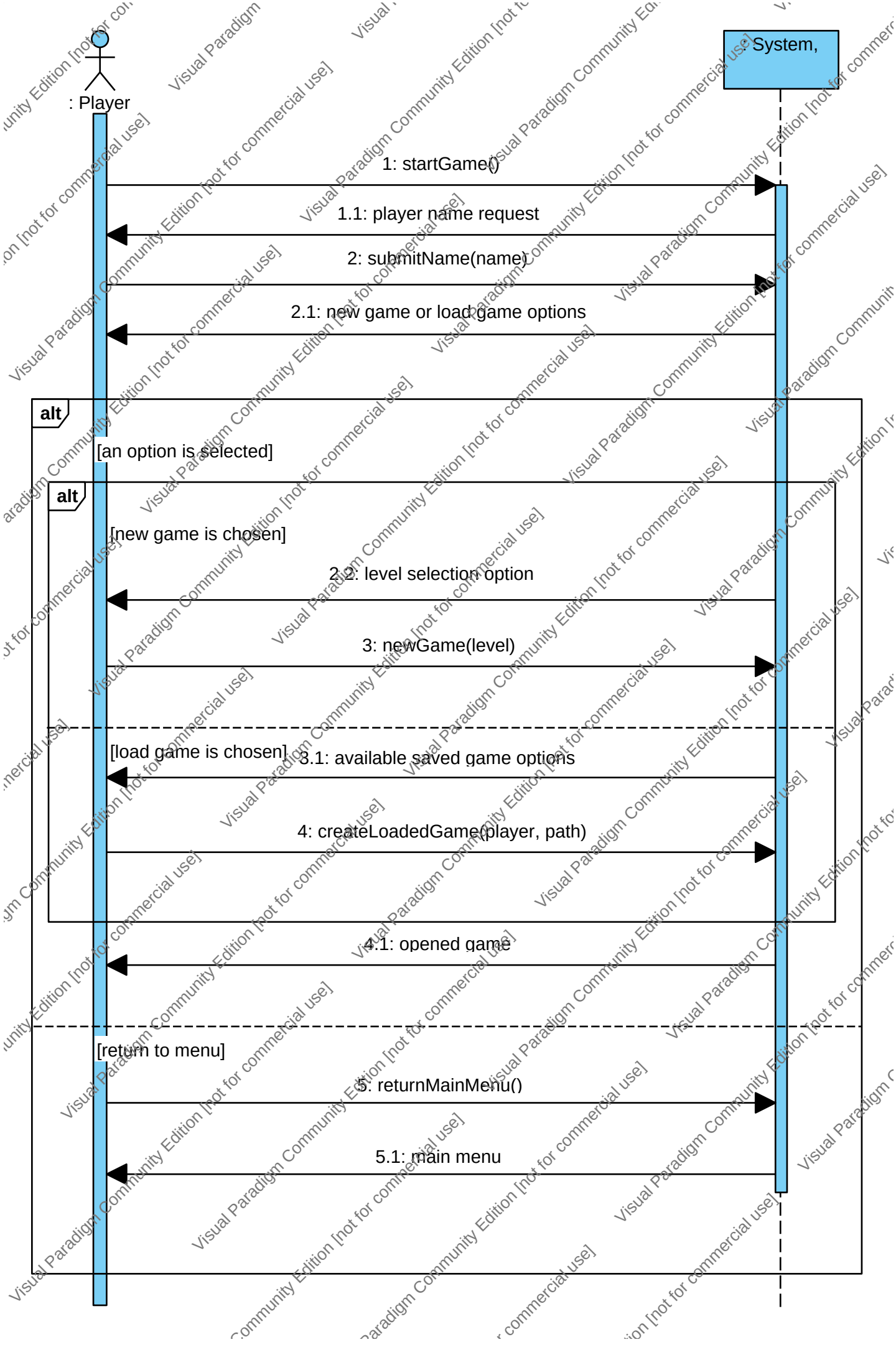

sd OpenGame
: Controller 1
: MenuGUI
1: start()
1.1: create()

