Homework 9

Asynchronous Controller and Views

Northeastern University

CS5004 – Object Oriented Design

Team: TBH

Chia-Hsiang Hsu Tai

Bhoomika Gupta

Harrison Pham

UML Class Diagram

The general structure of our UML class diagram has remained the same, however since we added the GUI, there were a lot of new elements that must also be represented in the class diagram. From the model side, the relationship between the classes has remained the same with a few minor changes in the implementation details to account for image handling in the GUI. On the controller side, we introduced a new ViewController class and the IViewController interface to manage GUI interactions and the execution of commands in the model. We also added a new set of classes that represent the components needed to create the GameView. To support multiple game modes, we implemented the IEventHandler interface for text-based games and an IGuiEventHandler for GUI based games, which allowed us to create various input and output configurations to pass to the controller. This approach allows us to keep the Model-View-Controller dynamic intact if we ever wanted to introduce a new type of input and output configuration.

The UML has grown significantly since Homework 8. We were unable to fit the whole class diagram into a single page while making sure it was readable and clear. We have attached a separate PDF file in the submission with the UML Class Diagram.

Written Scenarios

Written Scenario #1

The player starts the game in Hallway 1. The room description says that there is a notebook in this room. The player takes the notebook and places it in their inventory. They then attempt to move south, but unfortunately there is no path, so they remain in Hallway 1. The player then chooses to go north and enters Hallway 2, where there is a key and hair clippers. They take both items. The player decides to go north again, but there is a locked door. The player checks their inventory, and they see a key. They use the key to unlock the door. Success! The key turns and unlocks the door. Now that the door is unlocked, the player then chooses to examine the lock.

