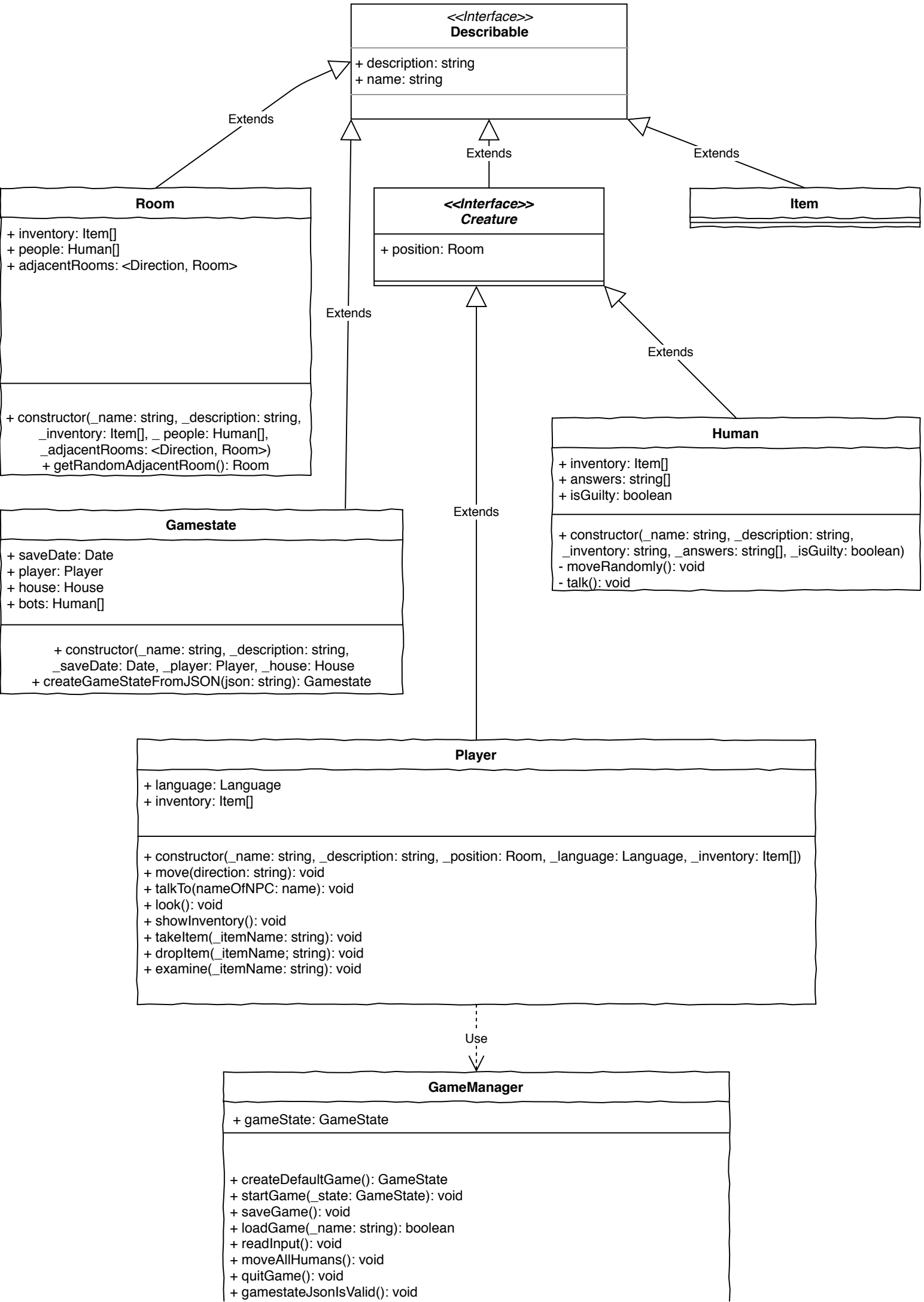

<<Interface>>
Describable
+ description: string
+ name: string
Extends
Extends
Extends
Room
+ inventory: Item[]
+ people: Human[]
+ adjacentRooms: <Direction, Room>
+ constructor(_name: string, _description: string, _inventory: Item[], _ people: Human[], _adjacentRooms: <Direction, Room>)
+ getRandomAdjacentRoom(): Room
<<Interface>>
Creature
+ position: Room
Item
Extends
Extends
Human
+ inventory: Item[]
+ answers: string[]
+ isGuilty: boolean
+ constructor(_name: string, _description: string, _inventory: string, _answers: string[], _isGuilty: boolean)
- moveRandomly(): void
- talk(): void
Gamestate
+ saveDate: Date
+ player: Player
+ house: House
+ bots: Human[]
+ constructor(_name: string, _description: string, _saveDate: Date, _player: Player, _house: House
+ createGameStateFromJSON(json: string): Gamestate
Extends
Player
+ language: Language
+ inventory: Item[]
+ constructor(_name: string, _description: string, _position: Room, _language: Language, _inventory: Item[])
+ move(direction: string): void
+ talkTo(nameOfNPC: name): void
+ look(): void
+ showInventory(): void
+ takeItem(_itemName: string): void
+ dropItem(_itemName; string): void
+ examine(_itemName: string): void
Use
GameManager
+ gameState: GameState
+ createDefaultGame(): GameState
+ startGame(_state: GameState): void
+ saveGame(): void
+ loadGame(_name: string): boolean
+ readInput(): void
+ moveAllHumans(): void
+ quitGame(): void
+ gamestateJsonIsValid(): void

