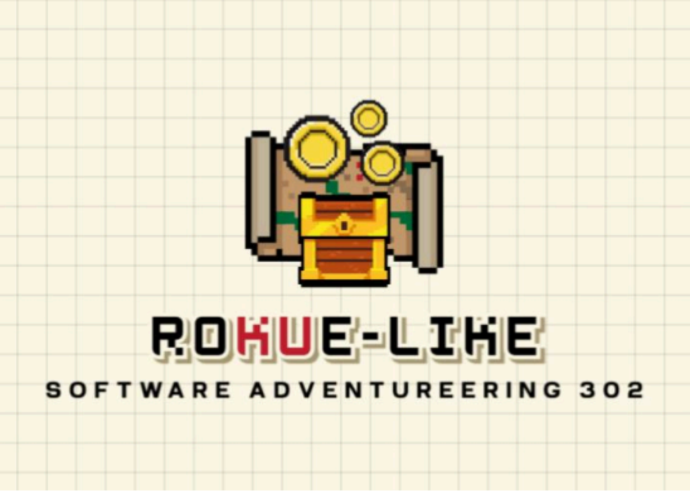


# COMP302 SOFTWARE ENGINEERING

*Fall 2024 Term Project*
*Final Report*

Group 9

Abdullah Yusuf Yıldırım
Ahmet Umut Akduman
Halil İbrahim Kanpak
İrfan Can Beşer
Mehmet Kantar
Utku Çubukçu
Toygar İlhan Bozkurt

KOÇ ÜNİVERSİTESİ
COMPUTER ENGINEERING

# Table of Contents

# Vision

## Introduction

We envision an easy to understand user-controlled game, with a game mode that allows the player to customize their game experience, called building mode.

## Positioning

### *Business Opportunity*

Rogue like games have been successful over the years with examples such as The binding of Isaac, Hades or Enter the Gungeon. These games present an easy to understand but hard to master game mechanics which captivates players. With the addition of build mode which allows user expression, RoKUlike game possesses an incredible potential.

### *Problem Statement*

The rogue-like genre has been defined by the random event system that appears in many parts of the game, especially during the level creation system. Although this presents an unpredictability aspect to the game which can make the game more exciting, it can also frustrate players with unlucky level design such as having many hard levels in proximity. Our game gives the opportunity to create levels to the players, so every player can fine-tune their desired level of difficulty, which makes frustrating game experiences less likely to happen. This system also has the additional benefit of encouraging player creativity.

### *Product Position Statement*

The system is for creative people who are not used to playing Rogue like games before. Our game aims to give an easy-to-understand set of mechanics that allows people to express their creative side, which allows the users to have fun and boost their creative side.

## Stakeholder Descriptions

The players are mostly people who are familiar with the general idea of video games and dream of making a game someday, but may or not be familiar with the rogue-like genre specifically. These people are usually creative people who are interested in designing and playing their own games.

## *Key High-Level Goals and Problems of the Stakeholders*

| High-Level Goal | Priority | Problems and concerns | Current Solutions |
|---|---|---|---|
| A smooth to navigate and easy to understand user interface, which allows players to express themselves freely | High | For making the application smooth to operate, there needs to be universally recognized game icons such as save icon or the play game button. There can also be tutorials in the title screen of the game, but this can overwhelm the user and make them less likely to play the game.<br>Similarly, to allow players to express themselves according to the rules of the game, design of the game mechanics and game rules should be intuitive and easy to understand. | Most rogue-like games assume a familiarity with the game genre, so their tutorials can be confusing and not helpful to the player. A commonly used tactic is using web wikis to detail the various aspects of the game, which can overwhelm the user and disrupt the game flow. |

## *User-Level Goals*

The player is mainly trying to fulfill their creative side by building the game levels and experiencing an easy and fun challenge during the gameplay parts of the game.

# Teamwork

Since the process of announcing the project and forming the teams took a long time, we had less time to complete the project. For this reason, from the moment the project was announced, we created a roadmap for ourselves and distributed tasks, just as we had done in each previous stage. During the development phase of the project, we started with small drafts from the very beginning, reducing our iteration times, and continued to refine the work until we achieved the final product. In terms of task distribution, we divided into two teams for Phase 1: a 4-person team focused on Play Mode and another team responsible for Build Mode and other menus. In Phase 2, each member contributed by adding improvable features, and with good communication, we reached the final product.

**Phase 1:**

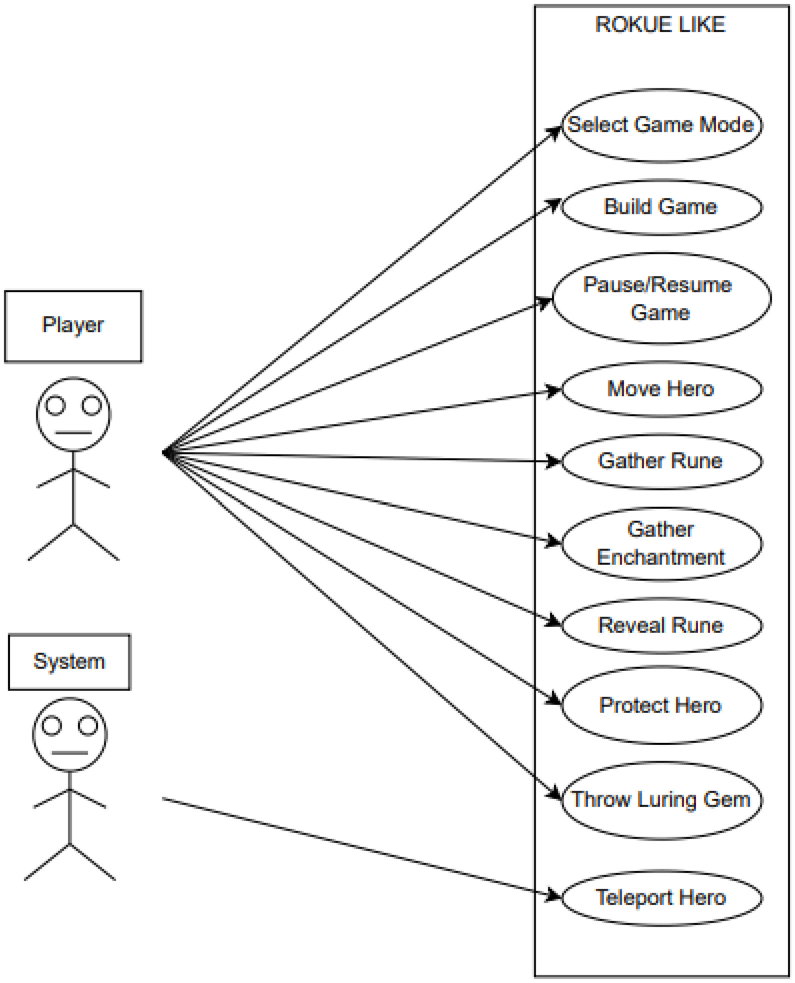
- **Play Mode:** Abdullah, Ahmet, Utku, Toygar
- **Build Mode and others:** Halil, İrfan, Mehmet

**Phase 2:**

- **Wizard:** Toygar
- **Save/Load:** Ahmet, Halil
- **Play Mode Side Screen:** İrfan, Utku
- **UI Updates and Extra Features:** Abdullah, Mehmet

For the Phase 1 and 2 documentation, detailed information can be found in the weekly agendas. As a team, we completed the documentation with an equal distribution of tasks.

# Use Case Diagram



# Use Case Narratives

**Use Case Name:** Move Hero

**Use Case ID:** UC-1

**Scope:** ROKUE LIKE-Phase 1

**Level:** User Goal

**Primary Actor:** Player

**Stakeholders and Interests:**
-Player: Wants to be able to move the hero in the chosen direction on the grid without delay, as even a millisecond of latency can result in monster damage.
-Developer: Wants a bug-free game that offers a decent user experience.

**Preconditions:**
- The game is in play mode.

**Postconditions:**
-The hero was moved east/west/north/south in response to user keyboard commands.

**Main Success Scenario:**

1. The player presses the arrow keys that allow the Hero to move in the desired direction within a hall.
2. The hero goes east/west/north/south without delay based on the key tapped by the user.
3. Hero stays at its new location.
4. Hero can still move inside the hall's established boundaries.

**Extensions:**

*a. At any time, the System fails.

1. The player restarts the Game.

2a. The player reaches the visible boundaries of the hall.

1. Hero stops at its current location and does not move out of hall.

**Technology and Data Variations List:**
1a. The player uses the keyboard to move the Hero.

**Frequency of Occurrence:**
-Almost every time interval. The player will almost always move the hero to find the rune, collect new enchantments or escape from monsters except when the game is paused.

**Use Case Name:** Gather Enchantment

**Use Case ID:** UC-2

**Scope:** ROKUE LIKE-Phase 1

**Level:** User-goal

**Primary Actor:** Player

**Stakeholders and Interests:**
-Player: Wants to gather enchantments that occur across the map. Specifically interested in gathering extra time, hints about the location of rune, luring gems, cloak of protection, and extra lives.
-Developer: Wants the enchantments to spawn across random and available points in the map.

**Preconditions:**

1. The game is in play mode.

**Postconditions:**

1. The enchantments have been manipulated for players' decisions (used or sent to inventory).
2. Hero's inventory has been updated based on gathered enhancements.

**Main Success Scenario:**

1. The player left-clicks on enchantments that show up on the map.
2. The player decides between using the enchantment immediately or storing it in his/her inventory.
3. The number of the enchantment in the hero's inventory is increased by 1 if it is decided to be stored.

**Extensions:**
*a. At any time, the System fails.

1. The player restarts the game.

2a. In case the player inventory is full, the system returns the message “inventory is full”.

**Special Requirements:**
The player uses certain mouse interactions (left click) to gather up enchantments that are randomly located on the grid.
Extra time and Extra Life enchantments cannot be stored in the inventory.
Some methods and data structures to account items in the hero's inventory.

**Technology and Data Variations List:**
The player uses the mouse to execute it.

**Frequency of Occurrence:** A random type of Enhancement appears every 12 seconds.

**Use Case Name:** Gather Rune

**Use Case ID:** UC-3

**Scope:** ROKUE LIKE-Phase 1

**Level:** User-goal

**Primary Actor:** Player

**Stakeholders and Interests:**
-Player: Wants to find the object that hides the rune.
-Developer: Wants the rune to spawn inside chest objects and manipulate them according to the game rules.

**Preconditions:**

1. The game is in play mode.
2. Hero is within the range object, i.e. the Hero is located next to the object.

**Postconditions:**

1. The items inside the chest have been manipulated for players' decisions.
2. The chest has been closed.

**Main Success Scenario:**

1. The player triggers the objects.
2. Rune shows up.
3. Rune is obtained by the player.

**Extensions:**
*a. At any time, the System fails.

1. The player restarts the game.

1a. The hero is not within the range of any objects

1. Nothing happens after the player tries to interact with a chest.

1b. The hero is equidistant to two chests

1. The chest closest to the bottom left (with being in the bottom primed) is selected as the chest to be interacted with.

**Special Requirements:**
The player uses a certain keyboard action to move towards objects. Also uses mouse interactions to gather runes.
Some methods and data structures to account across all inventories.

**Technology and Data Variations List:**
The player uses the mouse and keyboard to execute it.

**Frequency of Occurrence:** Frequently occurring throughout the game as it is the main game mechanic until the game ends and finding the rune is the main purpose of the game.

**Use Case Name:** Pause/Resume Game

**Use Case ID:** UC-4

**Scope:** ROKUE LIKE-Phase 1

**Level:** User-Goal

**Primary Actor:** Player

**Stakeholders and Interests:**
Player: Wants to pause the game and then successfully continue from the paused moment

**Preconditions:**

1. The game is in playing mode.

**Postconditions:**

1. The game was paused/resumed.

**Main Success Scenario:**

1. The player starts the game and it's in running mode
2. The player presses a button to pause the game
3. Game is paused.
4. Pause button is replaced with the resume button.
5. The player presses a button to resume the game.

**Extensions:**
*a. At any time, the System fails.

1. The player restarts the game

2a. After pressing the resume button the game is still in pause mode

1. The player restarts the game

2b. After pressing the pause button the game is still running

1. The player restarts the game

3a. The player presses the desired button while the game is paused.

**Technology and Data Variations List:**
1a. The player uses the mouse to pause/resume the game.

**Frequency of Occurrence:**

Depends on the player; as long as the system is operational, the player can halt and continue.

**Start Game:** The actor is the player, who starts the game. The game begins with Building Mode.

**Build game:** Where the player is the actor. The user can adjust the game difficulty, and the fall speed of objects changes accordingly. The player can define the amount of game objects, unit L, and the shape of certain molecules.

**Run Game:** The actor is the player, and the player starts the game in Running Mode. It ends the Building mode. Running mode continues till the finish of the game.

**Use Case Name:** Teleport Hero

**Use Case ID:** UC-5

**Scope:** ROKUE LIKE-Phase 1

**Level:** User-Goal

**Primary Actor:** System

**Stakeholders and Interests:**
The system wants the hero to move on to the next hall.

**Preconditions:**

1. The game is in running mode.
2. The player successfully opened the door of the hall and was sufficiently close to the door.

**Postconditions:**

1. Hall is updated accordingly to the next stage.

**Main Success Scenario:**

1. The hero comes next to the door of the hall.
2. The hero opens the door.
3. The hero is teleported to a random location on the map of the next hall.
4. The map and hall are updated accordingly to the next stage.