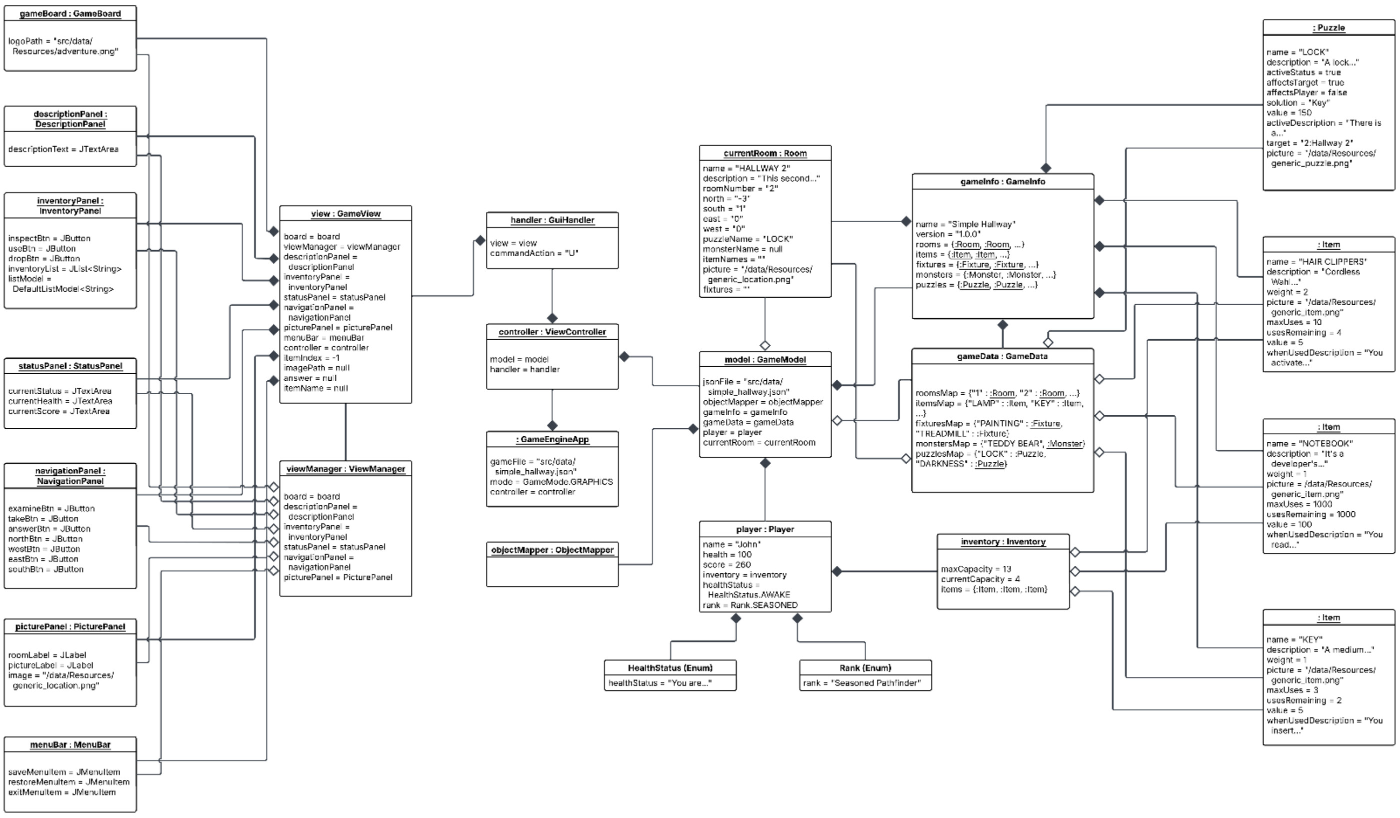
Written Scenario #2

The player is currently in Hallway 2 and decides to head north. They enter Hallway 3 and encounter a Teddy Bear monster. The monster attacks the player and deals 5 damage points to the player. The player then checks their inventory to see what items they have to fight the monster. The items in the player's inventory are displayed and the player sees they have a notebook, key, and hair clippers. The Teddy Bear notices the movements of the player and attacks the player again and their health is reduced by 5 points. The player then uses the hair clippers to defeat the Teddy Bear and their overall score increases. The player then decides to head north and enter the Exercise Room.