+ guessNPC(_name: string): void
+ printToConsole(_message: string): void

<<enum>>
Direction

+ North
+ East
+ West
+ South

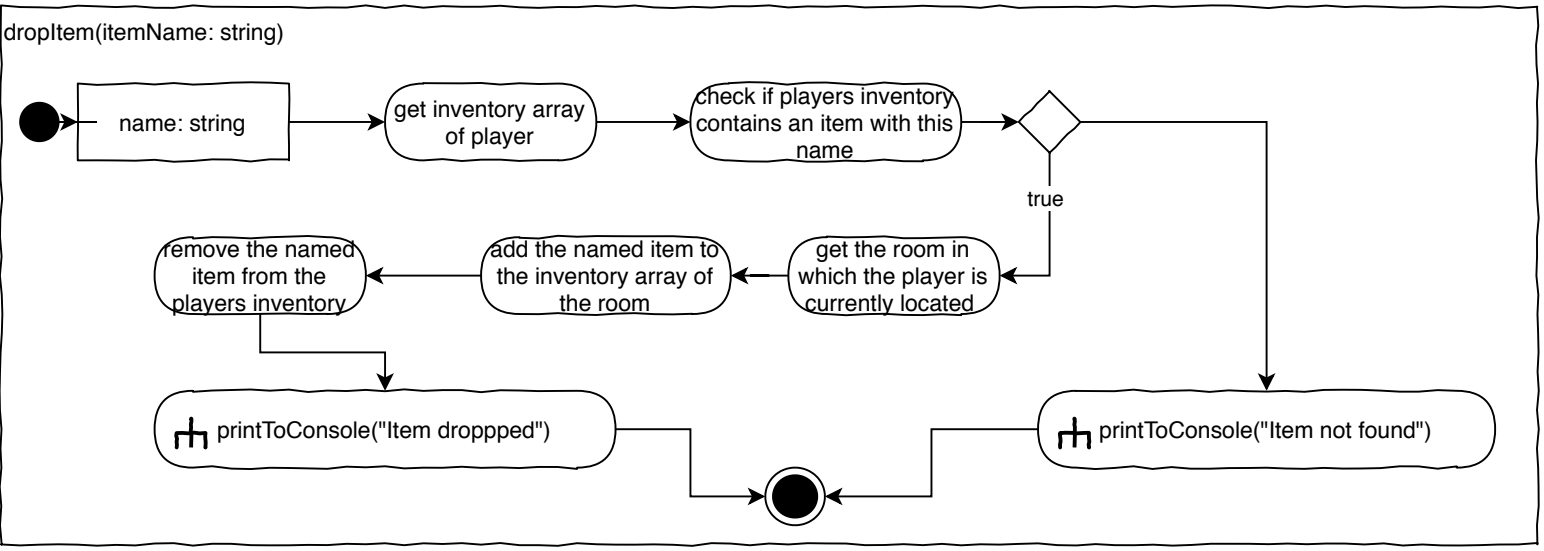
<<enum>>
Language

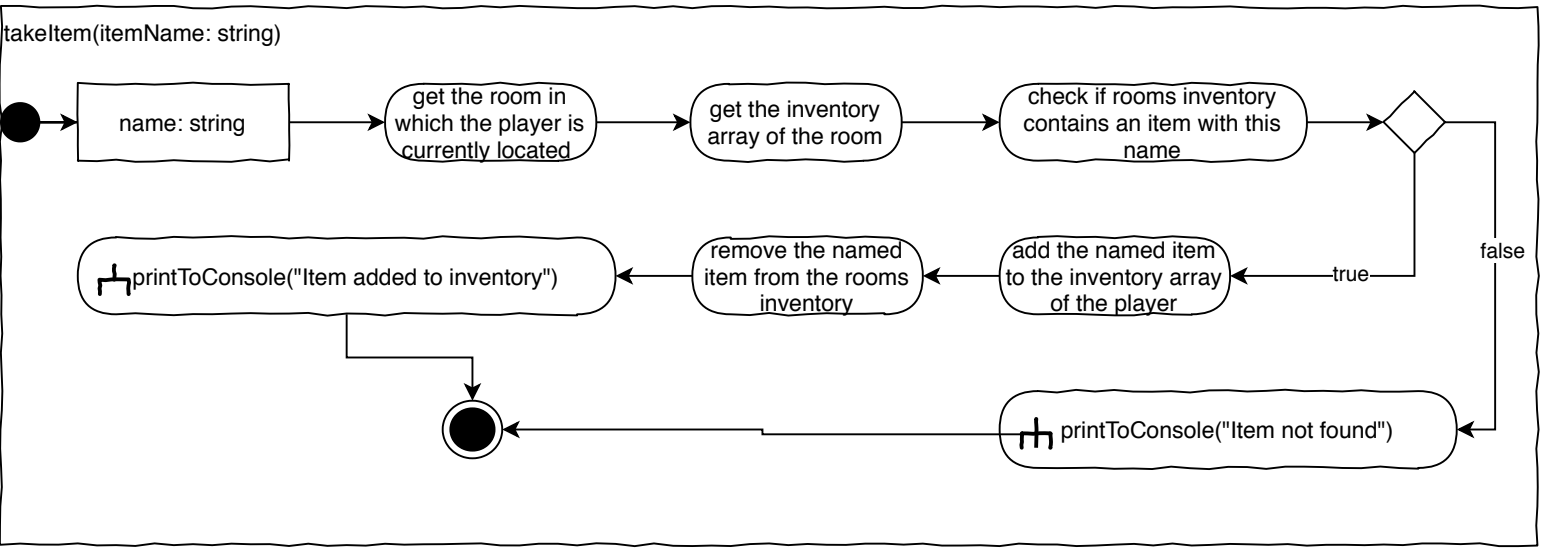
+ English
+ German
+ Frensh

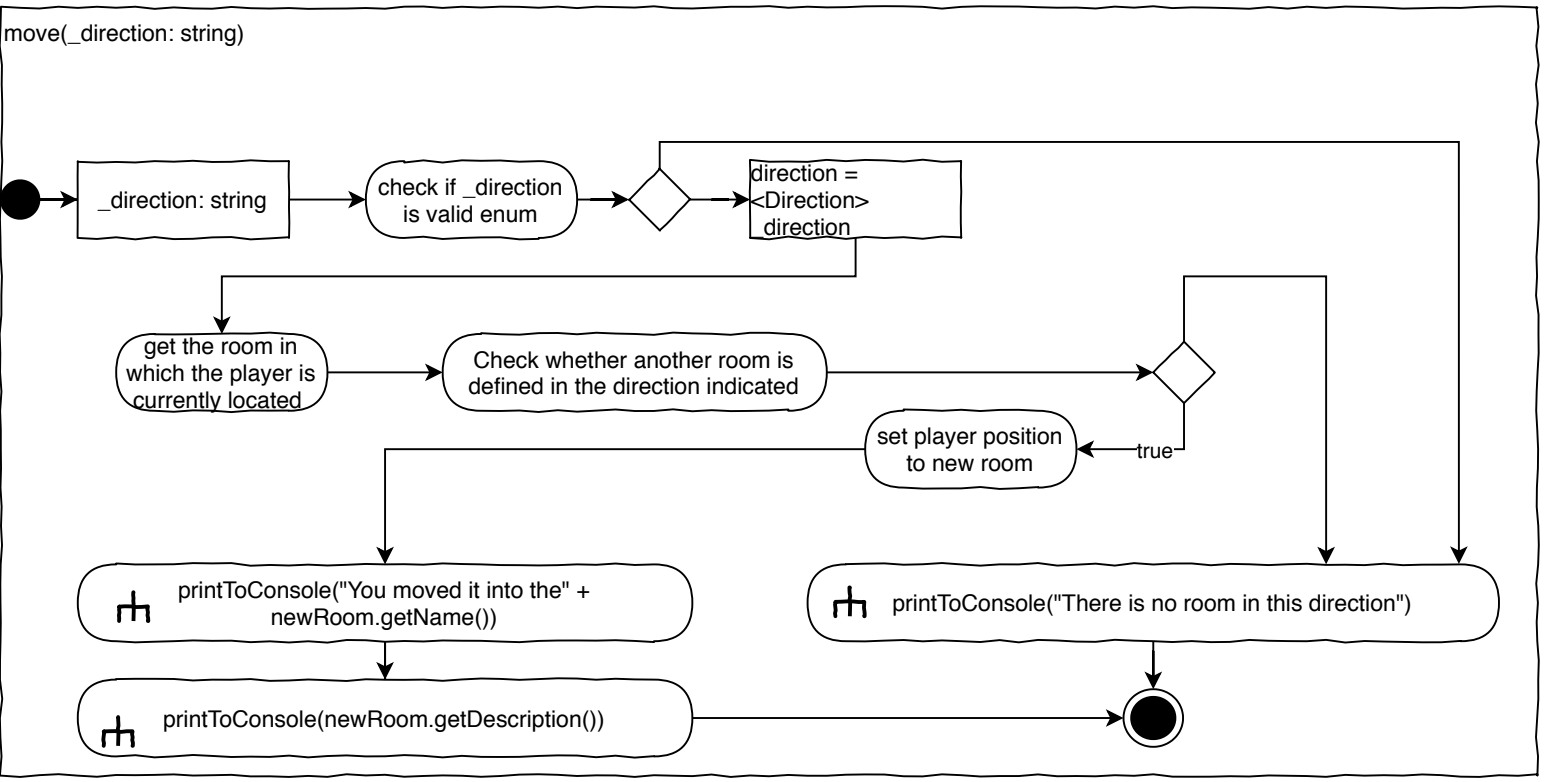
House

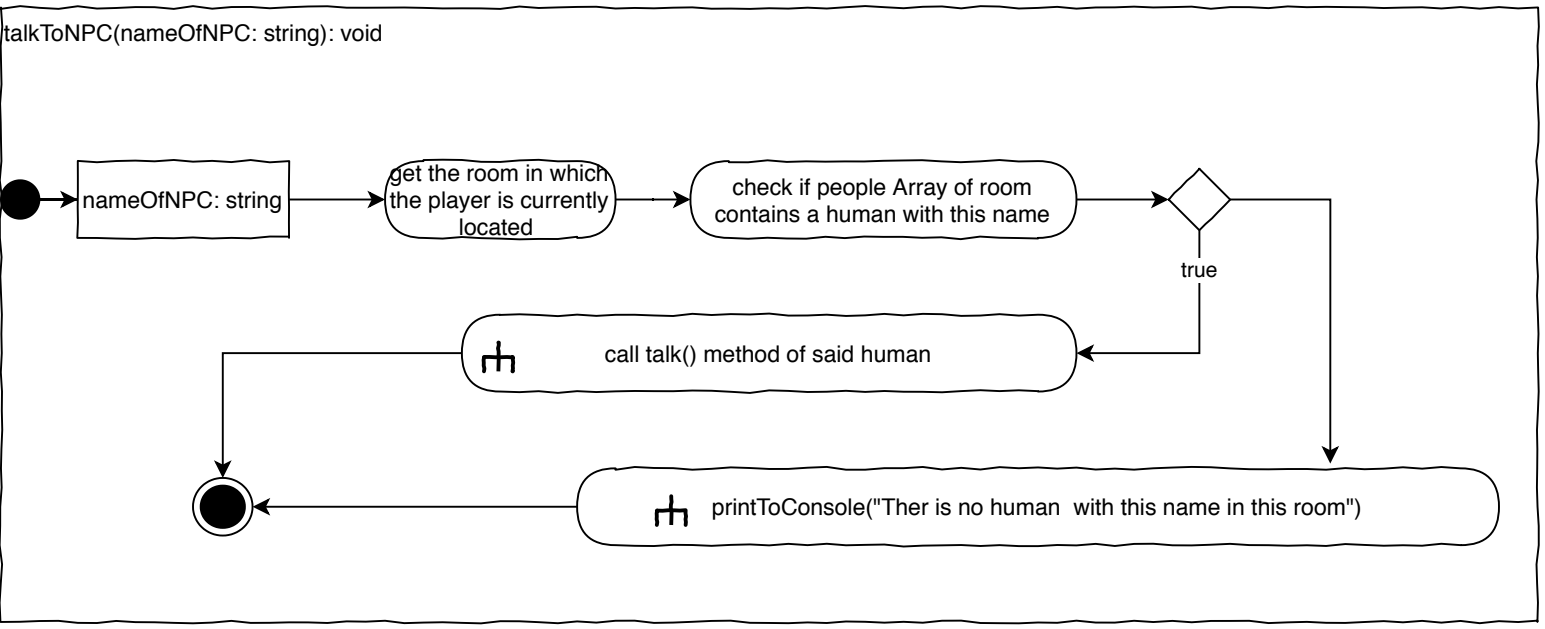
+ name: string
+ rooms: Room[]

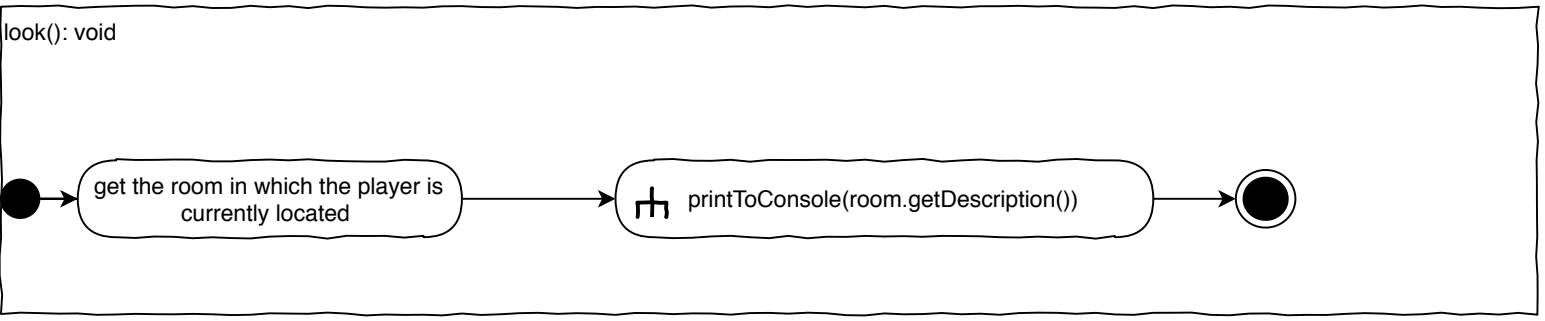
+ constructor(_name: string, _rooms: Room[])

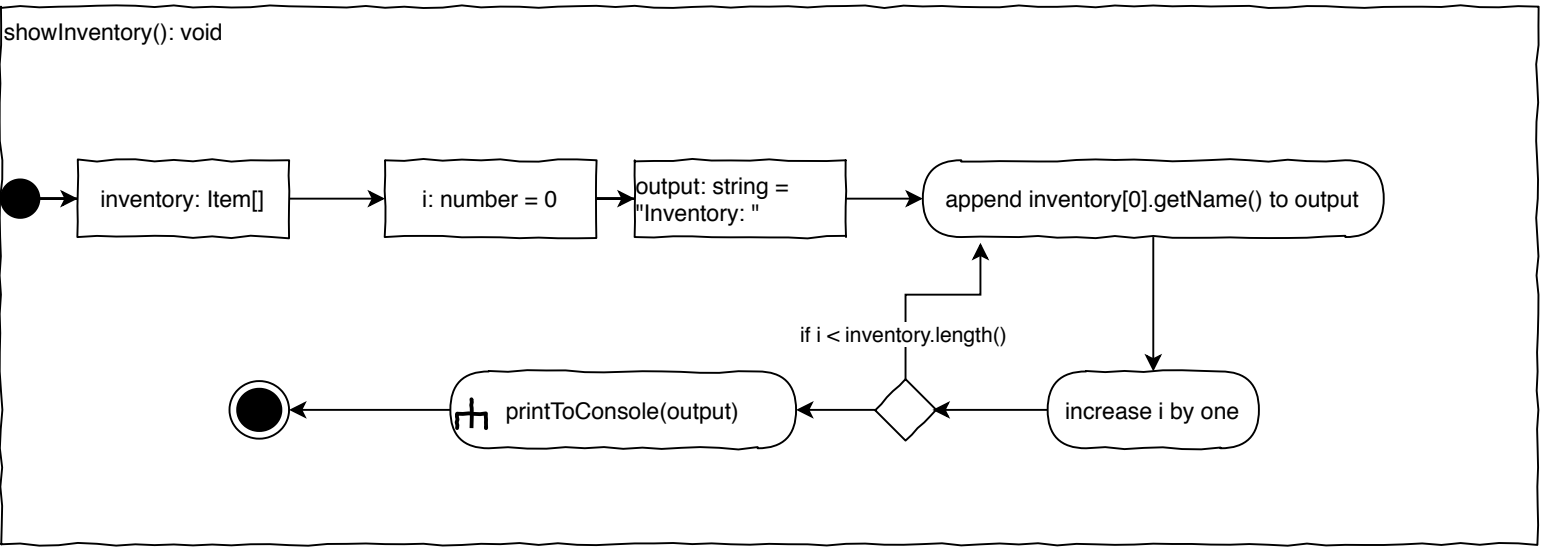

dropItem(itemName: string)
name: string
get inventory array of player
check if players inventory contains an item with this name
true