**Extensions:**
*a. At any time, the System fails.

1. The player restarts the game

1a. The player has not opened the door with the rune yet

1. The system returns a warning, indicating the door is not opened yet if the player is close to a door that is not open.

**Frequency of Occurrence:**
Happens once per every hall after the player acquires the rune, opens the door, and reaches for it. If a game is successfully completed, then the hero is teleported three times.

**Use Case Name:** Reveal Rune

**Use Case ID:** UC-6

**Scope:** ROKUE LIKE-Phase 1

**Level:** User Goal

**Primary Actor:** Player

**Stakeholders and Interests:**
-Player: Wants to cast reveal enchantment.

**Preconditions:**
-The game is in running mode.
-The hero has at least one reveal enchantment in his/her inventory.

**Postconditions:**
-The number of reveal enchantment has decreased in the hero's inventory.

**Main Success Scenario:**

1. The player presses the **R** button on the keyboard.
2. A 4x4 square containing the rune is displayed as a hint to the player.
3. The system decreases the number of "reveal" enchantments in the hero's inventory by 1.

**Extensions:**
*a. At any time, the System fails.

1. The player restarts the game

2a. There isn't any "reveal" left in the player inventory.

1. The system returns a warning, indicating there is no "reveal" enchantment remaining in the hero's inventory.

**Technology and Data Variations List:**
1a. The player uses the keyboard to use enhancement **Reveal**.

**Frequency of Occurence:**
The action can be done as long as there is at least one "reveal" item left in inventory.

**Use Case Name:** Protect Hero

**Use Case ID:** UC-7

**Scope:** ROKUE LIKE-Phase 1

**Level:** User Goal

**Primary Actor:** Player

**Stakeholders and Interests:**
-Player: Wants to equip the hero with the cloak of protection to hide the hero from archer monster

**Preconditions:**
-The game is in running mode.
-The player has at least one cloak of protection in his/her inventory.

**Postconditions:**
-The number of the cloak of protection enchantment has decreased in the player's inventory.

**Main Success Scenario:**

1. The player presses the **P** button on the keyboard.
2. The player equips the hero with the cloak of protection to make the hero invisible for archer monster until the determined time ends for the cloak of protection
3. The system decreases the number of cloak of protection enchantments in the player's inventory by 1.

**Extensions:**
*a. At any time, the System fails.

2. The player restarts the game

2a. There isn't any cloak of protection left in player inventory.

2. The system returns a warning, indicating there is no cloak of protection remaining in the player's inventory.

3a. The player is already equipped with one cloak of protection.

1. The remaining time of the cloak of protection resets to the initial time determined for the cloak of protection.

**Technology and Data Variations List:**
1a. The player uses the keyboard to use the enchantment **Cloak of Protection**.

**Frequency of Occurence:**
The action can be done as long as there is at least one cloak of protection left in inventory.

**Use Case Name:** Throw Luring Gem

**Use Case ID:** UC-8

**Scope:** ROKUE LIKE-Phase 1

**Level:** User Goal

**Primary Actor:** Player

**Stakeholders and Interests:**
-Player: Wants to throw a luring gem to fool the fighter monster

**Preconditions:**
-The game is in running mode.
-The player has at least one luring gem in his/her inventory.

**Postconditions:**
The number of luring gem enchantment has decreased in the player's inventory.

**Main Success Scenario:**

1. The player presses the **B** button on the keyboard and then one of the **W A S D** buttons to select the direction to throw.
2. The player throws the luring gem in the selected direction.
3. The system decreases the number of luring gem enchantments in the player's inventory by 1.

**Extensions:**
*a. At any time, the System fails.

1. The player restarts the game

2a. There isn't any luring gem left in the player inventory.

3. The system returns a warning, indicating there are no luring gems left in the player's inventory.

3a. The player is moving while throwing the luring gem.

2. The luring gem has been thrown as if the player was not moving at that location (the momentum of the player does not add up to the gem momentum)

**Technology and Data Variations List:**
1a. The player uses the keyboard to use enhancement **Luring Gem**.

**Frequency of Occurence:**
The action can be done as long as there is at least one luring gem left in inventory.

**Use Case Name:** Select Game Mode

**Use Case ID:** UC-9

**Scope:** ROKUE LIKE-Phase 1

**Level:** User-Goal

**Primary Actor:** Player

**Stakeholders and Interests:**
-Player: Wants to start the game by selecting either Build Mode or Play Mode. The player can use saved maps in previously designed build mode.

**Preconditions:**
-The game must be launched, and the player must be at the Main Menu screen.

**Postconditions:**
-The game mode is successfully initialized
-The player starts in either Build or Play Mode.

**Main Success Scenario:**

1. Game is initialized.
2. Player clicks a button to switch to the **Build Mode** screen.
3. The system transitions to **Build Mode**, where the player designs the inside of the halls.
4. The system transitions to **Play Mode**, where the player uploads the design of the worlds.
5. The system transitions to **Help**, and the player is informed about game mechanisms.

**Extensions:**
*a. At any time, the System fails.

1. The player restarts the Game.

1a. The player does not meet the minimum criteria for placing objects in the halls during Build Mode.

1. The system gives a warning indicating that the hall does not meet the minimum requirements.
    a. There must be at least 6 objects in the earth hall.
    b. There must be at least 9 objects in the air hall.
    c. There must be at least 13 objects in the water hall.
    d. There must be at least 17 objects in the fire hall.

2a. There are no saved Halls for play mode.

1. The player goes back to the Main Menu and select Build Mode.

**Technology and Data Variations List:**
The player uses a mouse click to select mode.

**Frequency of Occurrence:**
Selecting a game mode occurs at the start of every new game session.
If the player restarts the game after exiting or losing, they will need to select the game mode again.

**Use Case Name:** Build Game

**Use Case ID:** UC-10

**Scope:** ROKUE LIKE-Phase 1

**Level:** User-Goal

**Primary Actor:** Player

**Stakeholders and Interests:**

-Player: Wants to build and design the inside of each hall in the dungeon to create a challenging and interesting play area.

**Preconditions:**

1. The player has clicked **Build Mode** from the Main Menu.
2. The game transitioned to **Build Mode**.

**Postconditions:**

1. Number of objects populated is based on player's object placement choice along with the minimum number of object criteria..
2. The player proceeds to **Play Mode**.

**Main Success Scenario:**

1. The system presents the Hall of Earth as the first level to design.
2. The player selects and places various objects in the hall.
3. The player must meet the minimum requirement of objects for each hall:
4. At least 6 objects in the Hall of Earth.
5. The player completes the Hall of Earth and proceeds to the next halls:
6. Hall of Air (requires at least 9 objects),
7. Hall of Water (requires at least 13 objects),
8. Hall of Fire (requires at least 17 objects).
9. Once all halls are built, the player clicks Finish Build, and the system transitions to Play Mode.

**Extensions:**

*a. At any time, the System fails.

1. The player restarts the game

3-5a. The player does not place the required minimum number of objects in a hall.

1. System gives warning
2. The player continues placing objects until the minimum requirement is met.

3-5b. The player does not place the required minimum number of objects in a hall.

1. The player selects the object and clicks remove.

2. The system updates the hall accordingly.

3-5c. The player decides to exit Build Mode before completing all halls.

1. The player clicks the **Exit** button.
2. System prompts with a confirmation message: "Are you sure you want to exit? Unsaved progress will be lost."

**Technology and Data Variations List**:

1) The player uses a mouse to select and drag objects to their desired position within the hall.
2) The game displays the halls in a grid format to facilitate object placement.

**Frequency of Occurrence:** If the player restarts the game, they will need to design the halls again from scratch.

**Use Case Name:** Wizard Monster's Behavior (Updated)

**Use Case ID**: UC-11
**Scope**: ROKUE LIKE - Phase 1
**Level**: User-Goal
**Primary Actor**: Wizard Monster

**Stakeholders and Interests:**

- **Player**: Expects the wizard monster to add unpredictability and challenge based on the game's current state.
- **Developer**: Wants to implement time-based, dynamic behavior using the Strategy pattern for seamless transitions.

**Preconditions:**

- The game must be in **running/play mode**.
- The wizard monster must have spawned (triggered by game logic).
- The total game time or remaining time is tracked and updated regularly.

**Postconditions:**

- The wizard monster performs actions dynamically based on the percentage of time remaining and disappears after completing its action.

**Main Success Scenario:**

1. The wizard monster spawns in the hall.
2. The system calculates the percentage of time remaining in the game.
3. Based on the remaining time, the wizard monster chooses one of the following behaviors:
   - **<30% time remaining**:
     - Teleports the hero to a random empty location once.
     - Disappears immediately afterward.
   - **>70% time remaining**:
     - Changes the rune's location every 3 seconds until the time threshold drops below 70%.
   - **30%-70% time remaining**:
     - Remains idle in its current location.
     - Disappears after 2 seconds without taking any action.
4. If the time threshold changes while the wizard monster is present, it dynamically switches to the appropriate behavior.
5. The wizard monster disappears once its action is completed or its behavior duration ends.

**Extensions:**

- **a. System fails at any time**:
    - The player restarts the game; wizard monster's state is reset.
- **2a. Behavior conflict at threshold boundaries**:
    - The system resolves the conflict by applying the most appropriate behavior and logs the transition.

Frequency of Occurrence:

- The wizard monster appears randomly or at preset intervals, triggering the described behavior.

**Use Case Name:** Save/Load Game

**Use Case ID**: UC-12
**Scope**: ROKUE LIKE - Phase 1
**Level**: User-Goal
**Primary Actor**: Player

**Stakeholders and Interests:**

- **Player**: Wants to save the game state and load it later to continue from the exact point of progress.
- **Developer**: Aims to implement the feature using file-based storage (e.g., serialization) for saving and loading game states.

**Preconditions:**

- The game is in **running/play mode** (for saving).
- The main menu has a **Load** button for accessing previously saved files.

**Postconditions:**

- The game state is saved in a file or loaded from a selected file without losing progress.

**Main Success Scenario (Saving):**

1. The player clicks the **Save** button during gameplay.
2. The system prompts for a file name or generates one automatically (e.g., timestamp-based).
3. The system collects all relevant game data, including:
    - Hero’s position, time remaining, lives remaining.
    - Inventory contents (enchantment types and quantities).
    - Monster states (type, position, status).
    - Locations and statuses of objects (e.g., chests, runes).
4. The game state is serialized and stored in the chosen file format (e.g., JSON, XML, or Java serialization).
5. A confirmation message is displayed (e.g., “Game saved successfully”).

**Main Success Scenario (Loading):**

1. The player clicks the **Load** button on the main menu.
2. A list of saved files is displayed.
3. The player selects a file.
4. The system deserializes the selected file and restores the game state, including all hero, monster, and object positions.
5. The game resumes from the saved state.

**Extensions:**

- **a. System fails at any time**:
    - The player restarts the game.
- **2a. Save file is missing or corrupted**:

    - The system displays an error message (e.g., “Save file not found or corrupted”).
- **2b. Save permission denied**:
    - The system alerts the player to retry saving with appropriate permissions.

**Technology and Data Variations List:**

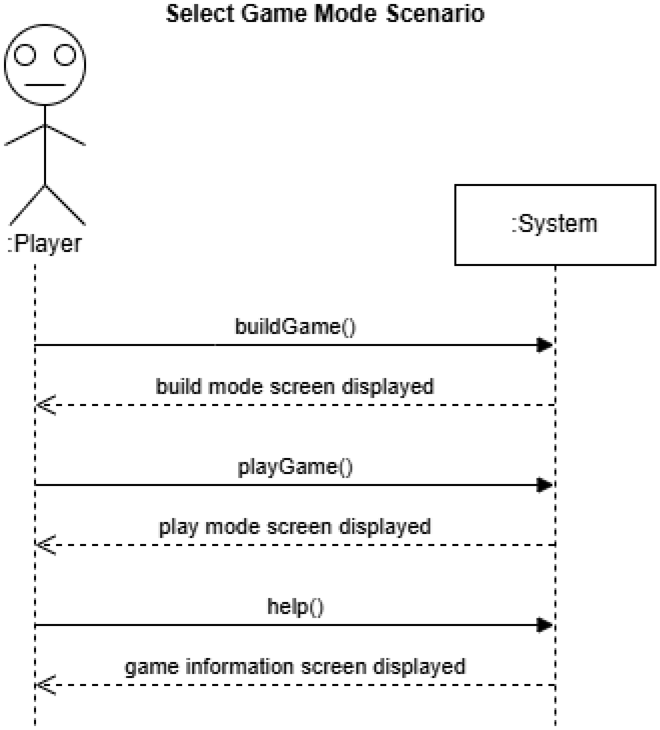
- Alternative file formats (e.g., JSON via Jackson) may be used for better readability.
- Multiple save files can be stored in a directory accessible by the game.

