Course: IT114-002-S2025
Assignment: IT114 Milestone 2 - Hangman
Student: Hitarth P. (hp627)
Status: Submitted | Worksheet Progress: 96%
Potential Grade: 9.63/10.00 (96.30%)
Received Grade: 0.00/10.00 (0.00%)
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-002-S2025/it114-milestone-2-hangman/grading/hp627

Instructions

1. Refer to Milestone2 of Hangman / Word guess
    1. Complete the features
2. Ensure all code snippets include your ucid, date, and a brief description of what the code does
3. Switch to the `Milestone2` branch
    1. `git checkout Milestone2`
    2. `git pull origin Milestone2`
4. Fill out the below worksheet as you test/demo with 3+ clients in the same session
5. Once finished, click "Submit and Export"
6. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github
    1. `git add .`
    2. `git commit -m "adding PDF"
    3. `git push origin Milestone2`
    4. On Github merge the pull request from `Milestone2` to `main`
7. Upload the same PDF to Canvas
8. Sync Local
    1. `git checkout main`
    2. `git pull origin main`

- Complete each section and task sequentially.
- Review the details and validation criteria for each task.
- Ensure subtasks are completed before the parent task.

83%

Section #1: (1 pt.) Payloads

83%

Task #1 (1 pt.) - Show Payload classes and subclasses

Combo Task:

Weight: *100%*
Objective: *Show Payload classes and subclasses*

Details:

- Reqs from the document
  - Provided Payload for applicable items that only need client id, message, and type
  - PointsPayload for syncing points of players
  - Each payload will be presented by debug output (i.e, properly override the toString() method like the lesson examples)

Image Prompt

Weight: *50%*

Details:

- Show the code related to your payloads (Payload, PointsPayload, and any new ones added)
- Each payload should have an overriden toString() method showing its internal data

Payload.java

PointsPayload.java

Missing Caption

Saved: 4/21/2025 11:49:28 AM

Text Prompt

Weight: *50%*

Details:

- Briefly explain the purpose of each payload shown in the screenshots and their properties

Your Response:

Section #2: (4 pts.) Lifecycle Events

Task #1 (0.57 pts.) - GameRoom Client Add/Remove

Combo Task:

Weight: *14.29%*
Objective: *GameRoom Client Add/Remove*

Image Prompt

Weight: *50%*

Details:

- Show the `onClientAdded()` code
- Show the `onClientRemoved()` code

Room.java

Saved: 4/21/2025 11:59:43 AM

Text Prompt

Weight: *50%*

Details:

- Briefly note the actions that happen in `onClientAdded()` (app data should at least be synchronized to the joining user)
- Briefly note the actions that happen in `onClientRemoved()` (at least should handle logic for an empty session)

Your Response:

onClientAdded(): The server enters the client to the room before sharing their arrival message to all users and pushes current room information to the new user. onClientRemoved(): In the process of leaving the room the client gets transferred out first while notifying others before executing a check to determine if the room needs resetting or should be closed.

Saved: 4/21/2025 11:59:43 AM

100%

Task #2 (0.57 pts.) - GameRoom Session Start

Combo Task:

Weight: *14.29%*
Objective: *GameRoom Session Start*

Details:

- Reqs from document
  - GameRoom loads the word list into memory from a text file for later use
  - First round is triggered

- Reset/set initial state

Image Prompt

Weight: *50%*

Details:

- Show the snippet of `onSessionStart()`

onSessionStart()

Session Start and first round trigger

Saved: 4/21/2025 12:11:12 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain the logic that occurs here (i.e., setting up initial session state for your project) and next lifecycle trigger

Your Response:

The startGame() and startRound() methods start the session by making gameInProgress true then

resetting strikes while clearing guessed letters and picking a random word of difficulty while beginning the turn timer. The game starts with a broadcast of "TURN:" + currentPlayerName triggering player action.