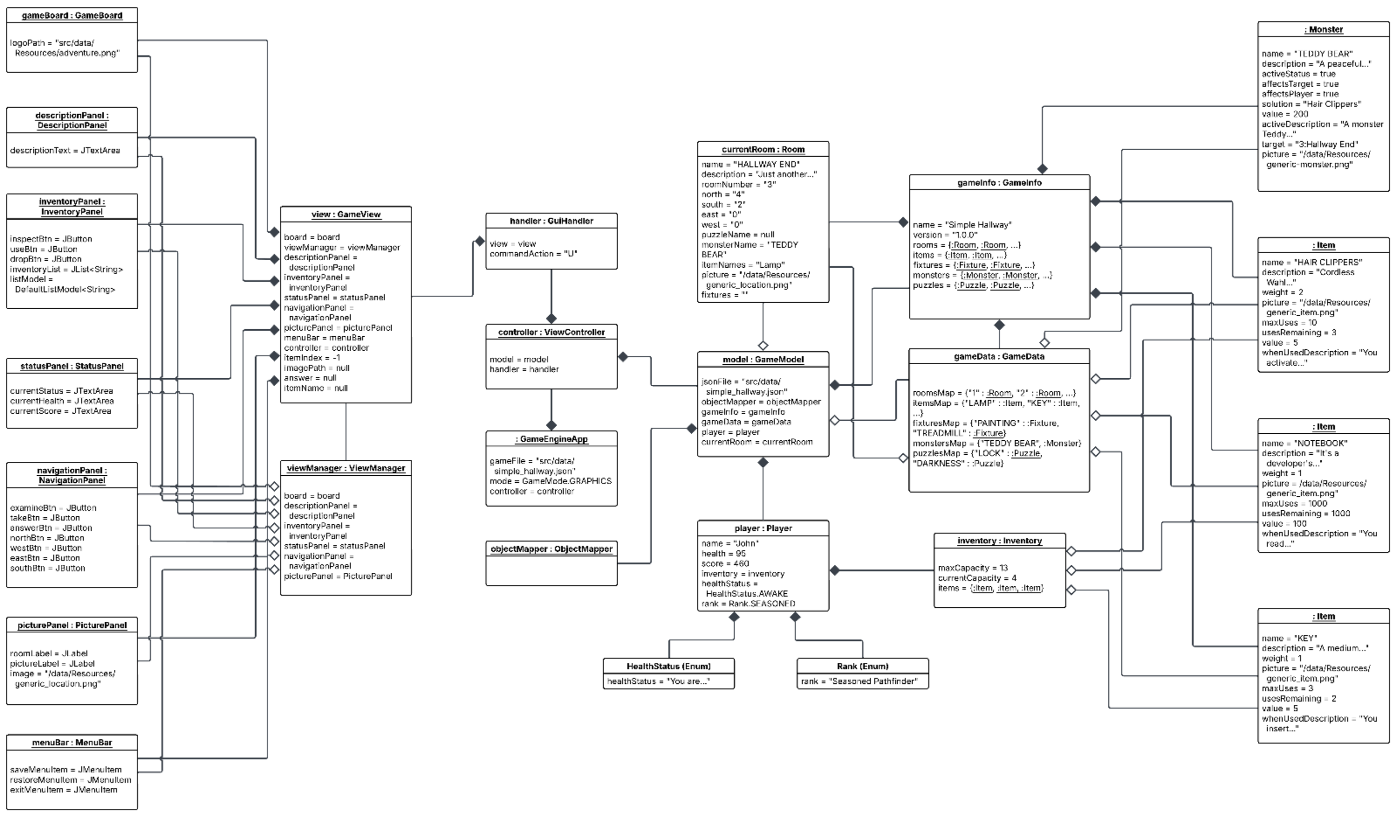
Object Diagrams

We have attached the object diagram to our submission as separate PDFs for a clearer image of the object diagrams.

The object diagram below is based on **Written Scenario #1**. It is a concrete representation of the game's state after the player solves the "locked door" puzzle by using the "key" item and proceeds to Hallway 2. The player's score is increased by the value of the puzzle, and the remaining uses for the "key" item is reduced by one.