**Frequency of Occurrence:**

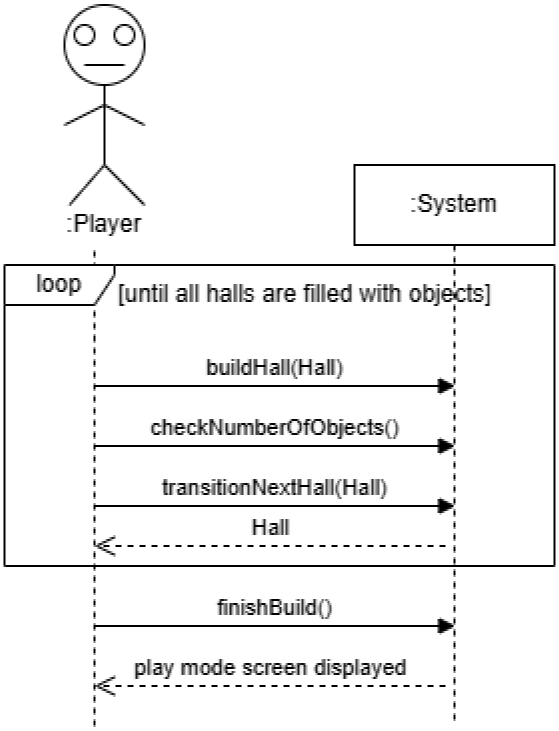
- The player can save the game at any time during gameplay.
- The player can load a saved game from the main menu before starting a new session.

# System Sequence Diagrams

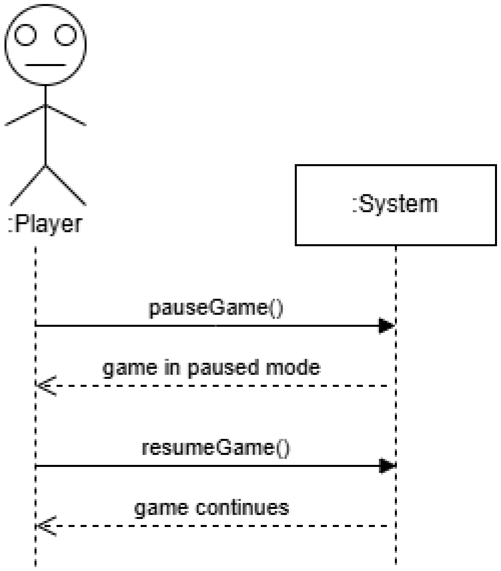
Select Game Mode Scenario


:Player
:System
buildGame()
build mode screen displayed
playGame()
play mode screen displayed
help()
game information screen displayed

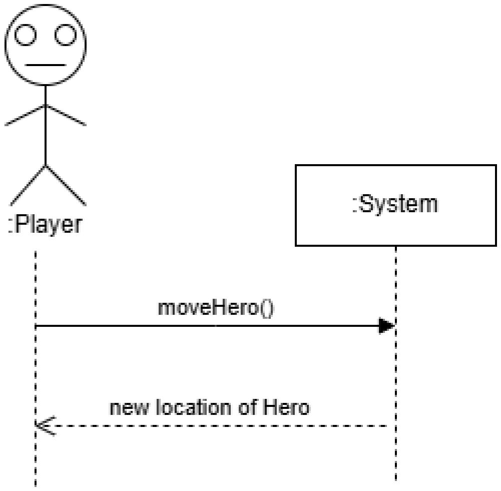
Build Game Scenario



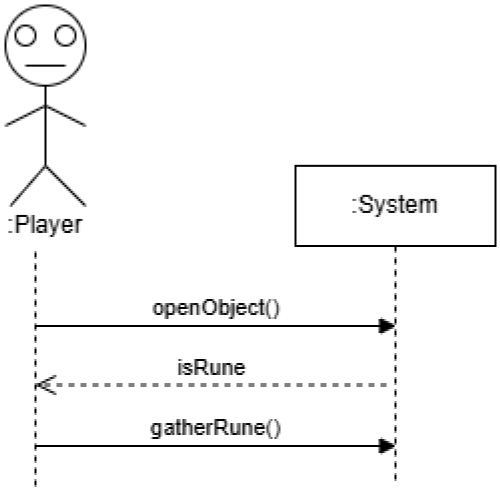
### Pause/Resume Game Scenario



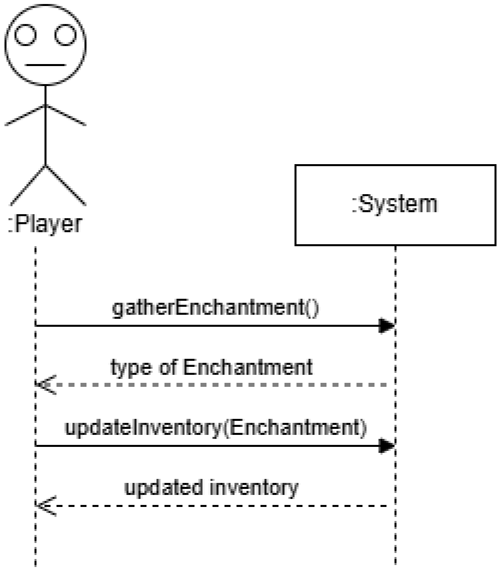
### Move Hero Scenario



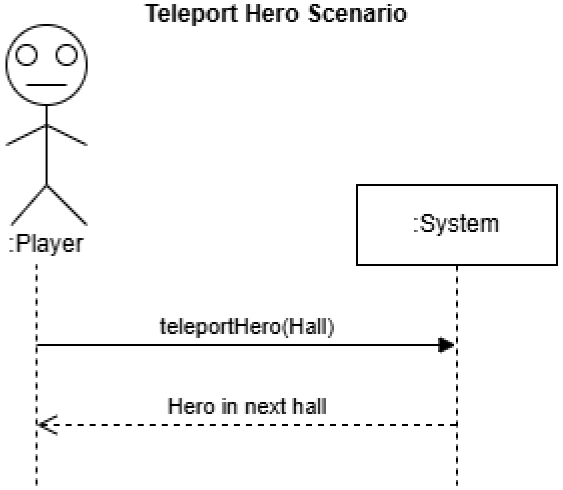
**Gather Rune Scenario**



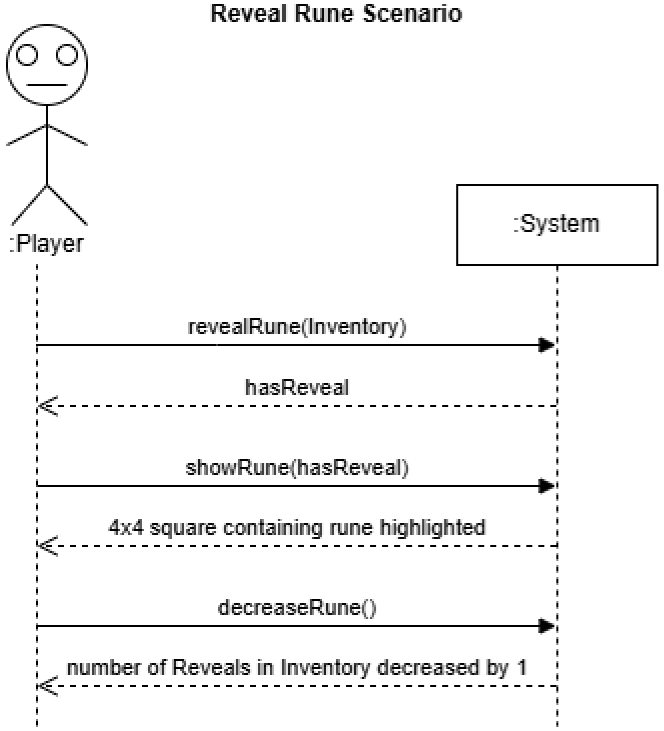
**Gather Enchantment Scenario**



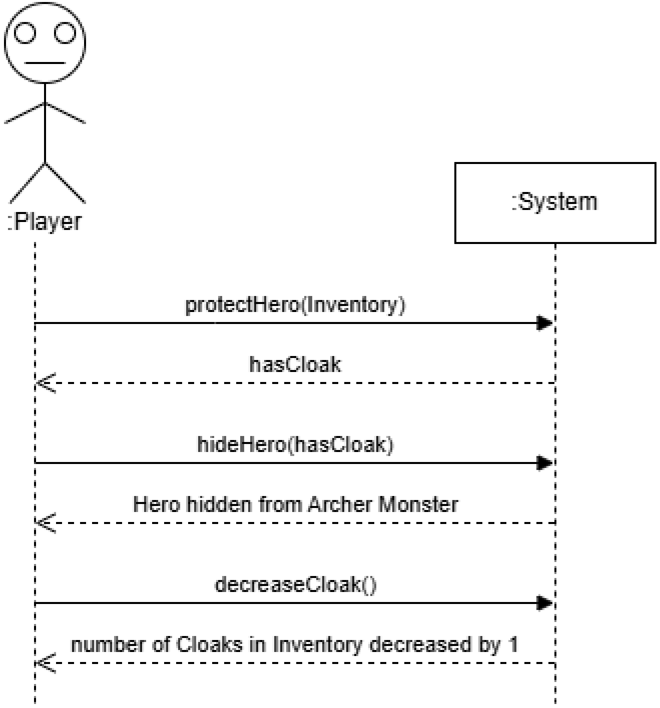
**Teleport Hero Scenario**



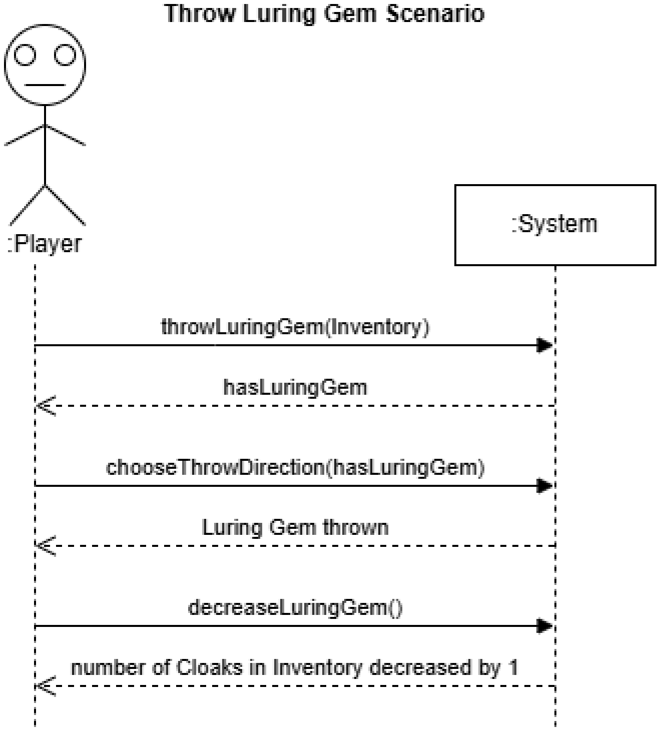
**Reveal Rune Scenario**


:Player
:System
revealRune(Inventory)
hasReveal
showRune(hasReveal)
4x4 square containing rune highlighted
decreaseRune()
number of Reveals in Inventory decreased by 1