Saved: 4/21/2025 12:11:12 PM

100%

Task #3 (0.57 pts.) - GameRoom Round Start

Combo Task:

Weight: *14.29%*
Objective: *GameRoom Round Start*

Details:

- Reqs from Document
  - GameRoom randomly chooses and removes a word from the in-memory question list
  - The blanks are shared with all players
  - The strikes/hangman resets if it's a new round after a hangman failure or word completion
  - GameRoom round timer begins

Image Prompt

Weight: *50%*

Details:

- Show the snippet of `onRoundStart()`

private void startRound()

Saved: 4/21/2025 1:10:15 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain the logic that occurs here (i.e., setting up the round for your project)

Your Response:

The game system resets the player strikes and guessed letters while picking a new word randomly from its memory bank before it broadcasts the word with blanks to all connected users. The game initializes a round timer and prepares the first player's round while starting the time period.

Saved: 4/21/2025 1:10:15 PM

100%

Task #4 (0.57 pts.) - GameRoom Turn Start

Combo Task:

Weight: *14.29%*
Objective: *GameRoom Turn Start*

Details:

- Reqs from Document
  - Pick next Player (Initially random, then round-robin)
  - Reset turn related status to allow player to do actions
  - GameRoom Turn timer begins

Image Prompt

Weight: *50%*

Details:

- Show the snippet of `onTurnStart()`

onTurnStart()

Saved: 4/21/2025 1:28:02 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain the logic that occurs here (i.e., setting up the turn for your project)

Your Response:

As the game starts a new round it clears all previous guesses while resetting player strike counts and choosing a fresh word from its in-memory list to block word repetitions. The game selects a random player to begin while the turn timer starts and sends messages to all clients about starting a new round with the current player and masked word display for gameplay.

Saved: 4/21/2025 1:28:02 PM

Task #5 (0.57 pts.) - GameRoom Turn End

Combo Task:

Weight: *14.29%*
Objective: *GameRoom Turn End*

Details:

- Reqs from Document
  - **Condition 1:** Turn ends when Player guesses a letter or word
  - **Condition 2:** Turn ends when the Turn Timer expires
  - **Condition 3:** Turn ends when the current Player Skips their turn
  - If not the last player, trigger turn start logic
  - If the last player, trigger round end logic
  - If end condition is met, trigger session end

Image Prompt

Weight: *50%*

Details:

- Show the snippet of `onTurnEnd()`

onTurnEnd()

Saved: 4/21/2025 1:45:53 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain the logic that occurs here (i.e., setting up the turn for your project)

Your Response:

The checkGameState() method decides if the round should end through three criteria consisting of players running out of turns and the word completion and all players remaining. The method endRound() is triggered if any of the specified conditions become true. The active player sequence continues through nextTurn() until another player reaches one of the exit criteria.

Saved: 4/21/2025 1:45:53 PM

100%

Task #6 (0.57 pts.) - GameRoom Round End

Combo Task:

Weight: *14.29%*

Objective: *GameRoom Round End*

Details:

- Reqs from Document
  - **Condition 1:** Round ends when word is completed
  - **Condition 2:** Round ends when Hangman Strikes reach limit
  - **Condition 3:** Round ends when the last player's turn finishes
  - Send the in-progress scoreboard to all clients sorted by highest points to lowest
  - Trigger Round Start if end condition not met; otherwise, trigger session end

Image Prompt

Weight: *50%*

Details:

- Show the snippet of `onRoundEnd()`

onRoundEnd()

broadcastScoreboard()

Saved: 4/21/2025 1:57:33 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain the logic that occurs here (i.e., cleanup, end checks, and next lifecycle events)

Your Response:

Through endRound() the system completes cleanup operations by ceasing timer operations while broadcasting an ordered player points scoreboard to every participant. The system will activate the endGame() routine if the game has progressed to the last round. The function starts the next round by executing startRound() if the round criteria are not met.