gameBoard : GameBoard
logoPath = "src/data/Resources/adventure.png"
descriptionPanel : DescriptionPanel
descriptionText = JTextArea
inventoryPanel : InventoryPanel
inspectBtn = JButton
useBtn = JButton
dropBtn = JButton
inventoryList = JList<String>
listModel = DefaultListModel<String>
statusPanel : StatusPanel
currentStatus = JTextArea
currentHealth = JTextArea
currentScore = JTextArea
navigationPanel : NavigationPanel
examineBtn = JButton
takeBtn = JButton
answerBtn = JButton
northBtn = JButton
westBtn = JButton
eastBtn = JButton
southBtn = JButton
picturePanel : PicturePanel
roomLabel = JLabel
pictureLabel = JLabel
image = "/data/Resources/generic_location.png"
menuBar : MenuBar
saveMenuItem = JMenuItem
restoreMenuItem = JMenuItem
exitMenuItem = JMenuItem
view : GameView
board = board
viewManager = viewManager
descriptionPanel = descriptionPanel
inventoryPanel = inventoryPanel
statusPanel = statusPanel
navigationPanel = navigationPanel
picturePanel = picturePanel
menuBar = menuBar
controller = controller
itemIndex = -1
imagePath = null
answer = null
itemName = null
viewManager : ViewManager
board = board
descriptionPanel = descriptionPanel
inventoryPanel = inventoryPanel
statusPanel = statusPanel
navigationPanel = navigationPanel
picturePanel = PicturePanel
handler : GuiHandler
view = view
commandAction = "U"
controller : ViewController
model = model
handler = handler
: GameEngineApp
gameFile = "src/data/simple_hallway.json"
mode = GameMode.GRAPHICS
controller = controller
objectMapper : ObjectMapper
currentRoom : Room
name = "HALLWAY 2"
description = "This second..."
roomNumber = "2"
north = "-3"
south = "1"
east = "0"
west = "0"
puzzleName = "LOCK"
monsterName = null
itemNames = ""
picture = "/data/Resources/generic_location.png"
fixtures = ""
model : GameModel
jsonFile = "src/data/simple_hallway.json"
objectMapper = objectMapper
gameInfo = gameInfo
gameData = gameData
player = player
currentRoom = currentRoom
player : Player
name = "John"
health = 100
score = 260
inventory = inventory
healthStatus = HealthStatus.AWAKE
rank = Rank.SEASONED
HealthStatus (Enum)
healthStatus = "You are..."
Rank (Enum)
rank = "Seasoned Pathfinder"
gameInfo : GameInfo
name = "Simple Hallway"
version = "1.0.0"
rooms = {:Room, :Room, ...}
items = {:Item, :Item, ...}
fixtures = {:Fixture, :Fixture, ...}
monsters = {:Monster, :Monster, ...}
puzzles = {:Puzzle, :Puzzle, ...}
gameData : GameData
roomsMap = {"1" : :Room, "2" : :Room, ...}
itemsMap = {"LAMP" : :Item, "KEY" : :Item, ...}
fixturesMap = {"PAINTING" : :Fixture, "TREADMILL" : :Fixture}
monstersMap = {"TEDDY BEAR", :Monster}
puzzlesMap = {"LOCK" : :Puzzle, "DARKNESS" : :Puzzle}
inventory : Inventory
maxCapacity = 13
currentCapacity = 4
items = {:Item, :Item, :Item}
: Puzzle
name = "LOCK"
description = "A lock..."
activeStatus = true
affectsTarget = true
affectsPlayer = false
solution = "Key"
value = 150
activeDescription = "There is a..."
target = "2:Hallway 2"
picture = "/data/Resources/generic_puzzle.png"
: Item
name = "HAIR CLIPPERS"
description = "Cordless Wahl..."
weight = 2
picture = "/data/Resources/generic_item.png"
maxUses = 10
usesRemaining = 4
value = 5
whenUsedDescription = "You activate..."
: Item
name = "NOTEBOOK"
description = "It's a developer's..."
weight = 1
picture = /data/Resources/generic_item.png"
maxUses = 1000
usesRemaining = 1000
value = 100
whenUsedDescription = "You read..."
: Item
name = "KEY"
description = "A medium..."
weight = 1
picture = "/data/Resources/generic_item.png"
maxUses = 3
usesRemaining = 2
value = 5
whenUsedDescription = "You insert..."

The object diagram below is based on **Written Scenario #2**. It is a concrete representation of the game's state when the player enters Hallway End, faces the "Teddy Bear" monster and defeats it with the "hair clippers" item. The player's score is increased by the value of the monster, and the remaining uses for the "hair clippers" item is reduced by 1.