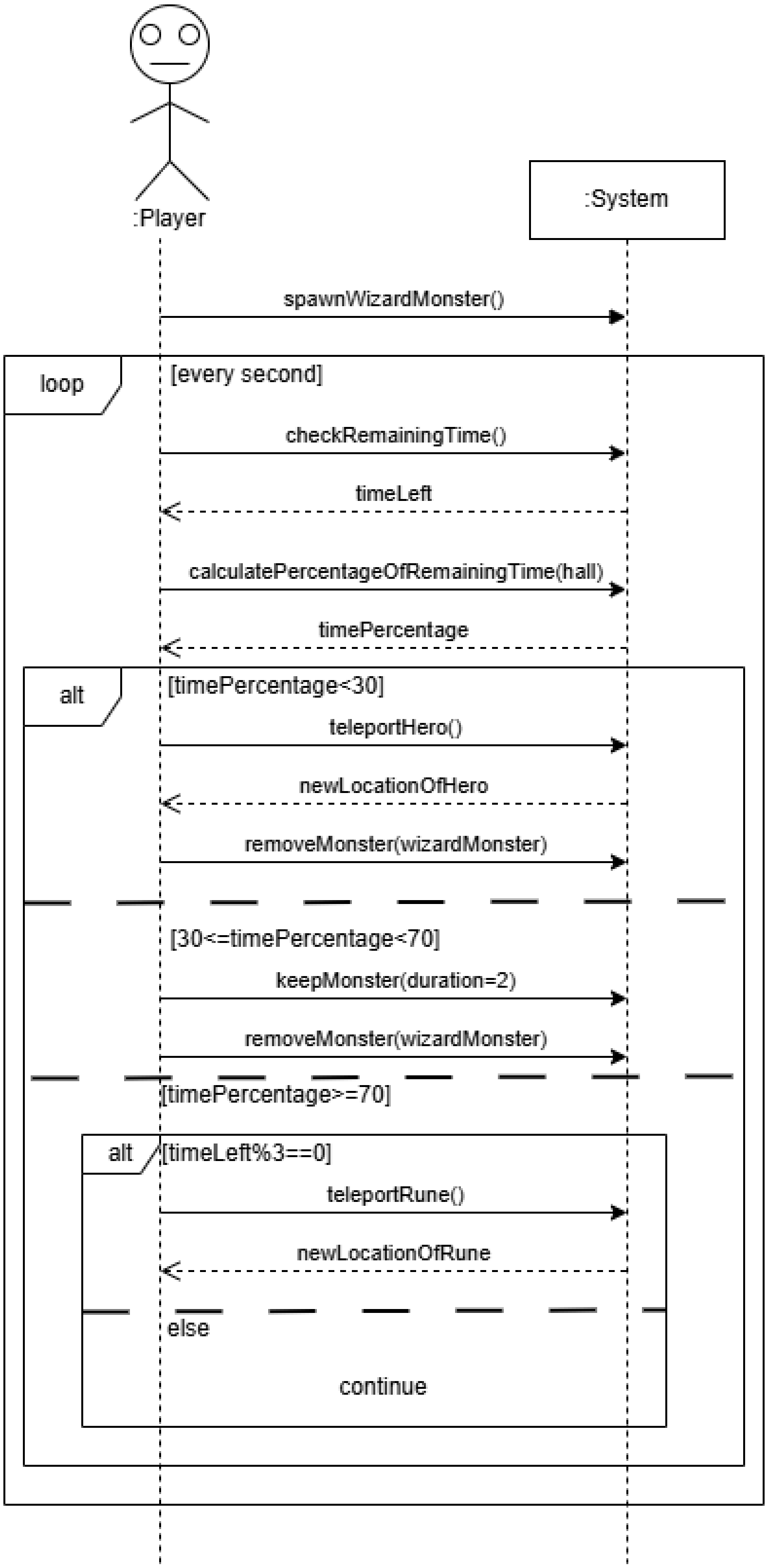
**Protect Hero Scenario**



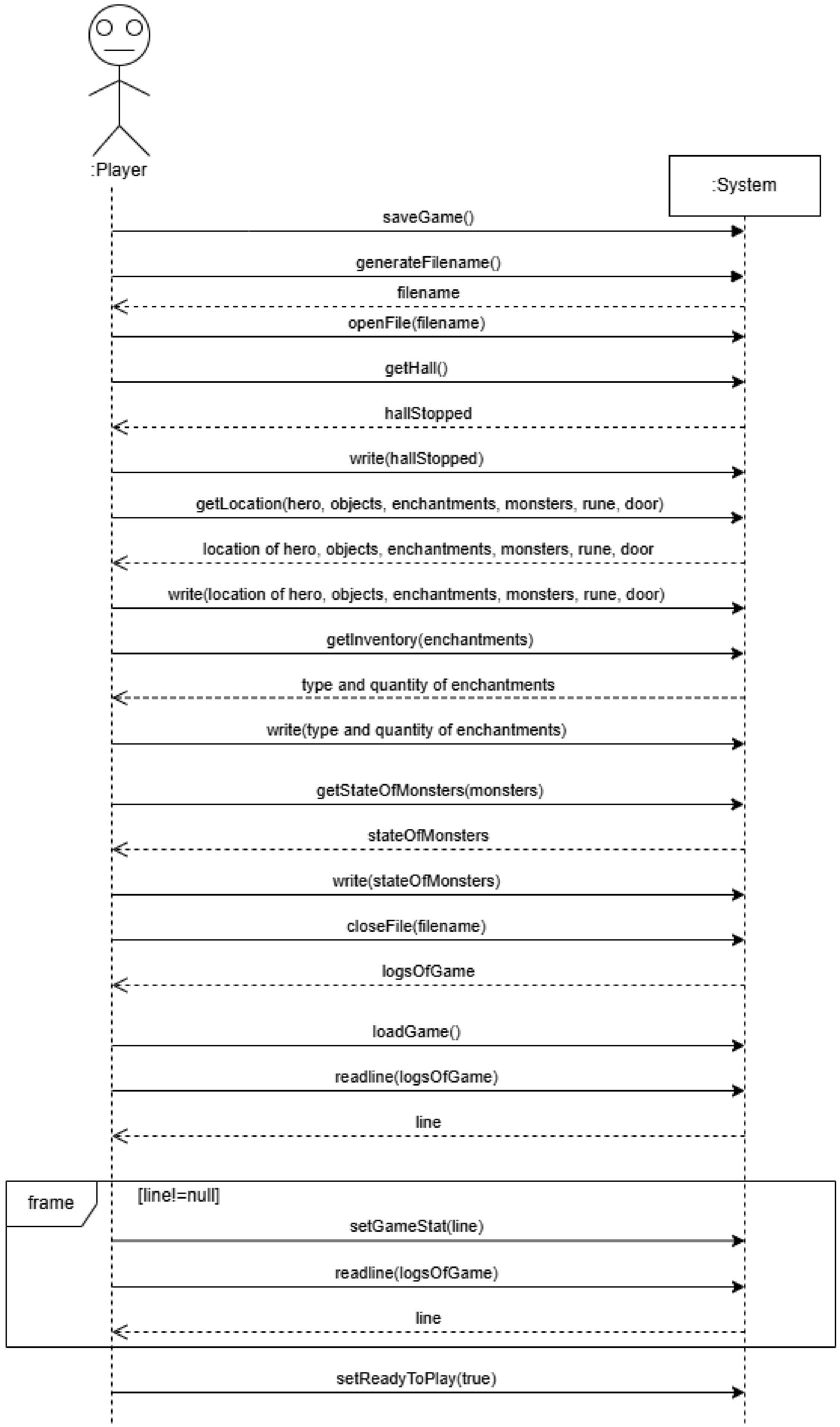
**Throw Luring Gem Scenario**


:Player
:System
throwLuringGem(Inventory)
hasLuringGem
chooseThrowDirection(hasLuringGem)
Luring Gem thrown
decreaseLuringGem()
number of Cloaks in Inventory decreased by 1

**Choose Wizard Monster Behavior**



**Save/Load Game**


:Player
:System
saveGame()
generateFilename()
filename
openFile(filename)
getHall()
hallStopped
write(hallStopped)
getLocation(hero, objects, enchantments, monsters, rune, door)
location of hero, objects, enchantments, monsters, rune, door
write(location of hero, objects, enchantments, monsters, rune, door)
getInventory(enchantments)
type and quantity of enchantments
write(type and quantity of enchantments)
getStateOfMonsters(monsters)
stateOfMonsters
write(stateOfMonsters)
closeFile(filename)
logsOfGame
loadGame()
readline(logsOfGame)
line
frame
[line!=null]
setGameStat(line)
readline(logsOfGame)
line
setReadyToPlay(true)

## Operation Contracts

**Operation:** pauseGame()

**References: Use Cases:** Pause/Resume Game

**Pre-conditions:** The game is in playing mode. Specifically, the game is running.

**Post-conditions:** The game was paused.

**Operation:** resumeGame()

**References: Use Cases:** Pause/Resume Game

**Pre-conditions:** The game is in playing mode. Specifically, the game is paused.

**Post-conditions:** The game resumed.

**Operation:** buildGame()

**References: Use Cases:** Select Game Mode

**Pre-conditions:** The game must be launched, and the player must be at the Main Menu screen.

**Post-conditions:** It shows us the map of all the halls.

**Operation:** playGame()

**References: Use Cases:** Select Game Mode

**Pre-conditions:** -All halls were filled with at least as many as the minimum number of objects as specified in the game rules.

**Post-conditions:** -The play mode is successfully initialized

-The map of the Hall of Earth was displayed.

**Operation:** help()

**References: Use Cases:** Select Game Mode

**Pre-conditions:** Game application is executed and working successfully.

**Post-conditions:** It helps us when clicked. Information regarding how to use the interface, play the game and some fundamental rules of the game were displayed.

**Operation:** buildHall(hall)

**References: Use Cases:** BuildGame

**Pre-conditions:** **-**The player has clicked Build Mode from the Main Menu.

-The game transitioned to Build Mode.

**Post-conditions:** -The objects are placed in the hall.

**Operation:** checkNumberOfObjects()

**References: Use Cases:** BuildGame

**Pre-conditions:** -The player has clicked Build Mode from the Main Menu.

-The game transitioned to Build Mode.

**Post-conditions:** The number of objects populated in the hall based on the player's object placement choice was compared with the minimum number of object criteria for that hall.

**Operation:** transitionNextHall(Hall)

**References: Use Cases:** BuildGame

**Pre-conditions:** The value of checkNumberOfObject must be at least the required minimum number.

**Post-conditions:** -Next hall was unlocked to place the objects on it.

**Operation:** finishBuild()

**References: Use Cases:** BuildGame

**Pre-conditions:** All halls must be successfully filled with objects according to the rules.

**Post-conditions:** The player proceeds to Play Mode.

**Operation:** moveHero()

**References: Use Cases:** Move Hero

**Pre-conditions:** The game is in play mode.

**Post-conditions:** -The hero was moved east/west/south/north in response to user keyboard commands.

-The hero remained within the boundaries of the hall.

**Operation:** openObject()

**References: Use Cases:** Gather Rune

**Pre-conditions:** -The game is in play mode.

-Hero is next to the object.

**Post-conditions:** The rune emerged if it was hidden in that object before opening. Otherwise, nothing was displayed and the object was closed.

**Operation:** gatherRune()

**References: Use Cases:** Gather Rune

**Pre-conditions:** -The game is in play mode

-We know that the rune exists.

**Post-conditions:** -The door of the hall opened.
-The system made a sound indicating that the door was opened.

**Operation:** gatherEnchantment()

**References: Use Cases:** Gather Enchantment

**Pre-conditions:** The game is in playing mode

**Post-conditions:** -The enchantments have been manipulated for players' decisions (used or sent to inventory).
-The enchantment has been moved to the inventory or used.

**Operation:** updateInventory(Enchantment)

**References: Use Cases:** Gather Enchantment

**Pre-conditions:** -The game is in play mode.

-We need to be nearby.

**Post-conditions:** Hero’s inventory has been updated based on gathered enhancements

**Operation:** teleportHero(Hall)

**References: Use Cases:** Teleport Hero

**Pre-conditions:** -The game is in running mode.

-The player successfully opened the door.

**Post-conditions:** -The hall is updated accordingly to the next stage.
-Hero is teleported to the next hall.

**Operation:** revealRune(Inventory)

**References: Use Cases:** Reveal Rune

**Pre-conditions:** -The game is in running mode.

**Post-conditions:** Number of reveal enchantments in the Inventory was checked.

**Operation:** showRune(hasReveal)

**References: Use Cases:** Reveal Rune

**Pre-conditions:** -The game is in running mode.

-The hero has at least one reveal enchantment in his/her inventory.

-The hero used one of his/her reveal enchantments.

**Post-conditions:** 4x4 square one of which contains the rune is highlighted.

**Operation:** decreaseRune()

**References: Use Cases:** Reveal Rune

**Pre-conditions:** The game is in running mode.

**Post-conditions:** -The number of reveal enchantments has decreased in the hero's inventory by 1.

**Operation:** protectHero(Inventory)

**References: Use Cases:** Protect Hero

**Pre-conditions:** **-**The game is in running mode.

**Post-conditions:** Number of Cloak of Protection enchantments in the Inventory was checked.

**Operation:** hideHero(hasReveal)

**References: Use Cases:** Protect Hero

**Pre-conditions:** The player has at least one cloak of protection in his/her inventory.

**Post-conditions:** The archer can no longer see the hero.

**Operation:** decreaseCloak()

**References: Use Cases:** Protect Hero

**Pre-conditions:** The game is in running mode.

**Post-conditions:** -The number of the cloak of protection enchantments has decreased in the hero's inventory by 1.

**Operation:** throwLuringGem(Inventory)

**References: Use Cases:** Throw Luring Gem

**Pre-conditions:** **-**The game is in running mode

**Post-conditions:** Number of Luring Gem enchantments in the Inventory was checked.

**Operation:** chooseThrowDirection(hasLuringGem)

**References: Use Cases:** Throw Luring Gem

**Pre-conditions:** **-**The game is in playing mode

-The player has at least one cloak of protection in his/her inventory.

**Post-conditions:** Fighter monster started moving towards the luring gem.

**Operation:** decreaseLuringGem()

**References: Use Cases:** Throw Luring Gem

**Pre-conditions:** The game is in running mode

**Post-conditions:** The number of Luring Gem enchantments has decreased in the hero's inventory by 1.

## PHASE 2

**Operation:** SpawnWizard()

**Preferences: Use Cases:** Choose Wizard Monster's Behavior

**Pre-conditions:** -The game must be in running mode

-The wizard monster must have spawned

-Total time and remaining time is tracked and updated regularly

**Post-conditions:** 1a: If less than 30% of the total time remains, the monster concludes the hero is close to losing, moves the player to a random empty location once, and disappears.

1b: If more than 70% of the total time remains, the monster concludes the hero is doing well, making the game challenging by changing the location of the rune every 3 seconds.

1c: If the remaining time is between 30%-70%, the monster is indecisive, stays in place for 2 seconds, then disappears without doing anything.