get the room in which the player is currently located
add the named item to the inventory array of the room
remove the named item from the players inventory
printToConsole("Item droppped")
printToConsole("Item not found")

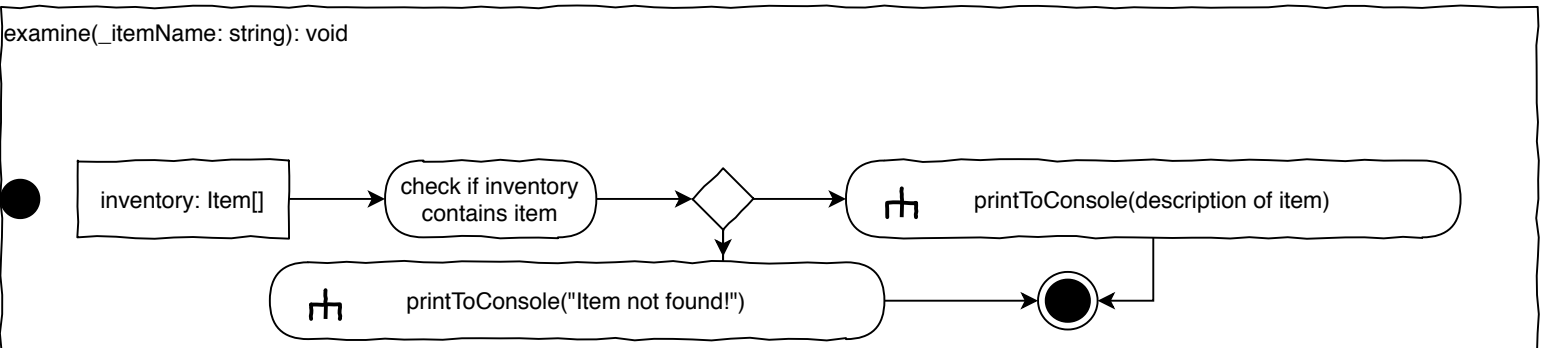

takeItem(itemName: string)
name: string
get the room in which the player is currently located
get the inventory array of the room
check if rooms inventory contains an item with this name
true
false
add the named item to the inventory array of the player
remove the named item from the rooms inventory
printToConsole("Item added to inventory")
printToConsole("Item not found")