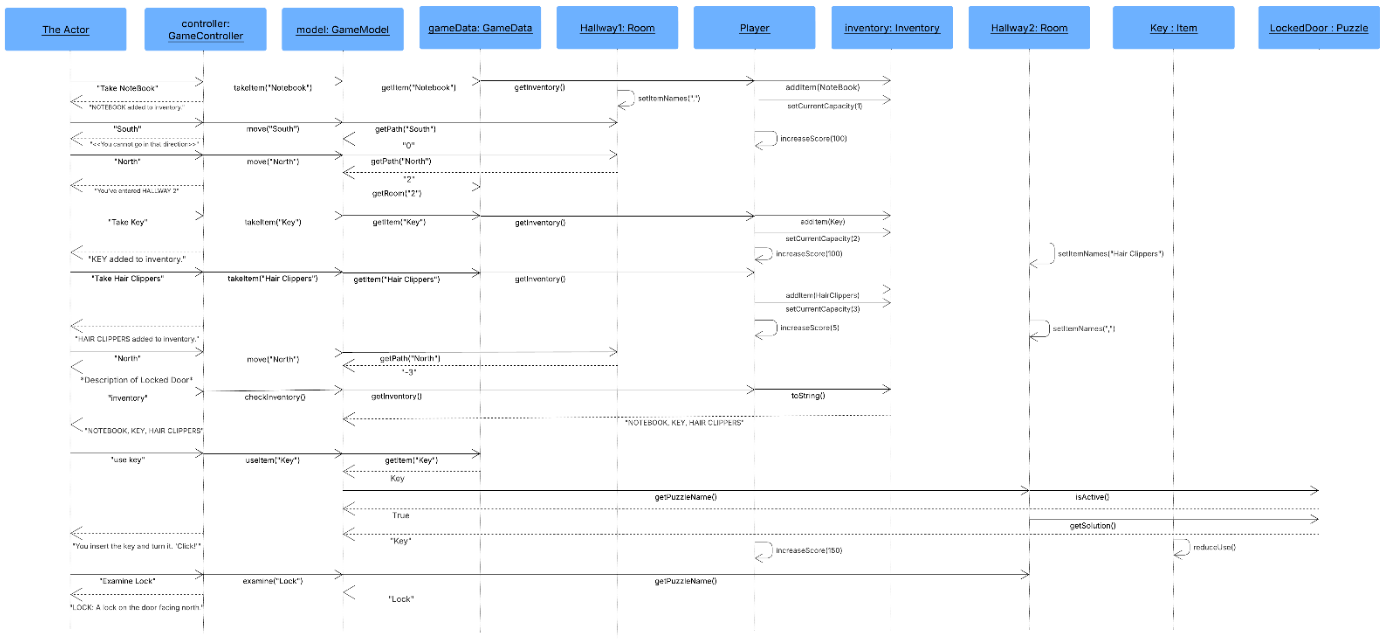

gameBoard : GameBoard
logoPath = "src/data/Resources/adventure.png"
descriptionPanel : DescriptionPanel
descriptionText = JTextArea
inventoryPanel : InventoryPanel
inspectBtn = JButton
useBtn = JButton
dropBtn = JButton
inventoryList = JList<String>
listModel = DefaultListModel<String>
statusPanel : StatusPanel
currentStatus = JTextArea
currentHealth = JTextArea
currentScore = JTextArea
navigationPanel : NavigationPanel
examineBtn = JButton
takeBtn = JButton
answerBtn = JButton
northBtn = JButton
westBtn = JButton
eastBtn = JButton
southBtn = JButton
picturePanel : PicturePanel
roomLabel = JLabel
pictureLabel = JLabel
image = "/data/Resources/generic_location.png"
menuBar : MenuBar
saveMenuItem = JMenuItem
restoreMenuItem = JMenuItem
exitMenuItem = JMenuItem
view : GameView
board = board
viewManager = viewManager
descriptionPanel = descriptionPanel
inventoryPanel = inventoryPanel
statusPanel = statusPanel
navigationPanel = navigationPanel
picturePanel = picturePanel
menuBar = menuBar
controller = controller
itemIndex = -1
imagePath = null
answer = null
itemName = null
viewManager : ViewManager
board = board
descriptionPanel = descriptionPanel
inventoryPanel = inventoryPanel
statusPanel = statusPanel
navigationPanel = navigationPanel
picturePanel = PicturePanel
handler : GuiHandler
view = view
commandAction = "U"
controller : ViewController
model = model
handler = handler
: GameEngineApp
gameFile = "src/data/simple_hallway.json"
mode = GameMode.GRAPHICS
controller = controller
objectMapper : ObjectMapper
currentRoom : Room
name = "HALLWAY END"
description = "Just another..."
roomNumber = "3"
north = "4"
south = "2"
east = "0"
west = "0"
puzzleName = null
monsterName = "TEDDY BEAR"
itemNames = "Lamp"
picture = "/data/Resources/generic_location.png"
fixtures = ""
model : GameModel
jsonFile = "src/data/simple_hallway.json"
objectMapper = objectMapper
gameInfo = gameInfo
gameData = gameData
player = player