**Operation:** SaveGame()

**Preferences: Use Cases:** Save Game

**Pre-conditions:** -The game is already in playing mode

-Playing mode has save button

**Post-conditions:** The game state is saved in a file or loaded from a selected file without losing progress.

**Operation:** LoadGame()

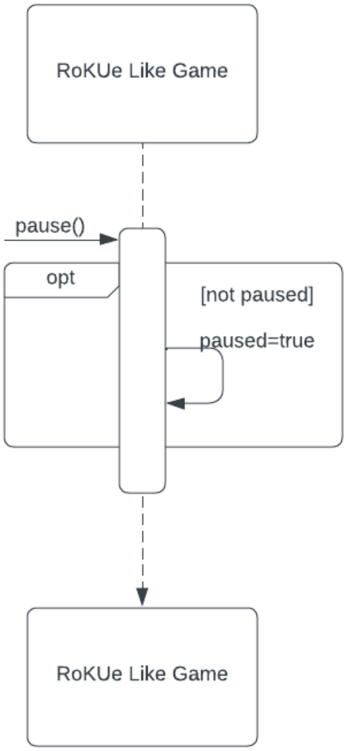
**Preferences: Use Case**s: Load Game

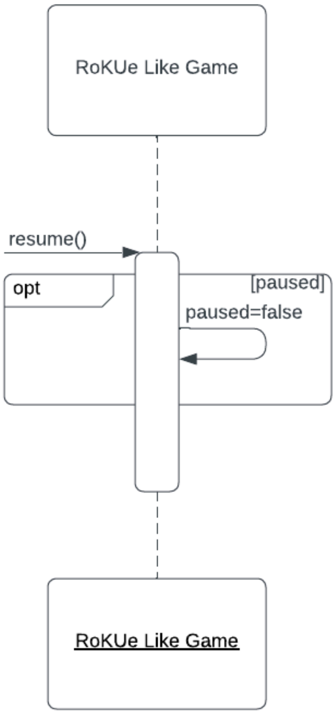
**Pre-conditions:** -The game is already in playing mode

-The main menu has load button

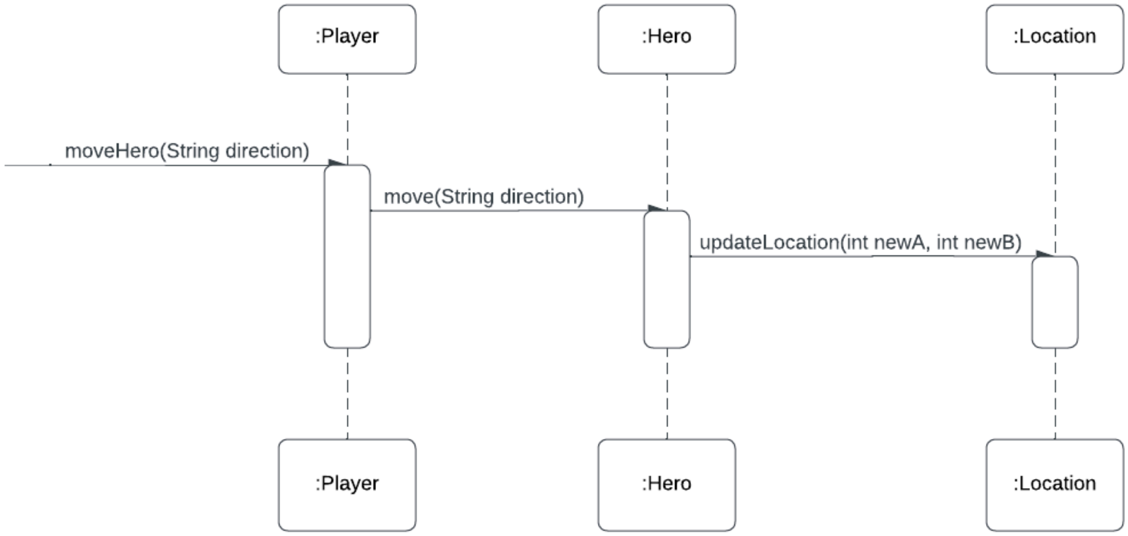
**Post-conditions:** Previously saved game will load correctly

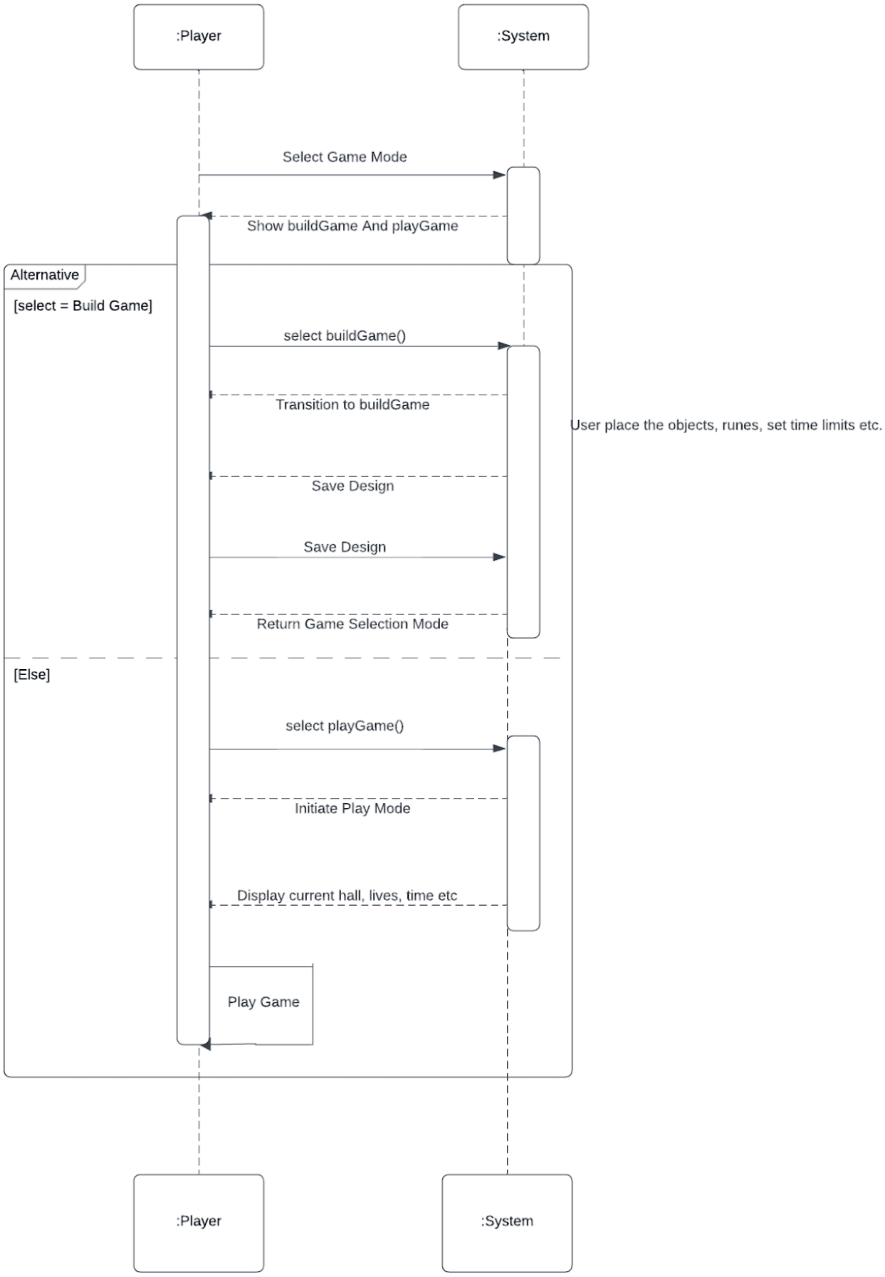
# Interaction Diagrams

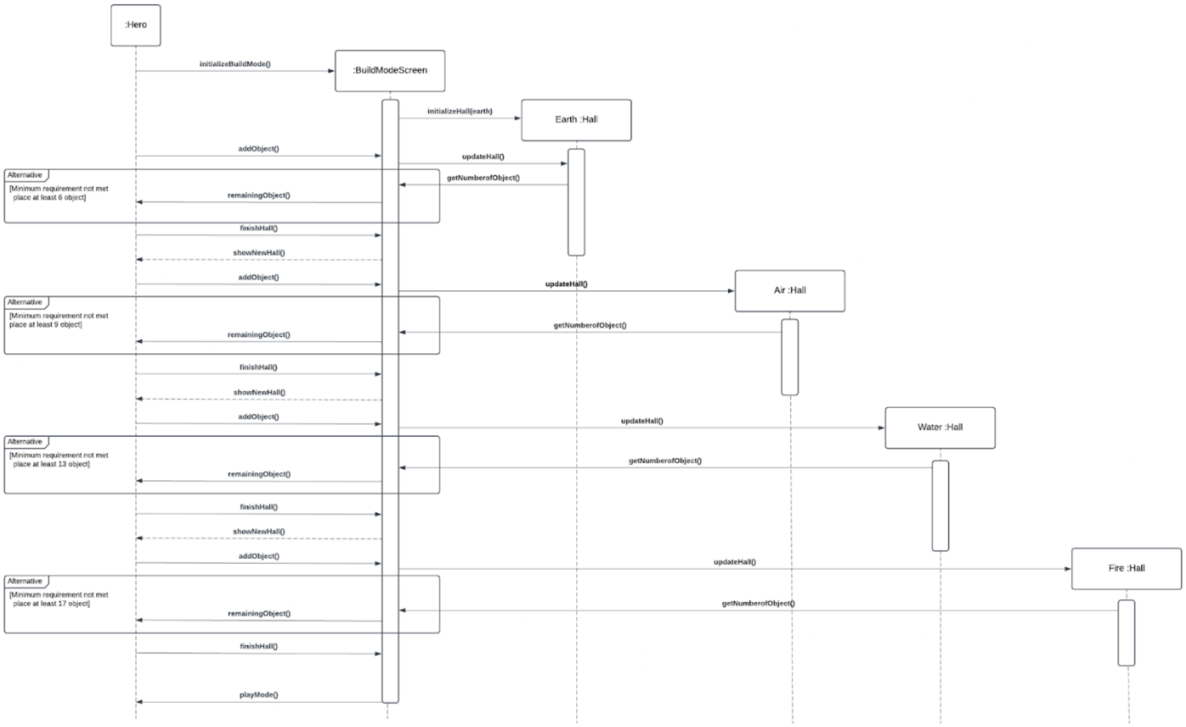


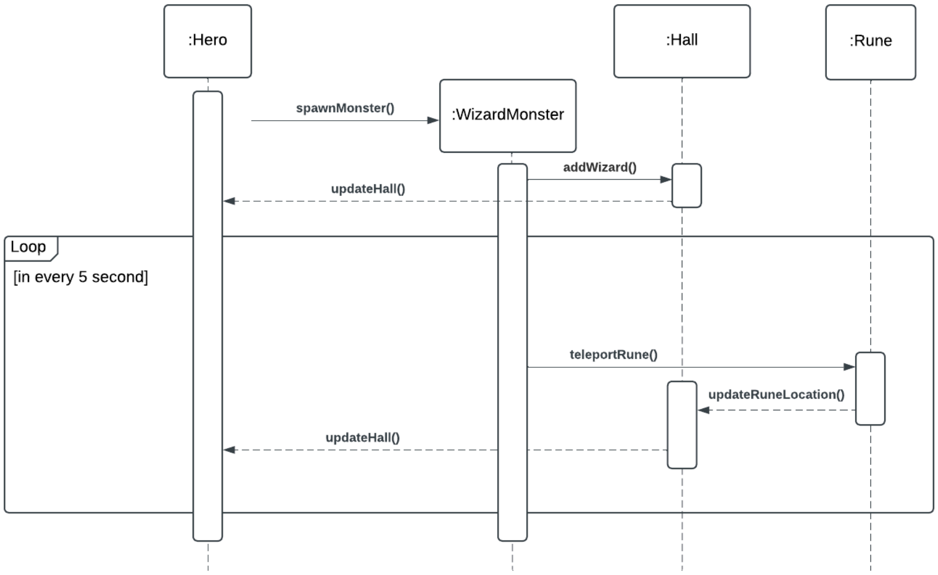


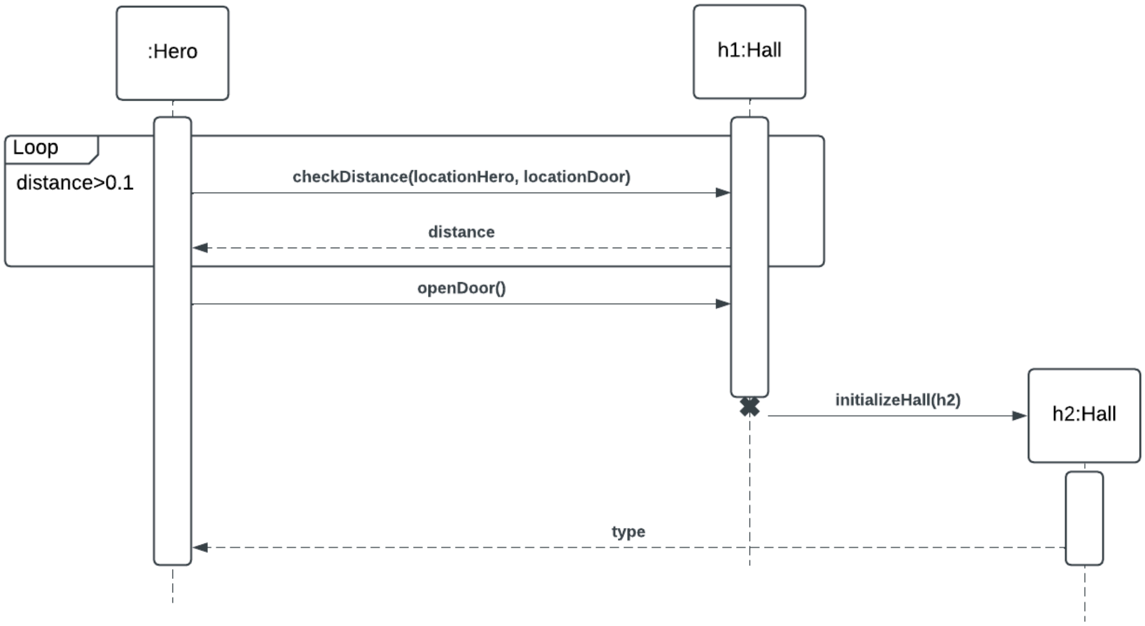
Move Hero

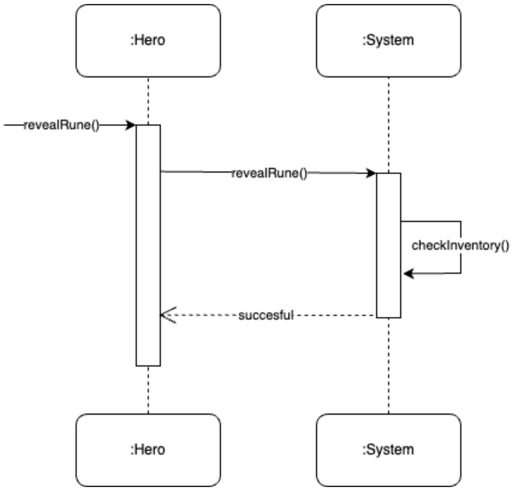

:Player
:Hero
:Location
moveHero(String direction)
move(String direction)
updateLocation(int newA, int newB)
:Player
:Hero
:Location