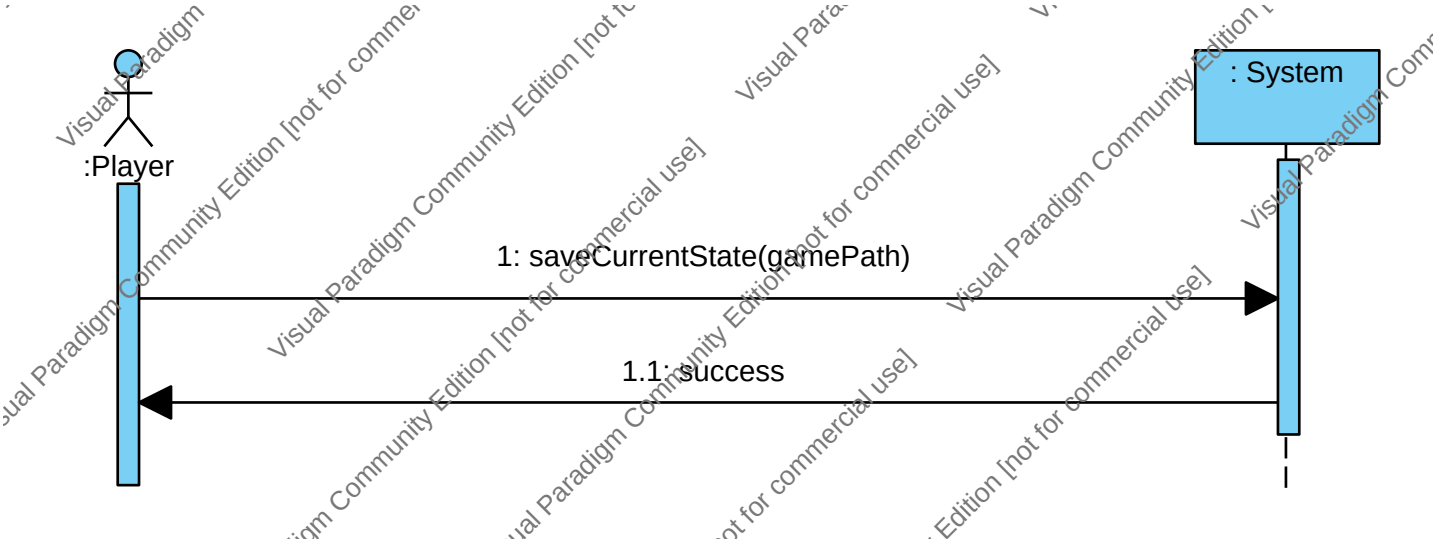

sd Play New Game
: Controller 1
: GameEngine 1
: Board 1
1: newGame(level)
1.1: createNewGame(level)
1.1.1: create()
1.1.2: newBoard(level)