currentRoom = currentRoom
player : Player
name = "John"
health = 95
score = 460
inventory = inventory
healthStatus = HealthStatus.AWAKE
rank = Rank.SEASONED
HealthStatus (Enum)
healthStatus = "You are..."
Rank (Enum)
rank = "Seasoned Pathfinder"
gameInfo : GameInfo
name = "Simple Hallway"
version = "1.0.0"
rooms = {:Room, :Room, ...}
items = {:Item, :Item, ...}
fixtures = {:Fixture, :Fixture, ...}
monsters = {:Monster, :Monster, ...}
puzzles = {:Puzzle, :Puzzle, ...}
gameData : GameData
roomsMap = {"1" : :Room, "2" : :Room, ...}
itemsMap = {"LAMP" : :Item, "KEY" : :Item, ...}
fixturesMap = {"PAINTING" : :Fixture, "TREADMILL" : :Fixture}
monstersMap = {"TEDDY BEAR", :Monster}
puzzlesMap = {"LOCK" : :Puzzle, "DARKNESS" : :Puzzle}
inventory : Inventory
maxCapacity = 13
currentCapacity = 4
items = {:Item, :Item, :Item}
: Monster
name = "TEDDY BEAR"
description = "A peaceful..."
activeStatus = true
affectsTarget = true
affectsPlayer = true
solution = "Hair Clippers"
value = 200
activeDescription = "A monster Teddy..."
target = "3:Hallway End"
picture = "/data/Resources/generic-monster.png"
: Item
name = "HAIR CLIPPERS"
description = "Cordless Wahl..."
weight = 2
picture = "/data/Resources/generic_item.png"
maxUses = 10
usesRemaining = 3
value = 5
whenUsedDescription = "You activate..."
: Item
name = "NOTEBOOK"
description = "It's a developer's..."
weight = 1
picture = /data/Resources/generic_item.png"
maxUses = 1000
usesRemaining = 1000
value = 100
whenUsedDescription = "You read..."
: Item
name = "KEY"
description = "A medium..."
weight = 1
picture = "/data/Resources/generic_item.png"
maxUses = 3
usesRemaining = 2
value = 5
whenUsedDescription = "You insert..."

Sequence Diagrams

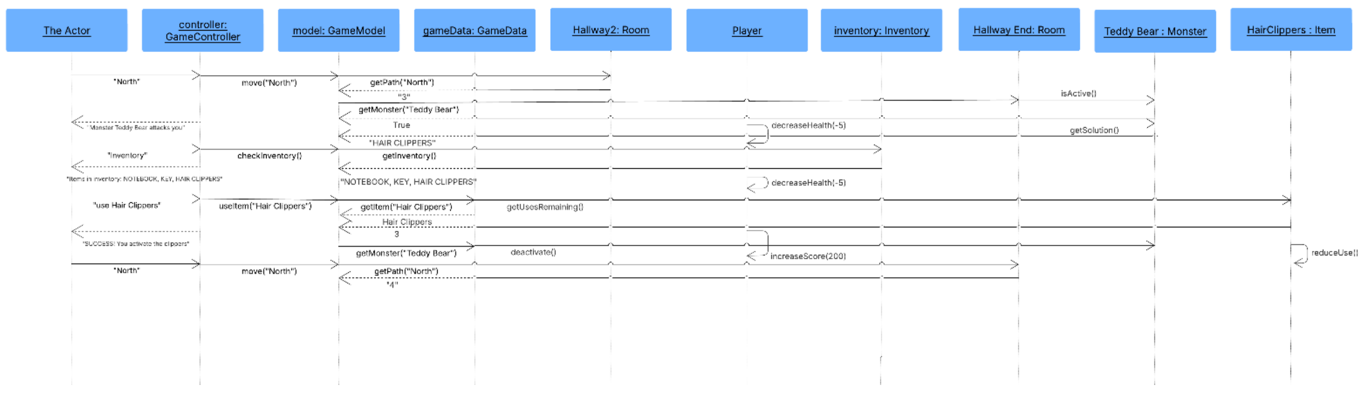
We have attached as a separate PDF file to our submission for a clearer view of all sequence diagrams. The new sequence diagram at the implementation level was too long to fit in the page so it will be in the separate PDF.