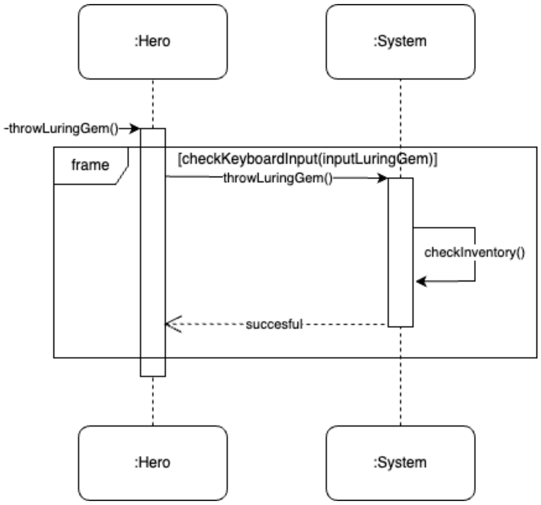

:Player
:System
Select Game Mode
Show buildGame And playGame
Alternative
[select = Build Game]
select buildGame()
Transition to buildGame
User place the objects, runes, set time limits etc.
Save Design
Save Design
Return Game Selection Mode
[Else]
select playGame()
Initiate Play Mode
Display current hall, lives, time etc
Play Game
:Player
:System

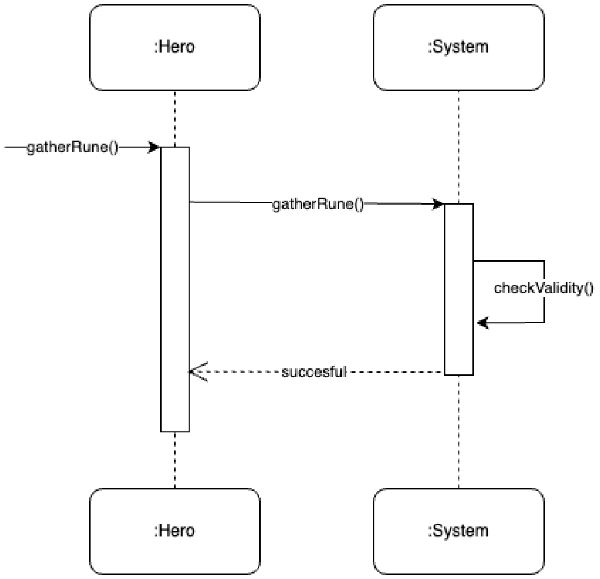

:Hero
initializeBuildMode()
:BuildModeScreen
initializeHall(earth)
Earth :Hall
addObject()
updateHall()
Alternative
[Minimum requirement not met
place at least 6 object]
getNumberofObject()
remainingObject()
finishHall()
showNewHall()
addObject()
updateHall()
Air :Hall
Alternative
[Minimum requirement not met
place at least 9 object]
getNumberofObject()
remainingObject()
finishHall()
showNewHall()
addObject()
updateHall()
Water :Hall
Alternative
[Minimum requirement not met
place at least 13 object]
getNumberofObject()
remainingObject()
finishHall()
showNewHall()
addObject()
updateHall()
Fire :Hall
Alternative
[Minimum requirement not met
place at least 17 object]
getNumberofObject()
remainingObject()
finishHall()
playMode()

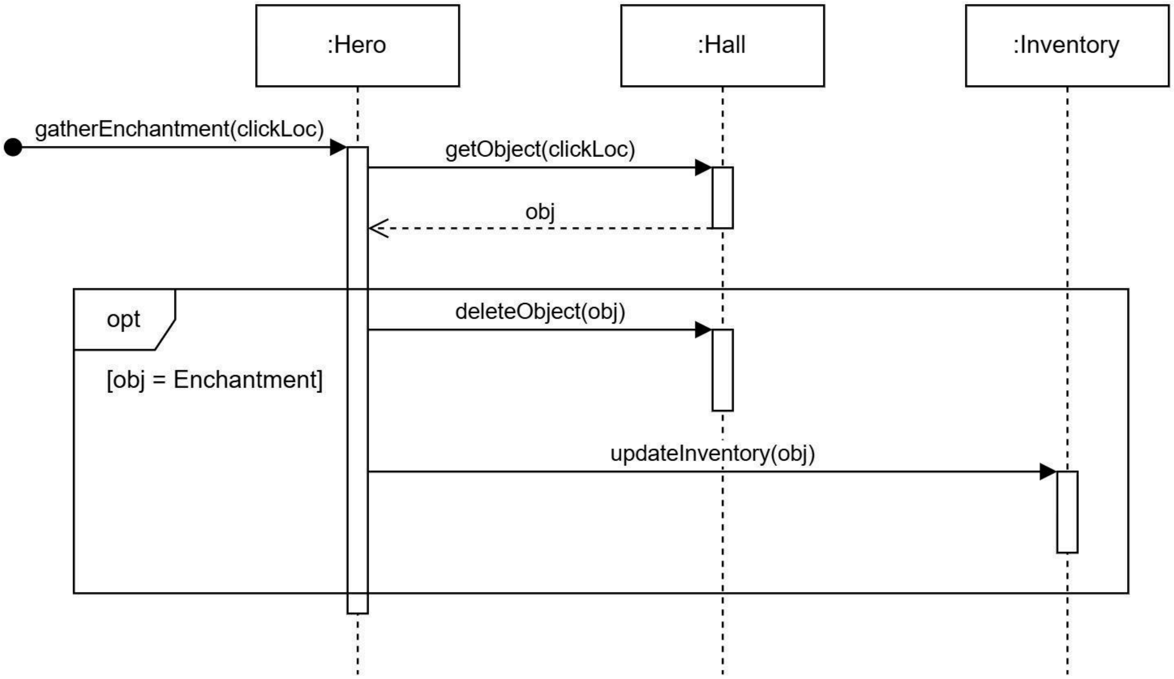

:Hero
:Hall
:Rune
spawnMonster()
:WizardMonster
addWizard()
updateHall()
Loop
[in every 5 second]
teleportRune()
updateRuneLocation()
updateHall()

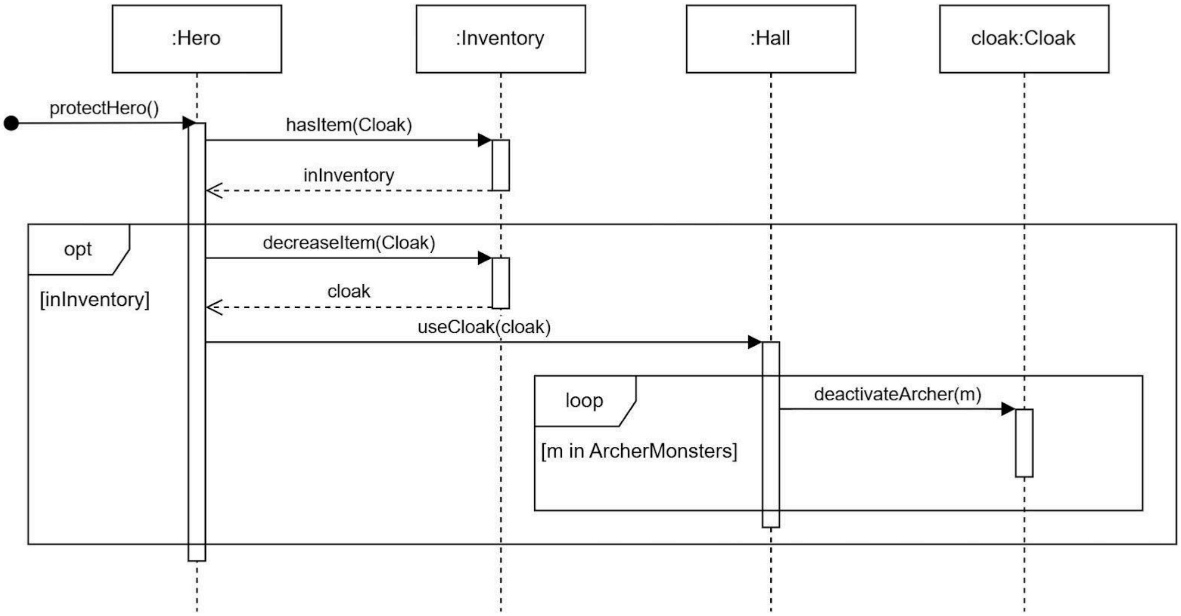

:Hero
h1:Hall
Loop
distance>0.1
checkDistance(locationHero, locationDoor)
distance
openDoor()
initializeHall(h2)
h2:Hall
type

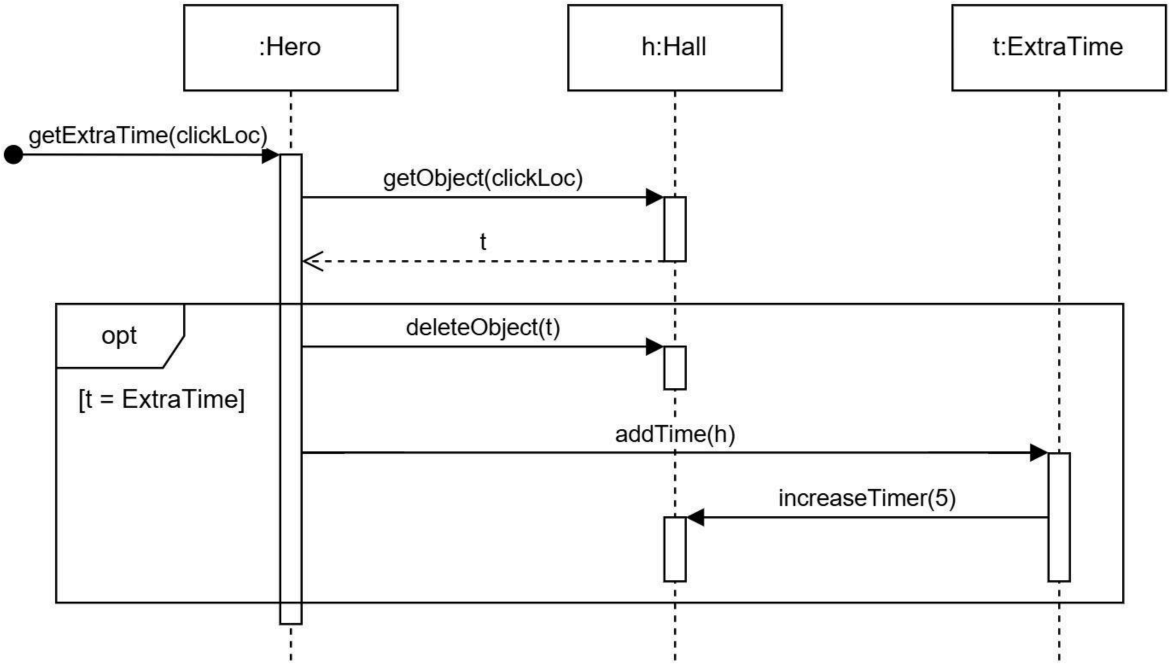

:Hero
:System
revealRune()
revealRune()
checkInventory()
succesful
:Hero
:System

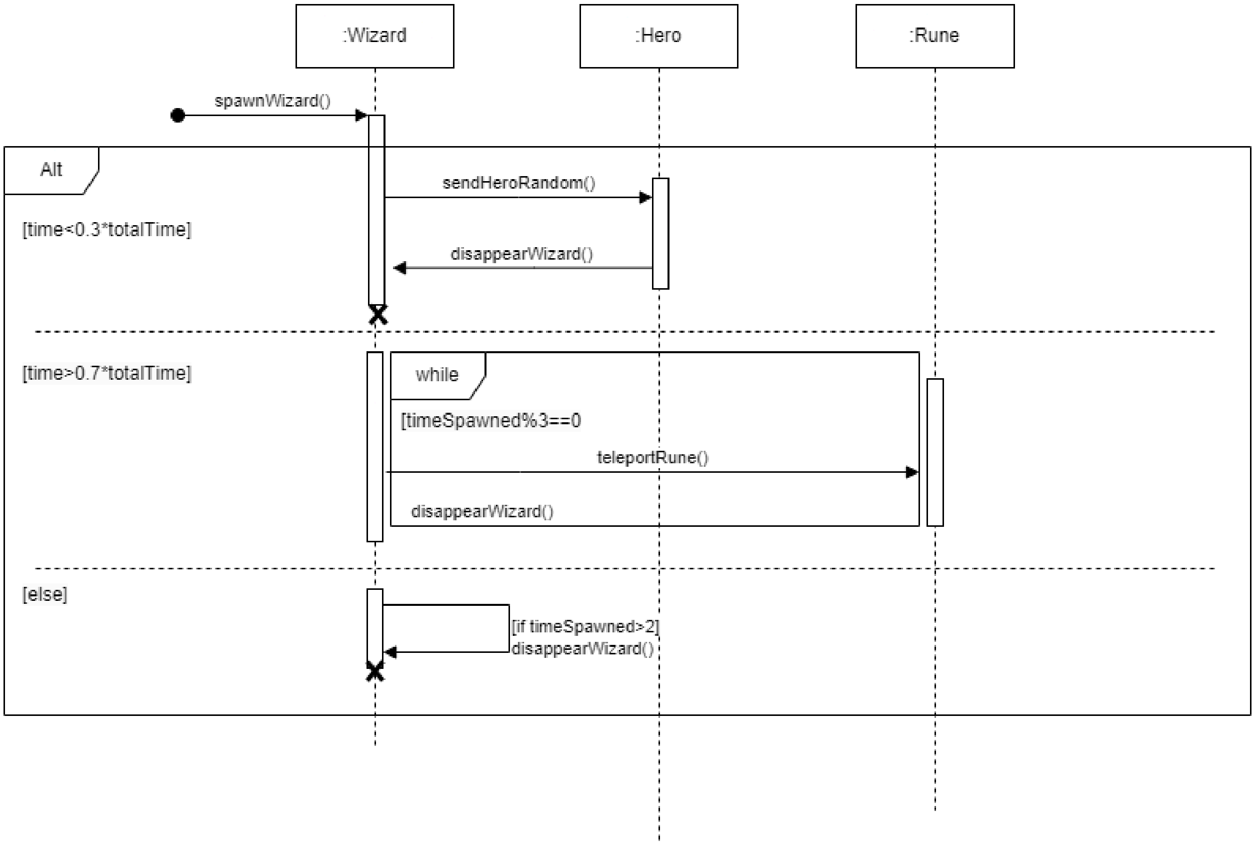

:Hero
:System
throwLuringGem()
frame
[checkKeyboardInput(inputLuringGem)]
throwLuringGem()
checkInventory()
succesful
:Hero
:System