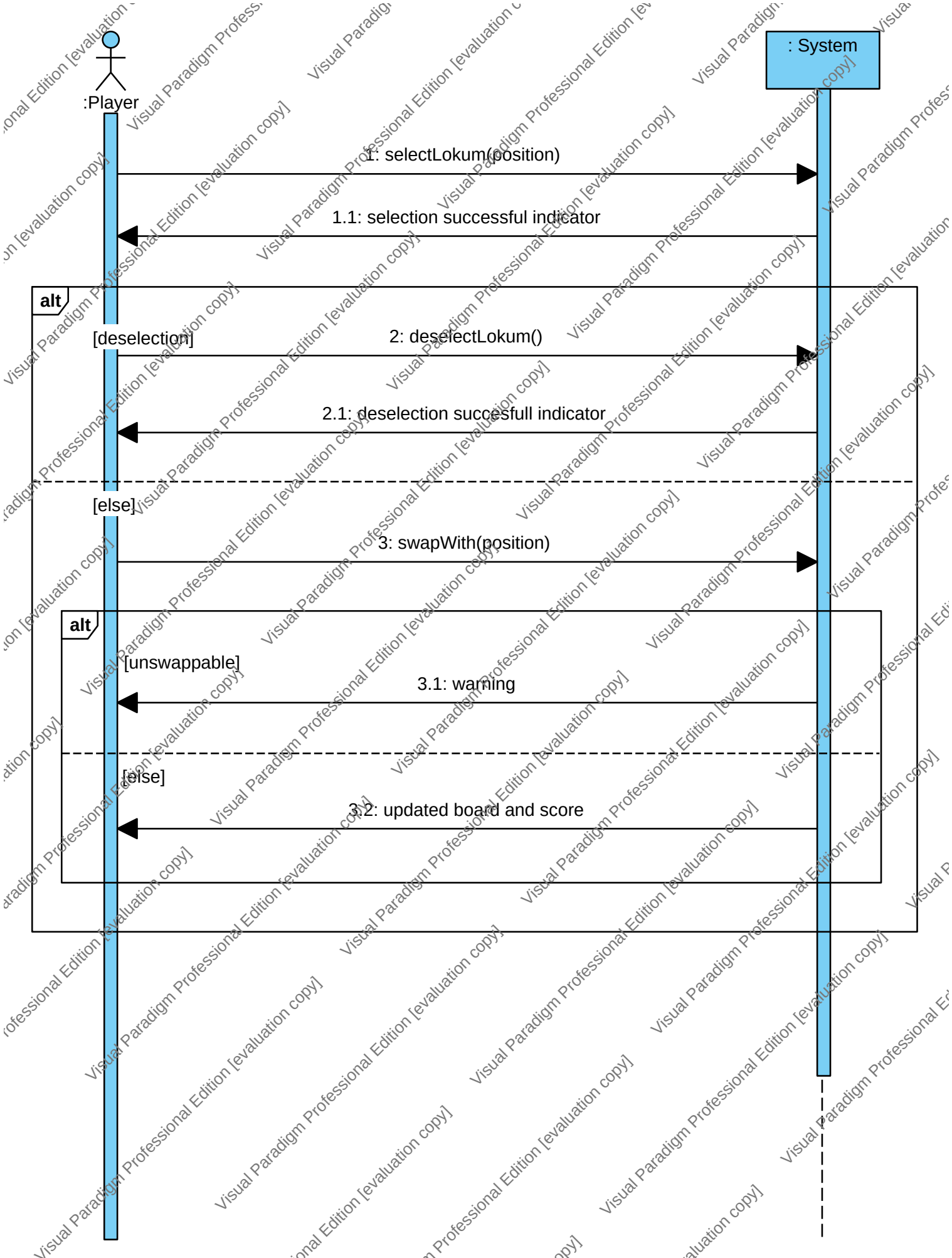

sd PlayGame
: Controller 1
: MenuGUI
: startGameGUI
1: playGame()
1.1: dispose()
1.2: create(level)