This ***new sequence diagram*** depicts the implementation level flow of events and messages. It shows the interaction between the View when the game is played in graphics mode. This sequence diagram shows the flow of the game and focuses specifically on the view and how the specific panels are updated as the user plays the game through the view. The view is separate and uses the controller and the event handler to manage and update the information displayed by the View. The View also has a ViewManager class that sets up all the panels and updates them as users interact with the view. This separation helps the View to focus on sending and receiving information from the controller.

The sequence diagram below depicts **Written Scenario #1**. In this scenario, the player attempts to move until they successfully move to a hallway that contains an obstacle. The obstacle is the "locked door" puzzle, which the player solves by using a "key" item found in their inventory.



The sequence diagram below depicts **Written Scenario #2**. In this scenario, the player moves to a new room that contains an obstacle. The obstacle is a “Teddy Bear” monster which attacks the player upon entering the room, decreasing their health. The player puts the monster to sleep by using an “hair clippers” item from their inventory.



Assumptions

As we reach the conclusion of this Adventure Game, our design implementation follows all the rules specified in the assignments and gameplay videos. However, there are still some assumptions that are unique to our game and can be found below:

Player Assumptions

- A player's score is determined through the sum of the following points:
    - Points from defeating a monster or solving a puzzle.
    - The sum of the item points currently in inventory.
- The player's rank thresholds are as follows:
    - Novice (starting rank), Seasoned (250+ points), Master (400+ points), Legendary (700+ points)

Item Assumptions

- Items that can no longer be used remain in the player's inventory until they are dropped by the player. Even if they can no longer be used, their presence in the player's inventory means that the item's points are counted towards the player's score.
- Items used to solve puzzles or defeat monsters also remain in the player's inventory.
- Items can be used on game entities (monsters, puzzles, etc.), but not on the player.

Command Assumptions

- We assume that when dealing with a puzzle, if the solution to a puzzle is an item, you use the "USE" command, whereas if the solution is an answer, we use the "ANSWER" command.

General Game Assumption

- We assume that for any game played in batch mode that gets input from a file, the file used has a command that terminates the game at some point. That means, that all source files should have a "QUIT" command to terminate the game.

References

- https://en.wikipedia.org/wiki/Adventureland_(video_game)
- https://iplayif.com/?story=http%3A%2F%2Fwww.ifarchive.org%2Fif-archive%2Fgames%2Fzcode%2FAdventureland.z5
- https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=c8fb2af239a3dca6c7d212a09bcf14769f12e245
- https://www.cs.cornell.edu/courses/cs5150/2014fa/slides/D2-use-cases.pdf
- https://refactoring.guru/design-patterns/command
- https://www.cs.rutgers.edu/courses/111/classes/fall_2011_tjang/texts/notes-java/GUI/layouts/20borderlayout.html
- https://stackoverflow.com/questions/68486793/how-do-i-stretch-a-component-over-multiple-columns-using-gridbaglayout
- https://docs.oracle.com/javase/8/docs/api/java/awt/Font.html
- https://www.youtube.com/watch?v=Kmgo00avvEw
- https://docs.oracle.com/javase/tutorial/uiswing/components/list.html
- https://stackoverflow.com/questions/21913408/reading-system-out-java-test

Shout Outs

- Big thank you to Soni for giving us ideas on how to incorporate our extensive UML into our documentation. Our UML Class diagram has grown extensively as the homework has progressed. We're sorry if it's very confusing! Also, she helped clear up some confusion we had surrounding how to create the sequence diagram on an implementation level, rather than on the analysis level.