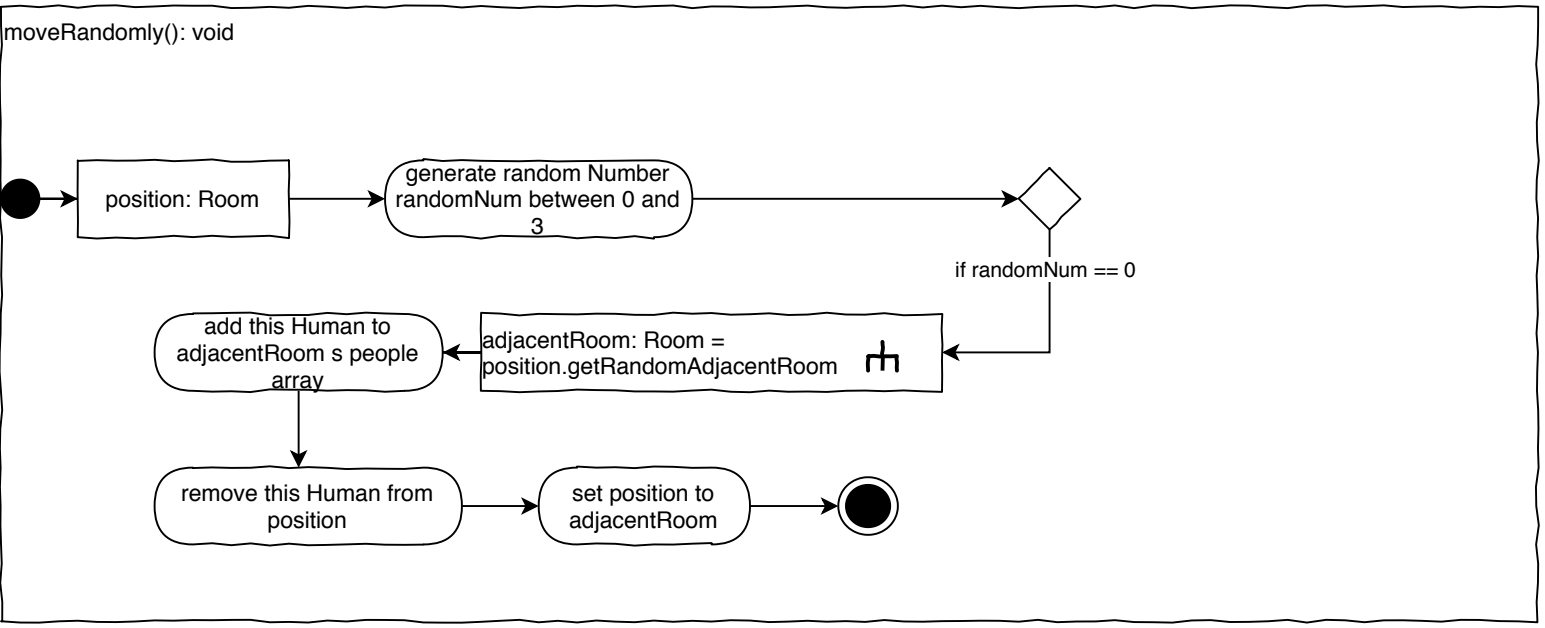

move(_direction: string)
_direction: string
check if _direction is valid enum
direction = <Direction> direction
get the room in which the player is currently located
Check whether another room is defined in the direction indicated
true
set player position to new room
printToConsole("You moved it into the" + newRoom.getName())
printToConsole("There is no room in this direction")
printToConsole(newRoom.getDescription())

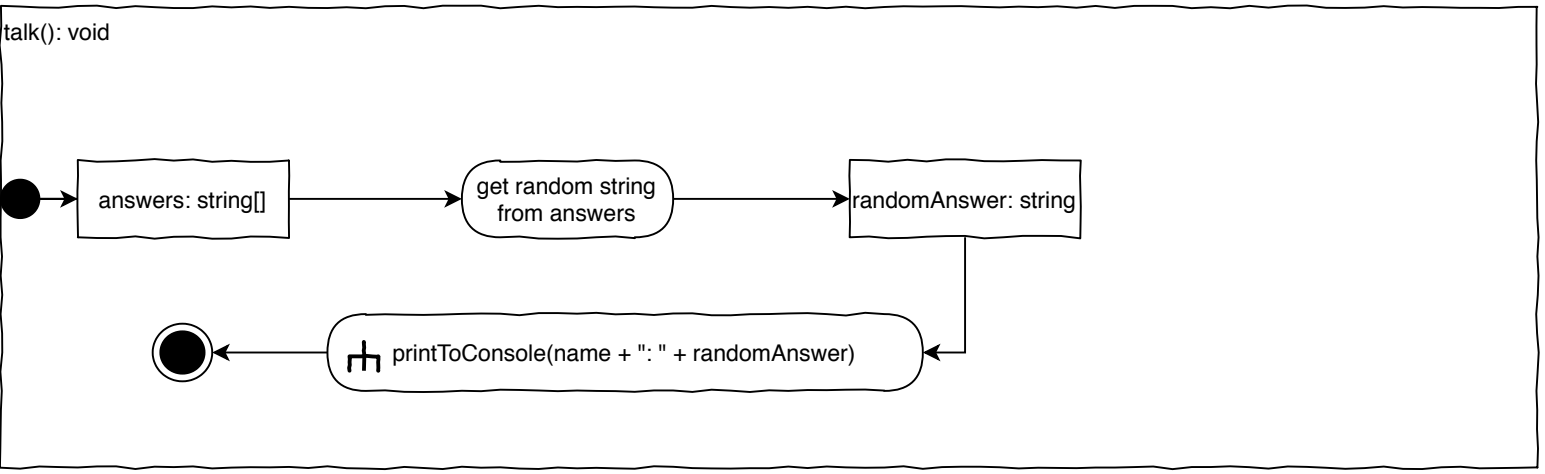

talkToNPC(nameOfNPC: string): void
nameOfNPC: string
get the room in which the player is currently located
check if people Array of room contains a human with this name
true
call talk() method of said human
printToConsole("Ther is no human with this name in this room")

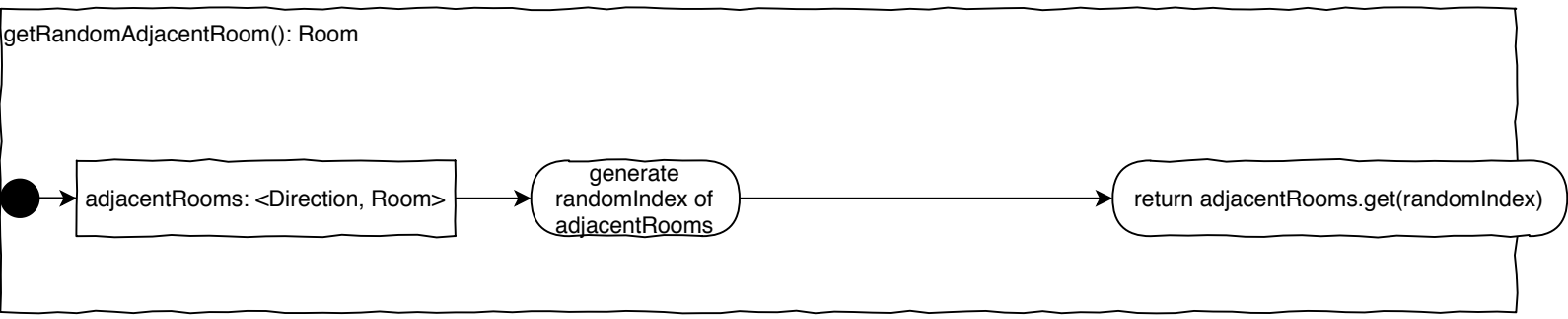

look(): void
get the room in which the player is currently located
printToConsole(room.getDescription())