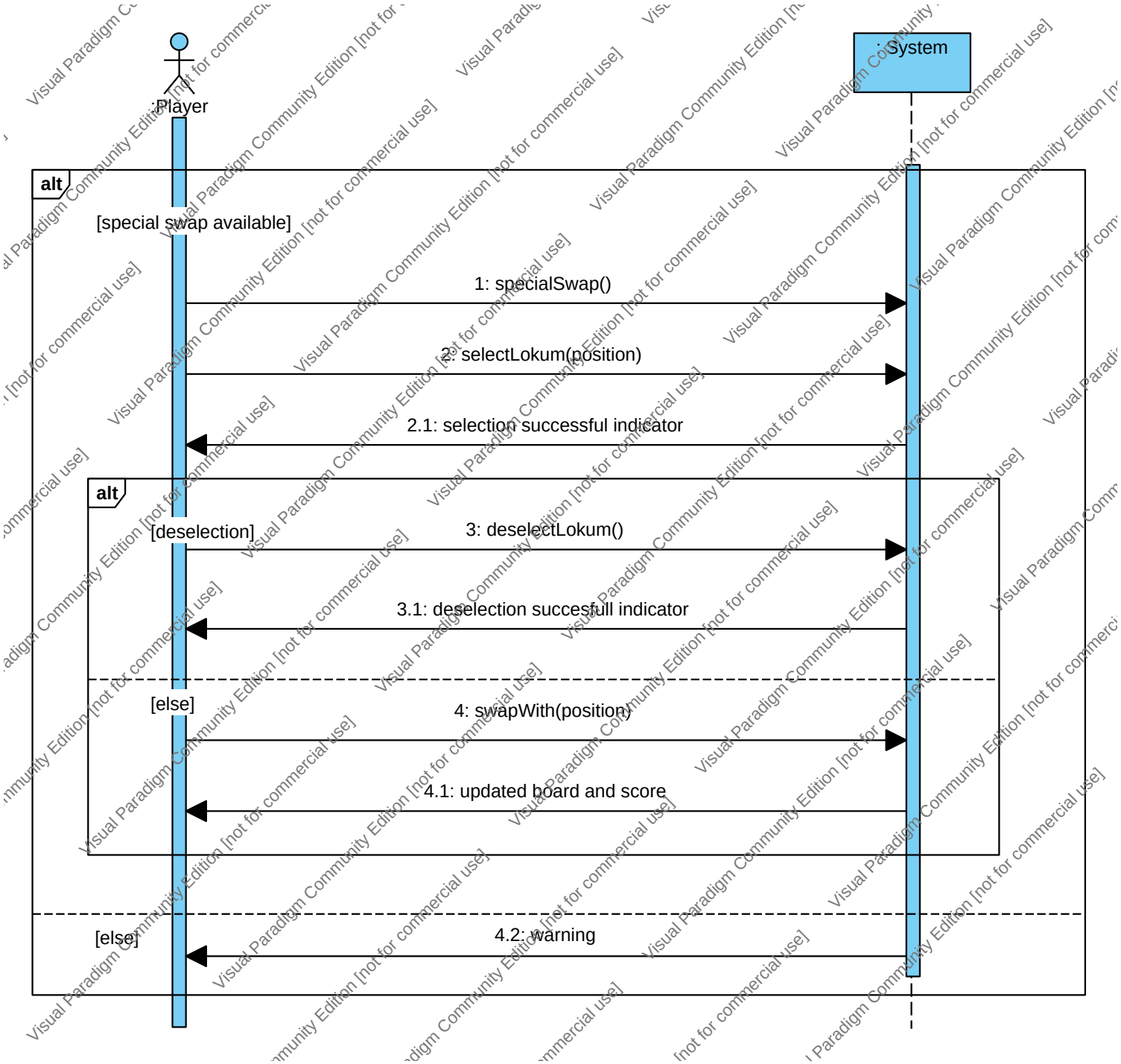

sd SaveGame
GUIEngine 1
GameEngine 1
GameSaver 1
1: saveGame(path : String)
1.1: saveGame(path:String)
1.1.1: saveCurrentGame(path:String)