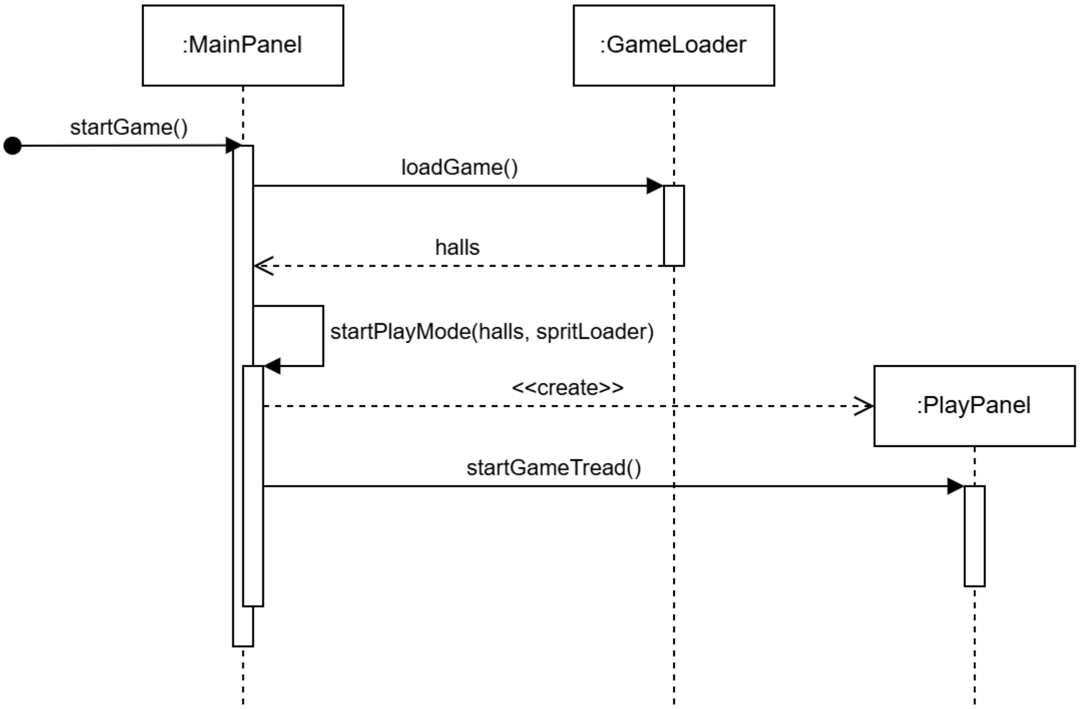

:Hero
:System
gatherRune()
gatherRune()
checkValidity()
succesful
:Hero
:System

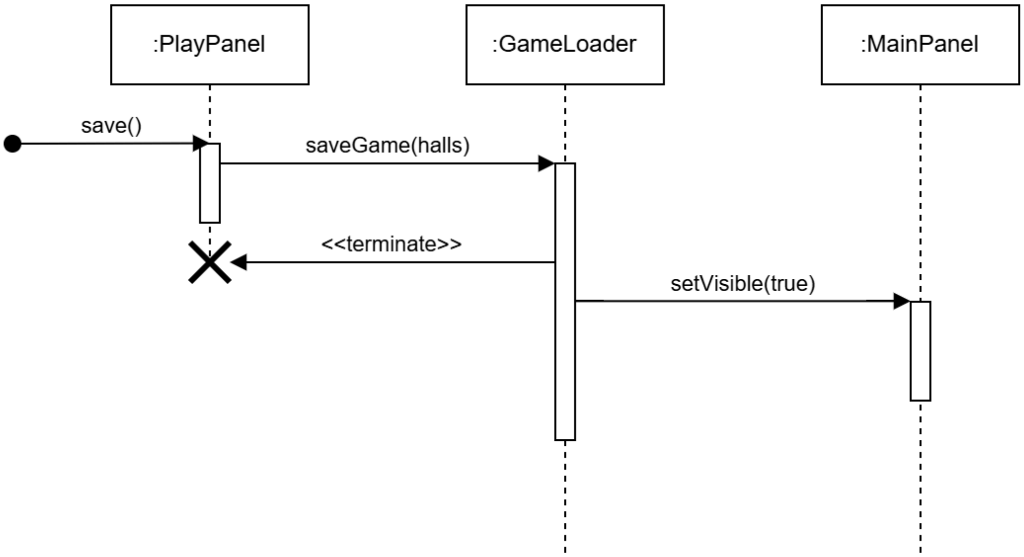

:Hero
:Hall
:Inventory
gatherEnchantment(clickLoc)
getObject(clickLoc)
obj
opt
deleteObject(obj)
[obj = Enchantment]
updateInventory(obj)

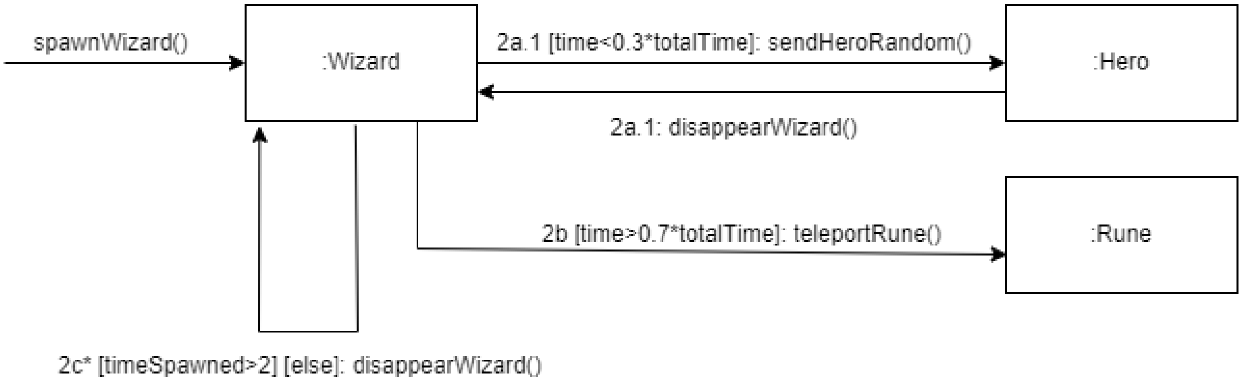

:Hero
:Inventory
:Hall
cloak:Cloak
protectHero()
hasItem(Cloak)
inInventory
opt
[inInventory]
decreaseItem(Cloak)
cloak
useCloak(cloak)
loop
[m in ArcherMonsters]
deactivateArcher(m)

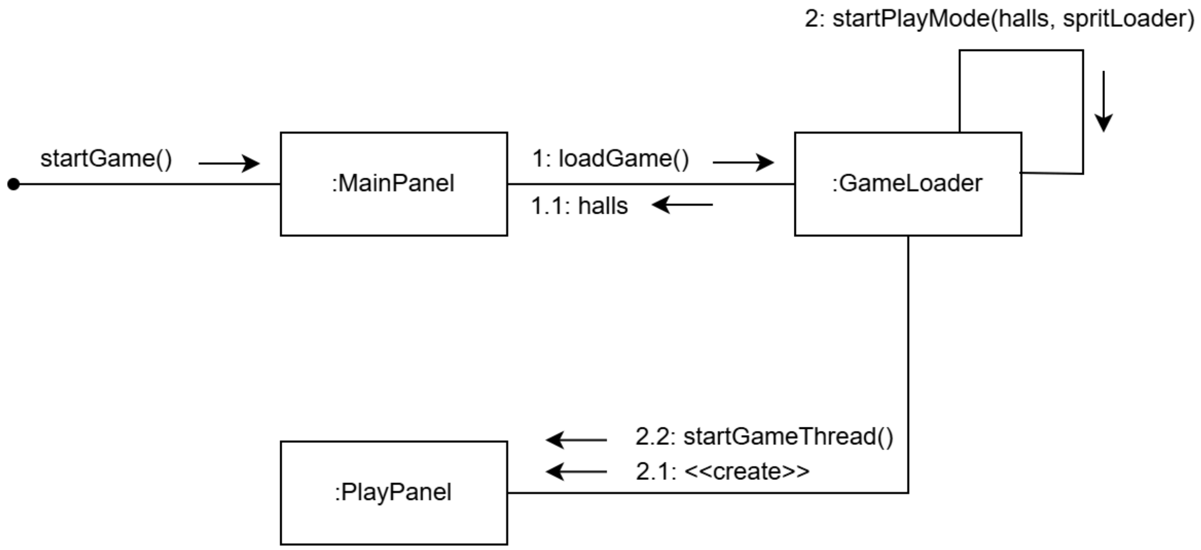

:Hero
h:Hall
t:ExtraTime
getExtraTime(clickLoc)
getObject(clickLoc)
t
opt
[t = ExtraTime]
deleteObject(t)
addTime(h)
increaseTimer(5)

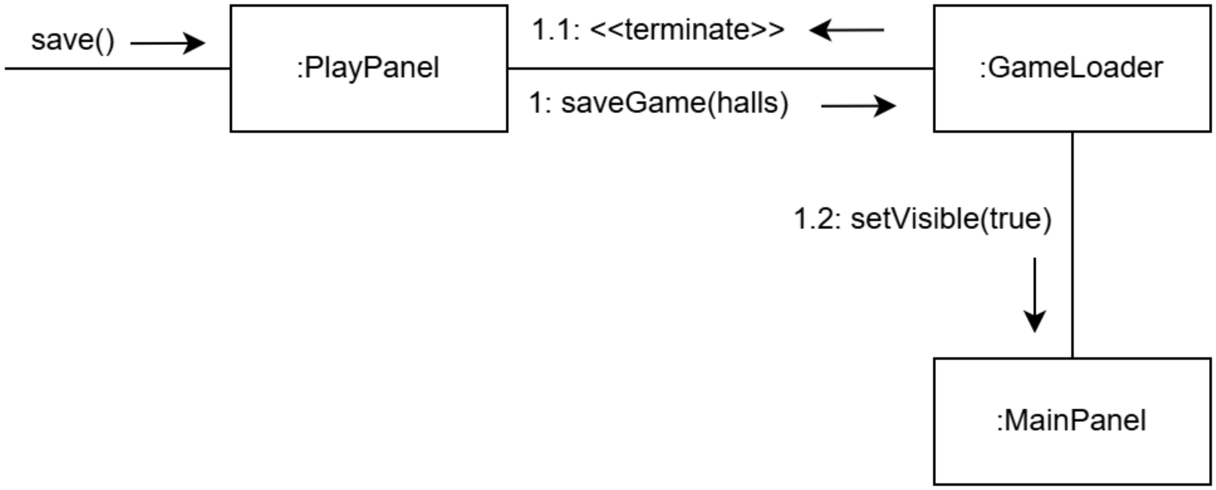

:Wizard
:Hero
:Rune
spawnWizard()
Alt
sendHeroRandom()
[time<0.3*totalTime]
disappearWizard()
[time>0.7*totalTime]
while
[timeSpawned%3==0
teleportRune()
disappearWizard()
[else]
[if timeSpawned>2]
disappearWizard()


:MainPanel
:GameLoader
startGame()
loadGame()
halls
startPlayMode(halls, spritLoader)
<<create>>
:PlayPanel
startGameTread()