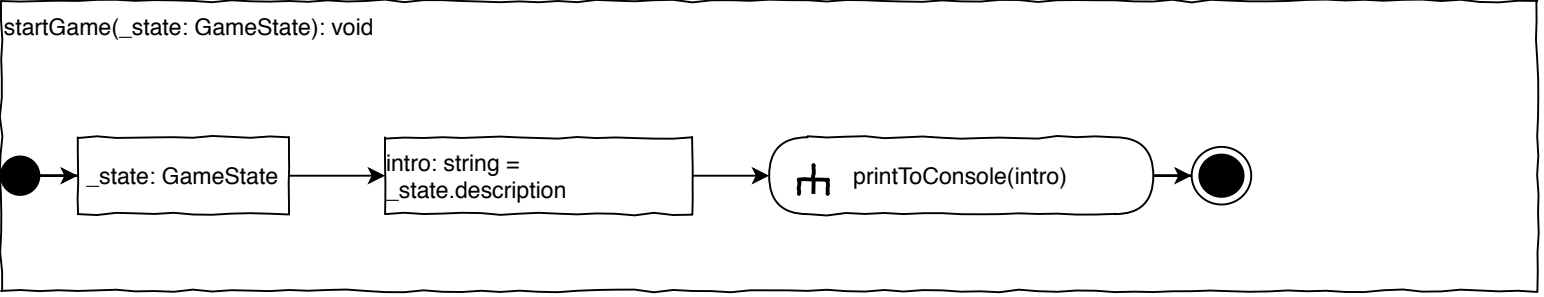

showInventory(): void
inventory: Item[]
i: number = 0
output: string = "Inventory: "
append inventory[0].getName() to output
if i < inventory.length()
increase i by one
printToConsole(output)

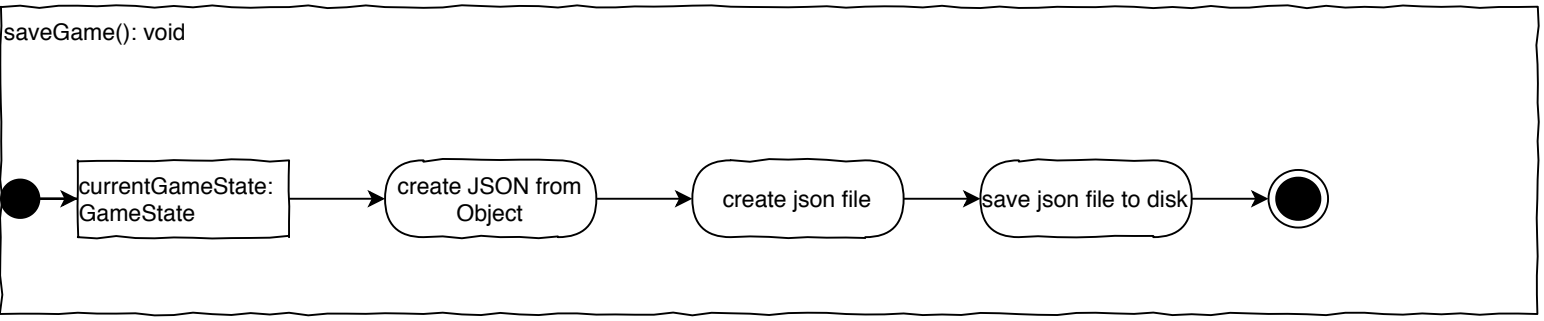

examine(_itemName: string): void
inventory: Item[]
check if inventory contains item
printToConsole(description of item)
printToConsole("Item not found!")

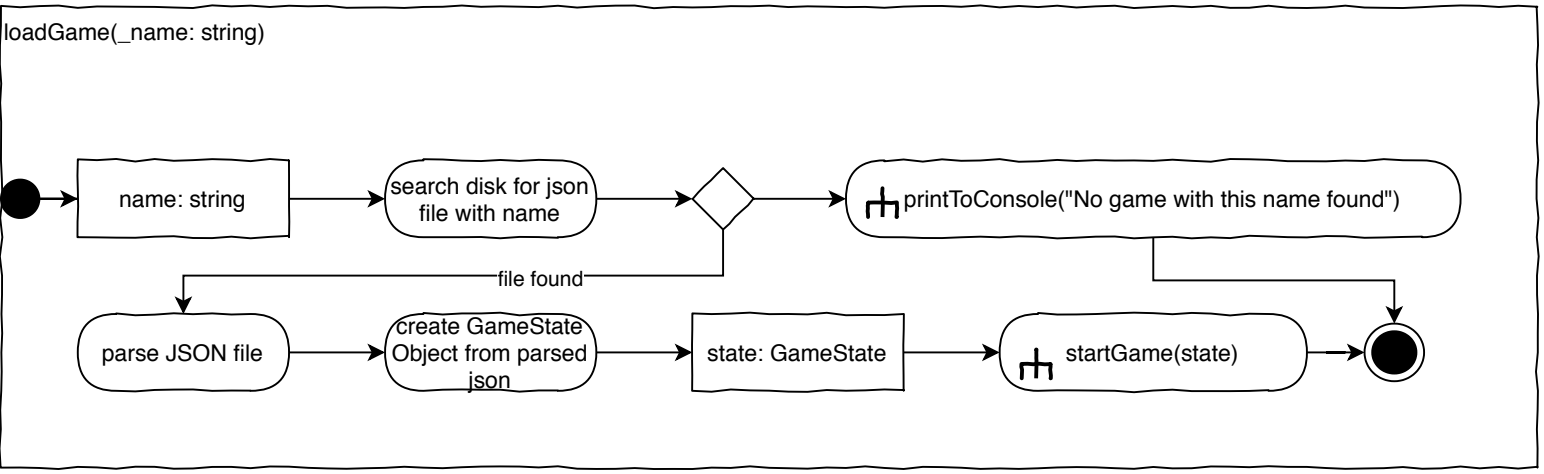

moveRandomly(): void
position: Room
generate random Number randomNum between 0 and 3
if randomNum == 0
adjacentRoom: Room = position.getRandomAdjacentRoom
add this Human to adjacentRoom s people array
remove this Human from position
set position to adjacentRoom