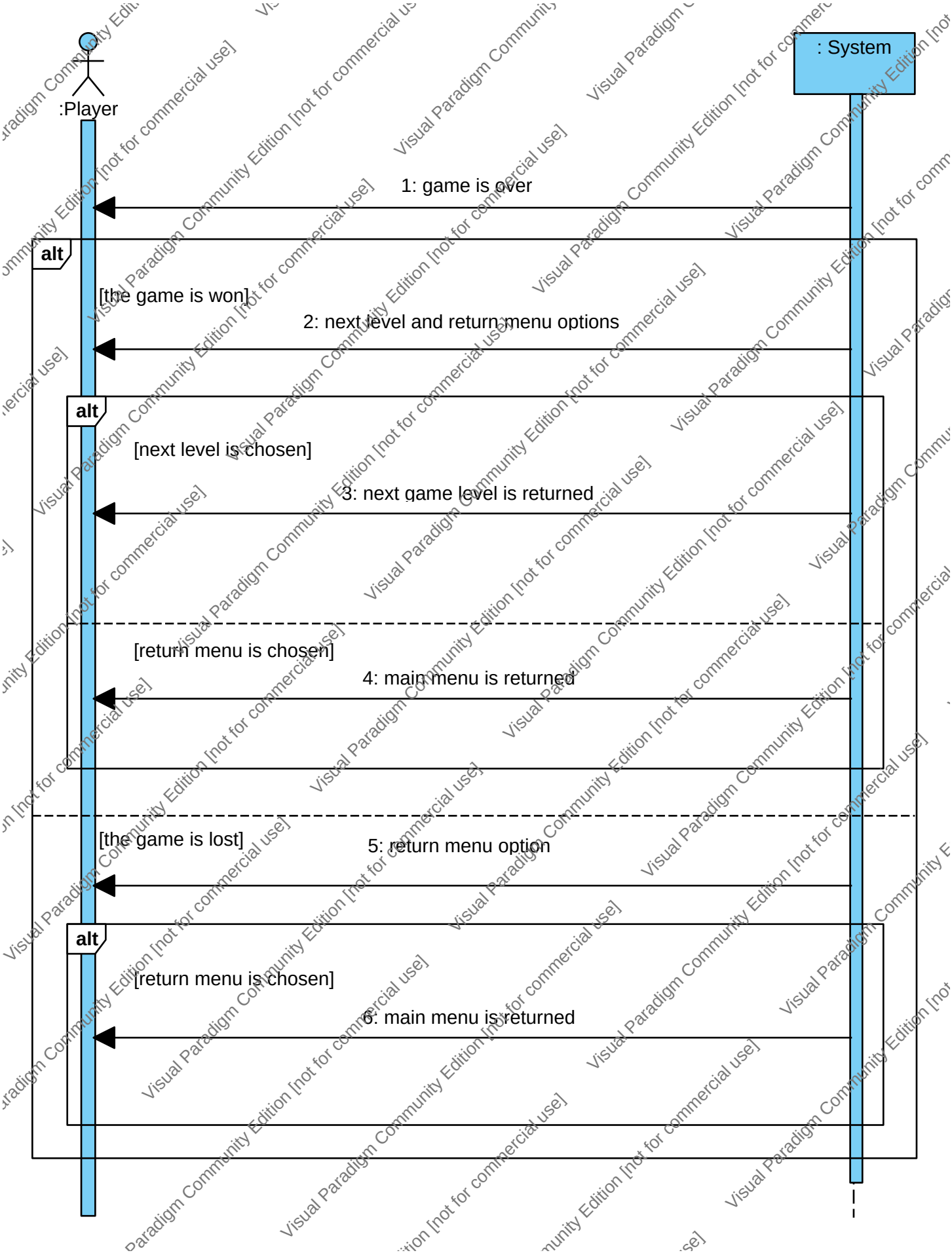

sd Swap Lokums
: GameEngine 1
Board 1
: SubscriptionKeeper 1
1: lokumClicked(p : Position)
alt
[firstLokum == null]
1.1: chooseLokum(p : Position)
[else]
alt
[p == firstLokum.getPosition()]
1.2: deselectLokum()
[else]
1.3: swapWith(p : Position)
1.4: testSwap(firstLokum.getPosition(), p : Position)
1.5: swappable : boolean
alt
[swappable == true]
1.6: swapped(firstLokum.getPosition(),p : Position)
1.7: checkGame()
1.8: deselectLokum()


: Player
System,
1: startGame()
1.1: player name request
2: submitName(name)
2.1: new game or load game options
alt
[an option is selected]
alt
[new game is chosen]
2.2: level selection option
3: newGame(level)
[load game is chosen]
3.1: available saved game options
4: createLoadedGame(player, path)
4.1: opened game
[return to menu]
5: returnMainMenu()
5.1: main menu