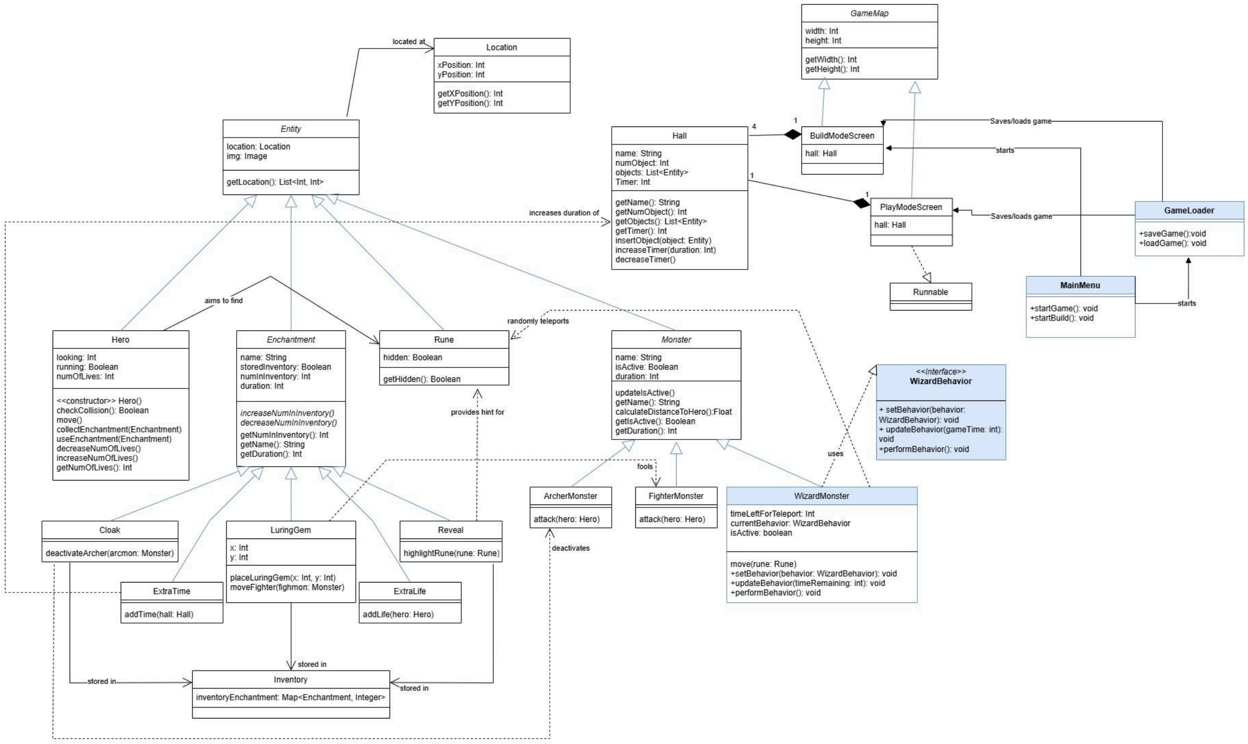

:PlayPanel
:GameLoader
:MainPanel
save()
saveGame(halls)
<<terminate>>
setVisible(true)


spawnWizard()
:Wizard
2a.1 [time<0.3*totalTime]: sendHeroRandom()
:Hero
2a.1: disappearWizard()
2b [time>0.7*totalTime]: teleportRune()
:Rune
2c* [timeSpawned>2] [else]: disappearWizard()

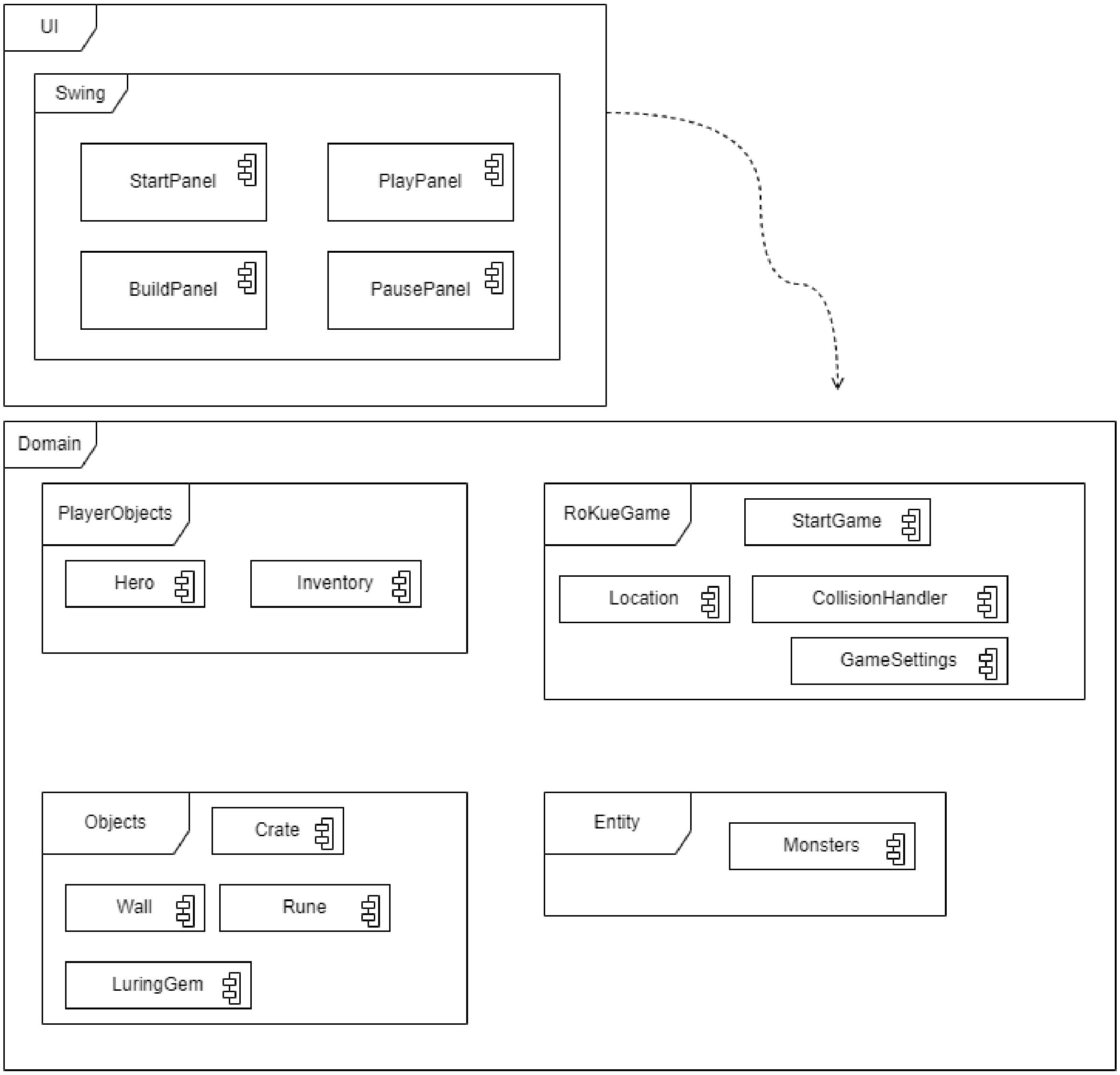

2: startPlayMode(halls, spritLoader)
startGame()
:MainPanel
1: loadGame()
1.1: halls
:GameLoader
2.2: startGameThread()
2.1: <<create>>
:PlayPanel


save()
:PlayPanel
1.1: <<terminate>>
1: saveGame(halls)
:GameLoader
1.2: setVisible(true)
:MainPanel

# Class Design Diagrams



## Logical Architecture



# Supplementary Specifications

**Introduction**

This document is the repository of all RoKUlike 302-Phase 1 Requirements not captured in the use cases.

**Functionality**

The System does lots of actions in the Game Mode (Running the AI of the monsters, deciding the location of the rune etc.). The details are explained in Application-Specific Domain Rules.

**Usability**

Human Factors

- The game should ensure that all items except the rune are clearly visible to the Player on the screen.
- User interface should be easy to understand and navigate.
- The system should give warnings when the Player does something outside the specifications of the game rules.
- The system must respond to the user input quickly.

**Reliability**

Recoverability

If there is an error, Player can restart the Game.

**Performance**

Game must not crash. Game must run smoothly, without visible fps drops.

**Supportability**

Configurability

The Player can configure the Game in Building Mode.

Adaptability

The Game should be built in a way that makes adding new features easy.

**Implementation Constraints**

RoKUlike-302 Project Team uses Java (standard Java libraries, Java Swing, etc.) to implement the Game.

**Application-Specific Domain Rules**

General Game Rules:

- L is the default distance unit in the game. All the dimensions are going to be driven from this unit. By default, L is one pixel of the game, which is standard in all pixel art files
- The Hero, Fighter, and arrows from the Archer will move in L/secs.
- The size of the Hero is 16x16 L, along with the other enemies.
- There are 5 enchantments: Extra time, Extra life, Reveal, Cloak of Protection and Luring Gem.
- Enchantments can be found in the chests that are present in the Halls of the game.
- There are two main use types of the enchantments, namely:

  - Instantly used: Extra life and Extra time items are used up at the moment they are picked up, so they don’t take up inventory space.
  - Stored in inventory: Reveal, Cloak of Protection, and Luring Gem are stored in the inventory once they are picked up. They can be used with their dedicated keyboard key, and the amount of the item used is decreased from the inventory.
- The chests are places where the enchantments and runes can be found. They are placed in the game during build mode. There needs to be at least one chest that contains the rune.
- Rune is a special item that only has one instance in each hall; after collecting the rune, Player goes to exit to finish the Hall. If every hall is complete, the game ends.
- The rune and enchantments are picked up by opening the chests and pressing the move button that points towards it.
- Monsters are spawned to the Halls every 8 seconds at a random location.
- Archer monster shoots an arrow every second, if the arrow connects with the player they lose a life. Archer monster should be within a distance of 4 square to the hero to make he hero lose a life. If the Hero uses a Cloak of Protection, Archer cannot see the Hero.
- Fighter monster cannot see so they move around randomly. Fighter monster must be next to the hero to stab him/her with a dagger. If the Player placed a Luring Gem on the ground, the Fighter will walk towards it.
- Wizard monster does not move or attack the player. However, they teleport the rune to a random chest every 5 seconds and the hero is not able to view teleportation.
- Using Reveal enchantment causes a 4x4 grid window to be highlighted, which one of them showing the current location of the rune

Building Mode Rules:

- Player enters the Build mode from the title screen.
- In Build mode, the Player can see all 4 Halls of the game and has access to a sidebar that includes decorative items, chests, and walls that can be placed in the Halls.
- After the Player finishes designing the levels, Halls need to be checked for the number of items they contain, whether there is a path from anywhere to chests, and chests to the exit, to prevent soft locking.
- If the Halls satisfy the requirements, the data for these levels are stored so they can be used in the future.

Enchantments:
The enchantment number is increased by 1 when enchantment is picked up.
The enchantment number is decreased by 1 when enchantment is used up.
Enchantments are used by their dedicated keys on the keyboard.
If there are multiple of the same enchantment in the Inventory, the system displays how many of them exist on top of the respective enchantment icon.
If the Player tries to use an enchantment they do not have, the system gives a warning.

Lives:
Player starts with 3 lives at the beginning of the game.
Every arrow and stab causes the player to lose one life.
If the player loses all of their lives before finding the runes and going to exit, they lose the game.

Timer:
The player has initially has a duration of 5*(number of objects in the hall) seconds to complete the hall.

## Glossary

Hero: A character that is controlled by the player. Can move in 4 directions on the grid, pick up enchantments, and use enchantments

Enchantments: Items that appear with random type and at random location in the halls with a fixed time interval. They can be picked up, used by the Hero immediately, or stored in the Inventory for later use. Includes Reveal, Cloak of protection, luring gem, extra life and extra time

Monsters: Enemies encountered in the play mode. They aim to stop players from finishing the game. Includes archer, fighter, and wizard.

Rune: An item that is found in one of the chests of every hall of the game. The hero needs to pick up this item in order to complete the hall and the game

Halls: Levels in the game. Halls are created by the player in the Building mode. Includes Hall of Earth, Hall of Fire, Hall of Water, and Hall of Air.

Building Mode: The game starts with building mode. The player creates Halls of the game in this mode

Play mode: It starts after Building Mode. It is the mode where the hero encounters Monsters, and collects Enchantments and runes.

Inventory: This is the place where some type of Enchantments are kept. The player can use enchantments if they have them in their inventory.

Reveal: One of the enchantments in the Halls. Can be used by pressing **R**. Highlights a 4x4 square one of which contains the rune.

Cloak of protection: One of the enchantments in the Halls. Can be used by pressing **P**. Hides the Hero from the archer monster for 20 seconds.

Luring gem: One of the enchantments in the Halls. Can be used by pressing **B**, then pressing one of the following buttons: **W** **A** **S** **D**. Luring gem is thrown based on the direction of the pressed key. Used to fool the fighter monster

Extra time/Extra life: Two of the enchantments in the Halls. Adds 5 seconds to the timer/ an extra life respectively.

Timer: a countdown that indicates the remaining time to complete the Hall. If it is zero, game is over and player loses the game

Archer: A type of monster that appears in the Hall and shoots an arrow every second. Needs to be near the hero to shoot him/her

Fighter: A type of monster that appears in the Hall and tries to stab the Hero with a dagger. Moves in random directions. Needs to be next to the hero to stab him/her with a dagger. Follows the Luring gem if one is present in the Hall

Wizard: A type of monster that appears in the Hall and teleports the rune every 5 seconds.
Cannot move and stays where it appears

Life: Total lives of the player. If it reaches 0, the game ends and the Player loses the game.