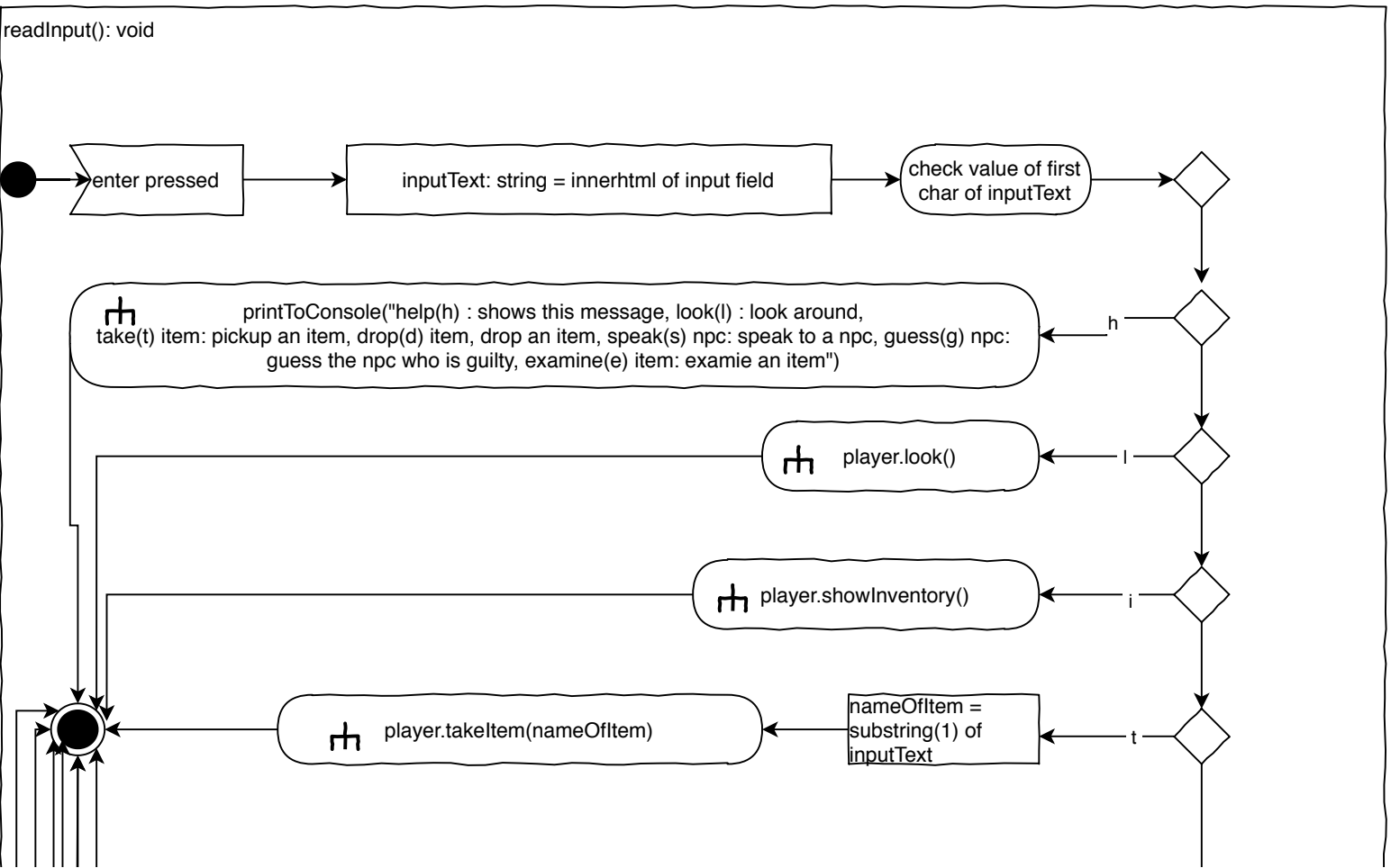

talk(): void
answers: string[]
get random string from answers
randomAnswer: string
printToConsole(name + ": " + randomAnswer)

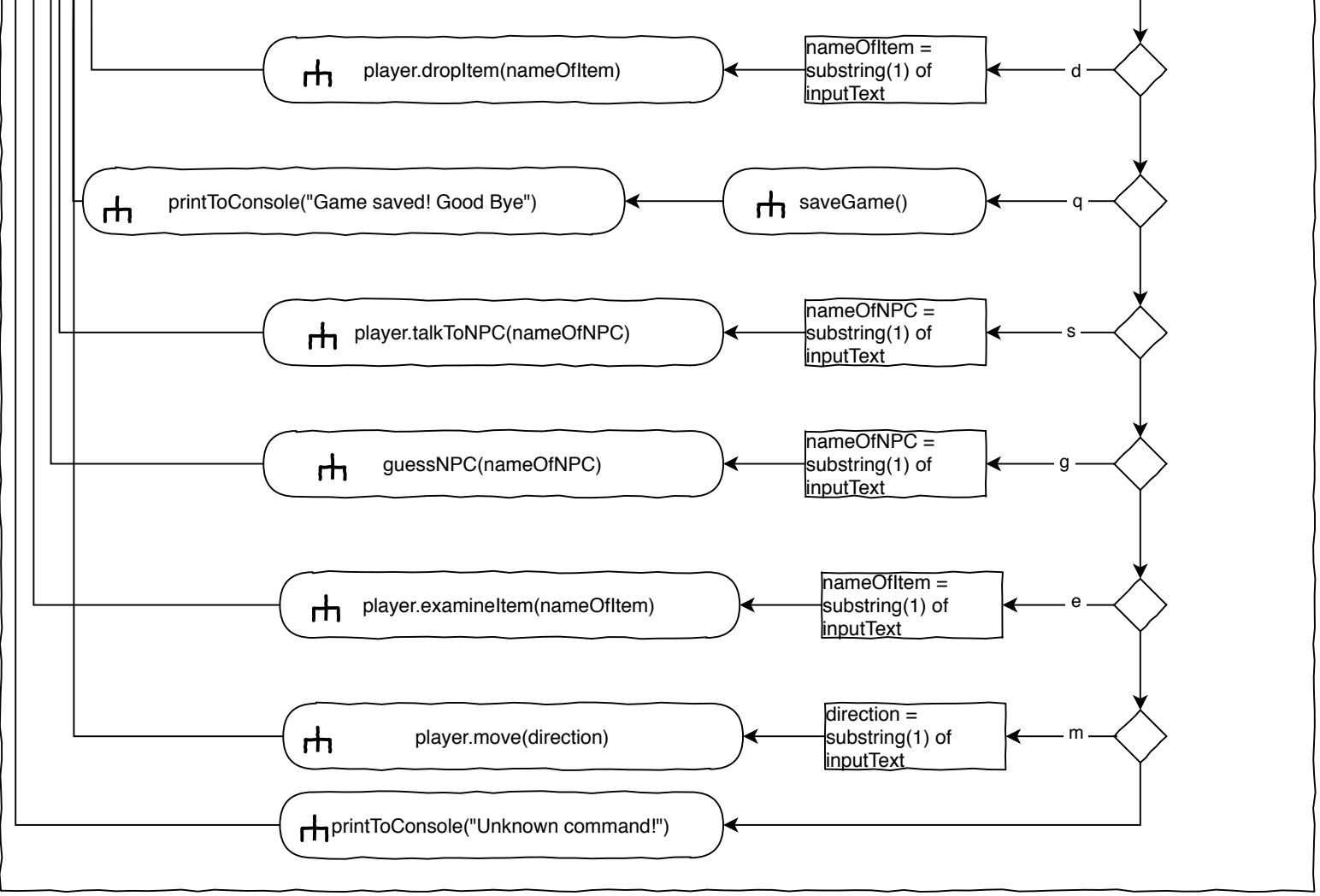

getRandomAdjacentRoom(): Room
adjacentRooms: <Direction, Room>
generate randomIndex of adjacentRooms
return adjacentRooms.get(randomIndex)

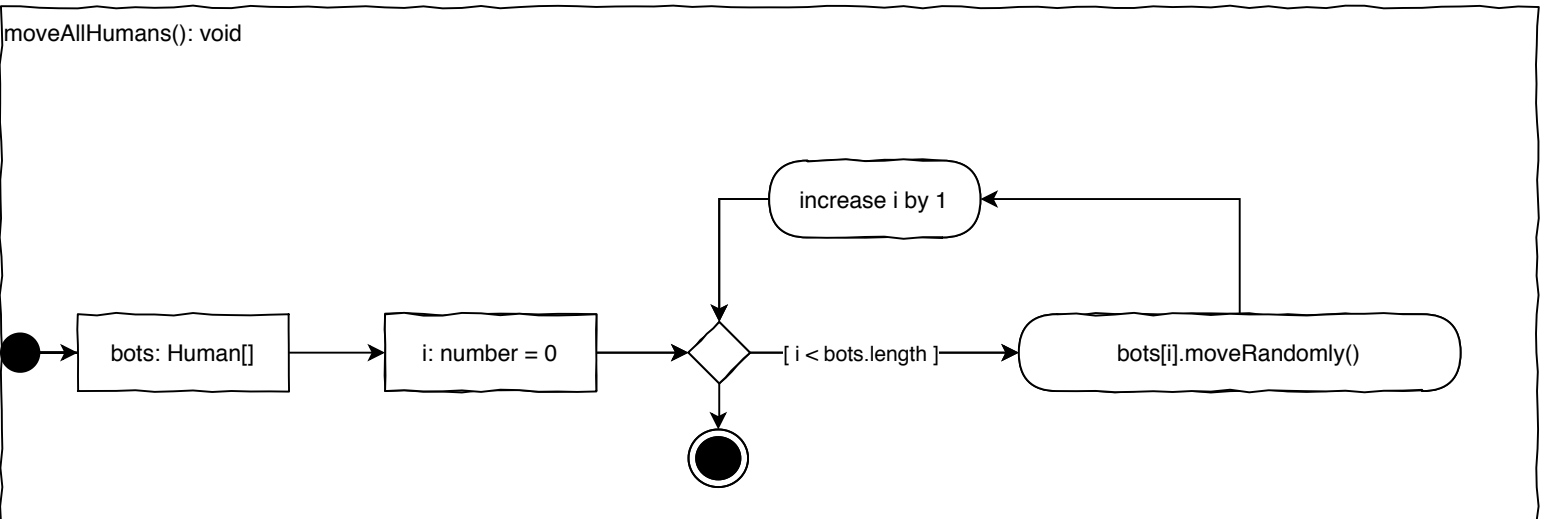

startGame(_state: GameState): void
_state: GameState
intro: string = _state.description
printToConsole(intro)


saveGame(): void
currentGameState: GameState
create JSON from Object
create json file
save json file to disk