:Player
: System
1: saveCurrentState(gamePath)
1.1: success


:Player
: System
1: selectLokum(position)
1.1: selection successful indicator
alt
[deselection]
2: deselectLokum()
2.1: deselection succesfull indicator
[else]
3: swapWith(position)
alt
[unswappable]
3.1: warning
[else]
3.2: updated board and score


:Player
: System
alt
[special swap available]
1: specialSwap()
2: selectLokum(position)
2.1: selection successful indicator
alt
[deselection]
3: deselectLokum()
3.1: deselection succesfull indicator
[else]
4: swapWith(position)
4.1: updated board and score
[else]
4.2: warning


:Player
: System
1: game is over
alt
[the game is won]
2: next level and return menu options
alt
[next level is chosen]
3: next game level is returned
[return menu is chosen]
4: main menu is returned
[the game is lost]
5: return menu option
alt
[return menu is chosen]
6: main menu is returned

Contract CO1: newGame

Operation: newGame(level: Level)

Cross References: Use Cases: UC1 – StartGame

Preconditions:

- The start game menu must be initialized.
- A level is selected.

Postconditions:

- A new game of the given level is initialized and shown.
- The saved game is overwritten by the current game.

Contract CO2: selectLokum

Operation: selectLokum(position: Position)

Cross References: Use Cases: UC2 – SwapLokums

Preconditions:

- The game must be initialized.
- The first Lokum lokum1 has not been set yet.

Postconditions:

- lokum1 is set to the Lokum whose position is given.
- An indication for selected Lokum is shown to user.

Contract CO3: deselectLokum

Operation: deselectLokum()

Cross References: Use Cases: UC2 – SwapLokums

Preconditions:

- The game must be initialized.
- The first Lokum lokum1 must be set with selectLokum(position: Position) already.

Postconditions:

- lokum1 as the first chosen Lokum is unset.
- The indication for lokum1 is removed.

Contract CO4: swapWith

Operation: swapWith(position: Position)

Cross References: Use Cases: UC2 – SwapLokums

Preconditions:

- The game must be initialized.
- The first Lokum lokum1 must be chosen with selectLokum(position: Position) already.

Postconditions:

- If lokum1 and the Lokum at the given position are consecutive and swap forms a triplet, a quatruplet or a quintuplet:
    - lokum1 and the Lokum at the given position are swapped.
    - Game.movesLeft is decremented by one.
    - lokum1 as the first chosen Lokum is unset.
- If lokum1 and the Lokum at the given position are not consecutive or swap does not form a triplet, a quatruplet or a quintuplet:
    - A warning indicating two Lokums are unswappable is shown.
    - lokum1 as the first chosen Lokum is unset.

Contract CO5: testSwap

Operation: testSwap (position1: Position, position2: Position)

Cross References: Use Cases: UC2 – SwapLokums

UC3 – SpecialSwapLokums

Preconditions:

- The game must be initialized.
- The first and the second Lokums lokum1 and lokum2 must be chosen already.

Postconditions:

- If lokum1 and lokum2 are valid and consecutive and the swap of those Lokums forms at least one triplet:
    - Returns true
- else
    - Returns false

Contract CO6: saveCurrentState

Operation: saveCurrentState (gamePath : String)

Cross References: Use Cases: UC4 – SaveGame

Preconditions:

- The game must be initialized.

Postconditions:

- In the Player's folder, a new XML file containing the current state of the game named the given string gamePath is created.

Contract CO7: createLoadedGame

Operation: createLoadedGame (player : Player, path : String)

Cross References: Use Cases: UC1 – StartGame

Preconditions:

- The XML file named path.xml in the folder named player must exist.

Postconditions:

- A new game with the saved board state, the saved score, the saved level number, the saved number of moves left is initialized.