Saved: 4/21/2025 1:57:33 PM

100%

Task #7 (0.57 pts.) - GameRoom Session End

Combo Task:

Weight: *14.29%*
Objective: *GameRoom Session End*

Details:

- Reqs from Document
  - **Condition 1:** Session ends when X rounds have passed
  - Send the final scoreboard to all clients sorted by highest points to lowest (include a game over message)
  - Reset the player data for each client server-side and client-side (do not disconnect them or move them to the lobby)
  - A new ready check will be required to start a new session

Image Prompt

Weight: *50%*

Details:

- Show the snippet of `onSessionEnd()`

onSessionEnd()

Supporting Method - Final ScoreBoard

Saved: 4/21/2025 2:04:28 PM

Text Prompt

Weight: *50%*

Details:

- Briefly explain the logic that occurs here (i.e., cleanup/reset, next lifecycle events)

Your Response:

Through the endGame() method the session concludes when a final scoreboard transfers to all clients alongside the reset of player points and ready status maintenance within their room. Following the session conclusion the gameplay enters the ready check phase which necessitates another round of player readiness before beginning the next game.

Saved: 4/21/2025 2:04:28 PM

Section #3: (4 pts.) Gameroom User Action And State

Task #1 (1.50 pts.) - Word Logic

Combo Task:

Weight: *37.5%*
Objective: *Word Logic*

Details:

- Reqs from document
  - Command: `/guess <word>`
    - [ ] GameRoom will check against the correct word
      - [ ] If it matches (exactly)
        - [ ] The guesser will receive points for each missing letter and bonus points for solving (i.e., 2x points for each empty space)
          - [ ] Points will be stored on the Player/User object
          - [ ] Sync the points value of the Player to all Clients
        - [ ] A message will be relayed that "X guessed the correct word {word} and got {points} points".
        - [ ] The round should end
      - [ ] If no matches
        - [ ] A strike will be incurred and synced to all Clients
        - [ ] A message will be relayed that "X guessed {word} and it was wrong"
        - [ ] Their turn should end

Image Prompt

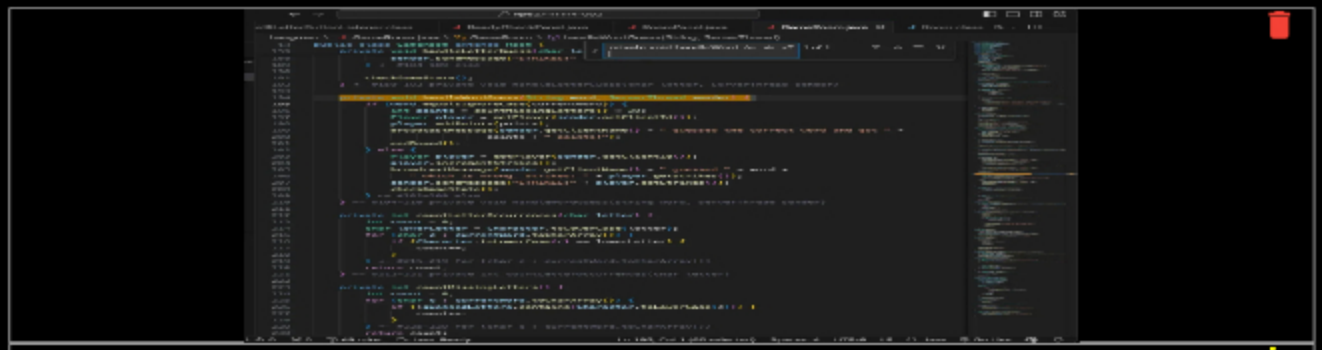
Weight: *50%*

Details:

- Show the code snippets of the following, and clearly caption each screenshot
- Show the Client processing of this command (process client command)
- Show the ServerThread processing of this command (process method)
- Show the GameRoom handling of this command (handle method)
- Show the sending/syncing of the results of this command to users (send/sync method)
- Show the ServerThread receiving this data (send method)
- Show the Client receiving this data (process method)

/guess

Command Handling



Word Guess Logic -handleWordGuess()

broadcastMessage()

100%

Task #2 (1.50 pts.) - Letter Logic

Combo Task:

Weight: *37.5%*
Objective: *Letter Logic*

Details:

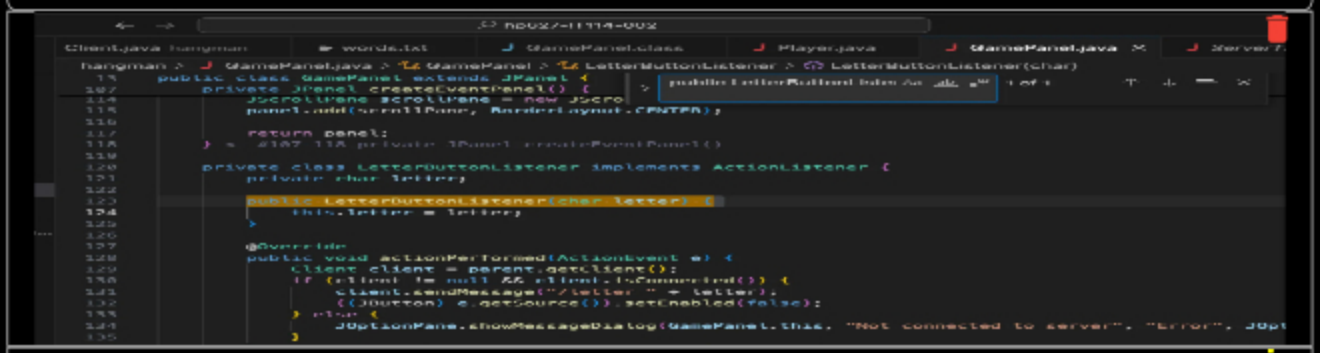
- Reqs from document
  - Command: `/letter <letter>`
    - [ ] GameRoom will check the occurrences of the letter that haven't been guessed this round
      - [ ] If any matches
        - [ ] The guesser will receive points for each slot the letter appears in if it has not been previously guessed for this word
          - [ ] I.e., points times the number of matched letters
          - [ ] Points will be stored on the Player/User object
          - [ ] Sync the points value of the Player to all Clients
        - [ ] A message will be relayed "X guessed {letter} and there were {number} {letter}'s which yielded {points} points
        - [ ] Their turn should end
      - [ ] If no matches
        - [ ] A strike will be incurred and synced to all Clients
        - [ ] A message will be relayed "X guess {letter} letter, which isn't in the word"
        - [ ] If strikes reach the limit the round ends
        - [ ] If strikes don't reach the limit the current Player's turn ends

Image Prompt

Weight: *50%*

Details:

- Show the code snippets of the following, and clearly caption each screenshot
- Show the Client processing of this command (process client command)
- Show the ServerThread processing of this command (process method)
- Show the GameRoom handling of this command (handle method)
- Show the sending/syncing of the results of this command to users (send/sync method)
- Show the ServerThread receiving this data (send method)
- Show the Client receiving this data (process method)



/letter

handleCommand()

handleLetterGuess()

Share guessed letters

Task #3 (0.50 pts.) - Skip Logic

Combo Task:

Weight: *12.5%*
Objective: *Skip Logic*

Details:

- Reqs from document
  - Command: `/skip`
    - Skips the player's turn if it's their turn (ends their turn)

Image Prompt

Weight: *50%*

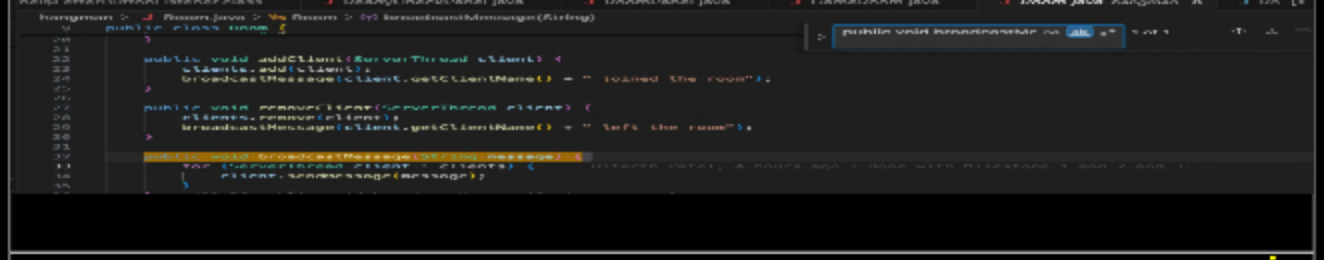
Details:

- Show the code snippets of the following, and clearly caption each screenshot
- Show the Client processing of this command (process client command)
- Show the ServerThread processing of this command (process method)
- Show the GameRoom handling of this command (handle method)
- Show the sending/syncing of the results of this command to users (send/sync method)
- Show the ServerThread receiving this data (send method)
- Show the Client receiving this data (process method)

/skip

/skip

skipTurn()



broadcastMessage()

100% Task #4 (0.50 pts.) - Game Cycle Demo

Image Prompt

Weight: *12.5%*
Objective: *Game Cycle Demo*

Details:

- Show examples from the terminal of a full session demonstrating each command and progress output
- This includes showing blanks, correct letters, incorrect letters, correct words, incorrect words, skips, scores, and scoreboards, etc
- Ensure at least 3 Clients and the Server are shown
- Clearly caption screenshots

Ready

Section #4: (1 pt.) Misc

Task #1 (0.33 pts.) - Github Details

Combo Task:

Weight: *33.33%*
Objective: *Github Details*

Image Prompt

Weight: *60%*

Details:
From the Commits tab of the Pull Request screenshot the commit history

Missing Caption

Saved: 4/21/2025 10:55:53 PM

Url Prompt

Weight: *40%*

Details:
Include the link to the Pull Request (should end in `/pull/#`)

URL #1
https://github.com/hitarthpat/hp627-

URL
https://github.com/hitarthpat/hp62

IT114-003/main...Milestone2

Saved: 4/21/2025 10:55:53 PM

100%

Task #2 (0.33 pts.) - WakaTime - Activity

Image Prompt

Weight: *33.33%*
Objective: *WakaTime - Activity*

Details:

- Visit the WakaTime.com Dashboard
- Click `Projects` and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary

Waka Time

100%

Task #3 (0.33 pts.) - Reflection

Weight: *33.33%*
Objective: *Reflection*

Sub-Tasks:

100% Task #1 (0.33 pts.) - What did you learn?

Text Prompt

Weight: *33.33%*
Objective: *What did you learn?*

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

I developed the design and implementation of a multiplayer Hangman game through Java programming with a client-server architecture system. Development of the project required the creation of GameRoom with Player components alongside three GUI panels which included ConnectionPanel, ReadyCheckPanel along with GamePanel. Altogether I gained experience with client management as well as multiplayer state synchronization and real-time user interaction control through socket programming and multithreading methods. Through this project I further strengthened my abilities to design objects in an object-oriented fashion while programming events and developing interactive interfaces.

Saved: 4/21/2025 11:00:14 PM

100% Task #2 (0.33 pts.) - What was the easiest part of the assignr

Text Prompt

Weight: *33.33%*
Objective: *What was the easiest part of the assignment?*

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

The most easiest aspect of this assignment involved developing the basic Java Swing user interface with CardLayout implementation. The execution of ConnectionPanel and RoomPanel came easily because these panels contained primarily form elements with buttons and basic event handlers. The process of linking panels through HangmanUI along with view switching became

easier after developing the fundamental layout.

Saved: 4/21/2025 11:01:22 PM

100%

Task #3 (0.33 pts.) - What was the hardest part of the assign

Text Prompt

Weight: *33.33%*
Objective: *What was the hardest part of the assignment?*

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

The greatest difficulty during this assignment involved the control of game logic while maintaining simultaneous client synchronization within the GameRoom class.Development of the GameRoom class involved thorough planning along with extensive testing to manage player turn control and strike recording and handle situations when players leave or switch modes. The most difficult aspect was maintaining consistent state updates across all connected clients while avoiding race conditions but ensuring state consistency.

Saved: 4/21/2025 11:05:16 PM