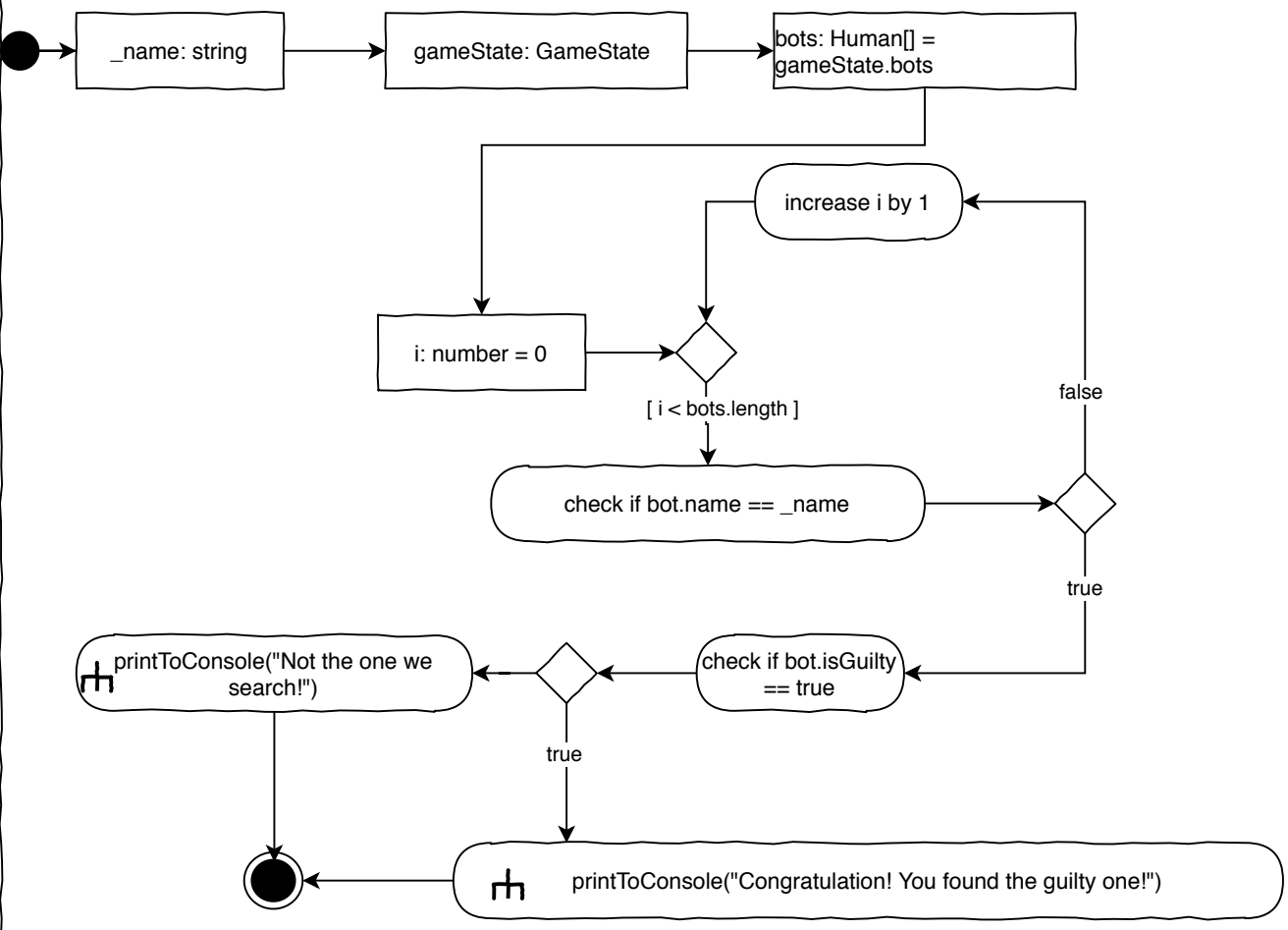

loadGame(_name: string)
name: string
search disk for json file with name
printToConsole("No game with this name found")
file found
parse JSON file
create GameState Object from parsed json
state: GameState
startGame(state)


readInput(): void
enter pressed
inputText: string = innerhtml of input field
check value of first char of inputText
h
printToConsole("help(h) : shows this message, look(l) : look around, take(t) item: pickup an item, drop(d) item, drop an item, speak(s) npc: speak to a npc, guess(g) npc: guess the npc who is guilty, examine(e) item: examie an item")
l
player.look()
i
player.showInventory()
t
nameOfItem = substring(1) of inputText
player.takeItem(nameOfItem)

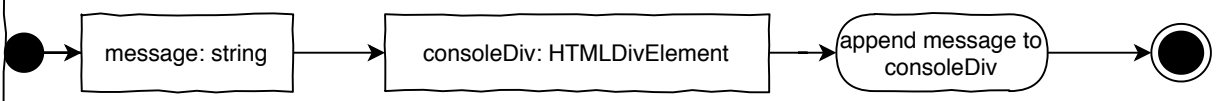

player.dropItem(nameOfItem)
nameOfItem = substring(1) of inputText
d
printToConsole("Game saved! Good Bye")
saveGame()
q
player.talkToNPC(nameOfNPC)
nameOfNPC = substring(1) of inputText
s
guessNPC(nameOfNPC)
nameOfNPC = substring(1) of inputText
g
player.examineItem(nameOfItem)
nameOfItem = substring(1) of inputText
e
player.move(direction)
direction = substring(1) of inputText
m
printToConsole("Unknown command!")

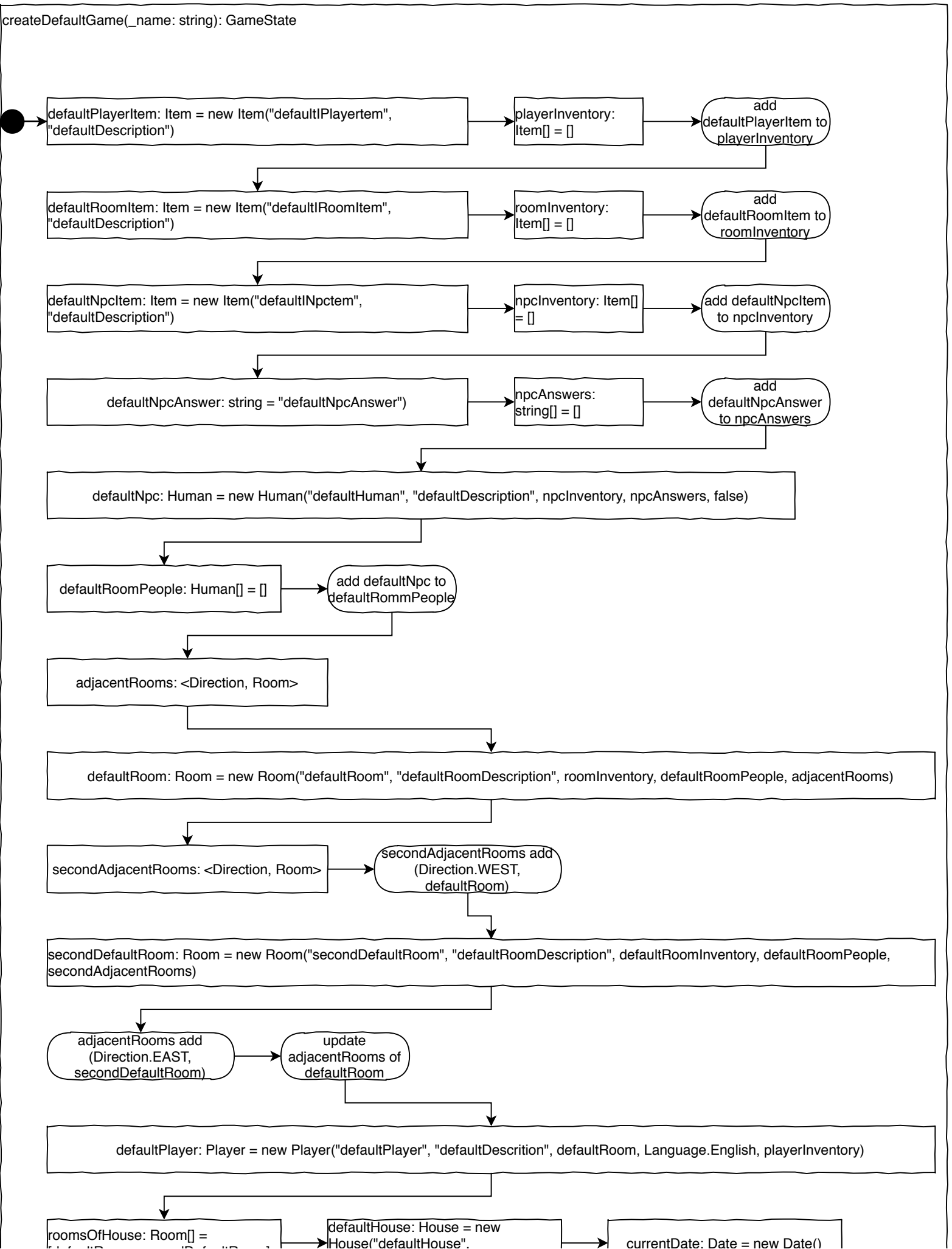

moveAllHumans(): void
bots: Human[]
i: number = 0
[i < bots.length]
bots[i].moveRandomly()
increase i by 1

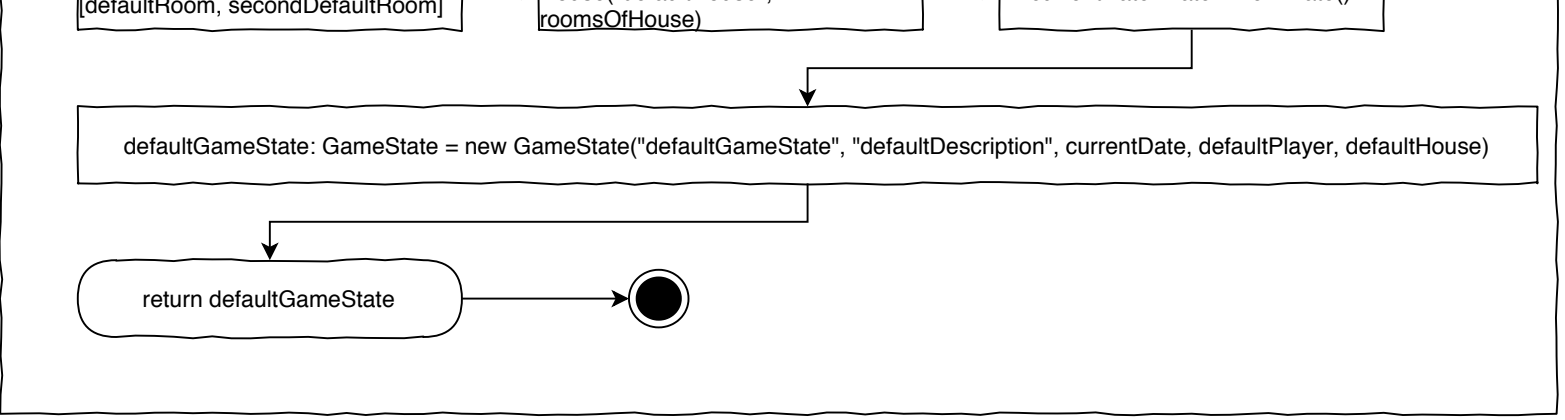
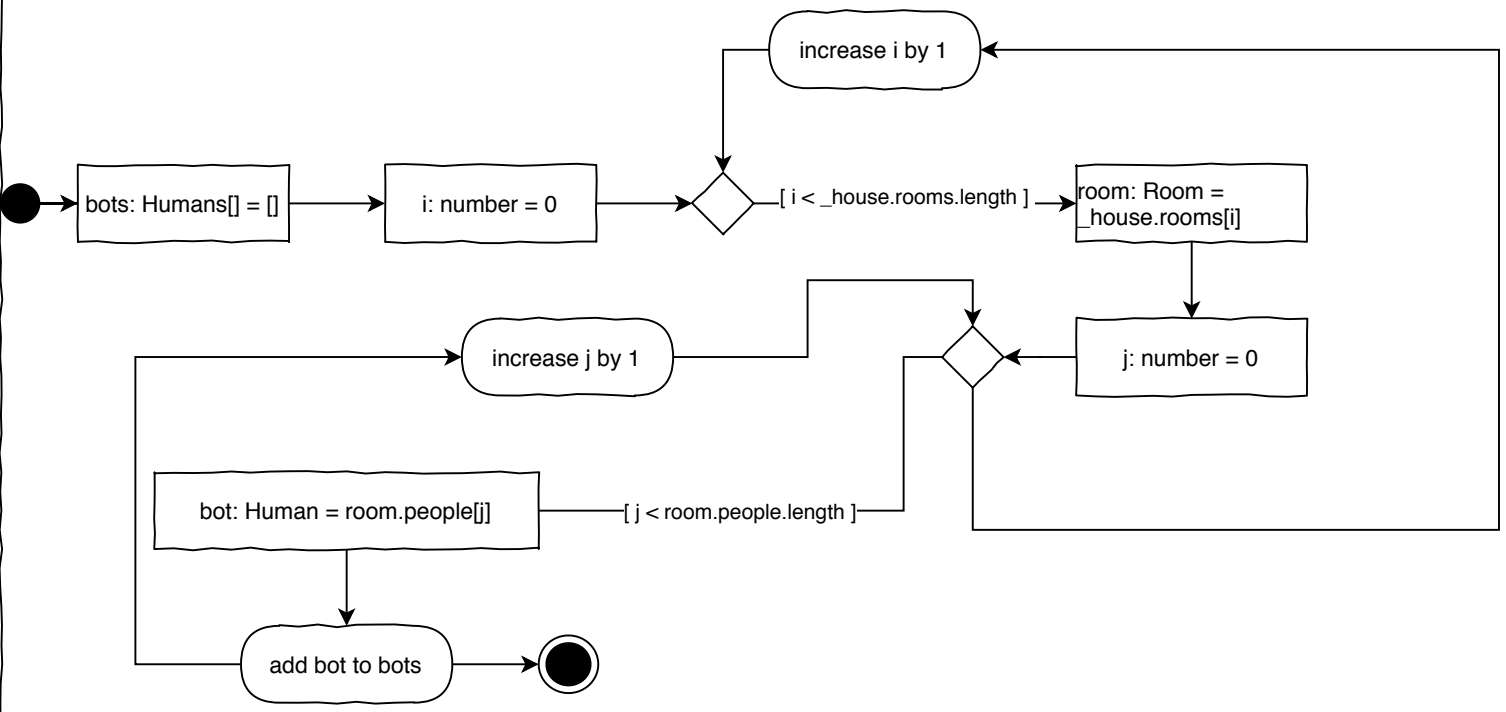
guessNPC(_name: string): void



printToConsole(message: string)




createDefaultGame(_name: string): GameState
defaultPlayerItem: Item = new Item("defaultIPlayertem", "defaultDescription")
playerInventory: Item[] = []
add defaultPlayerItem to playerInventory
defaultRoomItem: Item = new Item("defaultIRoomItem", "defaultDescription")
roomInventory: Item[] = []
add defaultRoomItem to roomInventory
defaultNpcItem: Item = new Item("defaultINpctem", "defaultDescription")
npcInventory: Item[] = []
add defaultNpcItem to npcInventory
defaultNpcAnswer: string = "defaultNpcAnswer")
npcAnswers: string[] = []
add defaultNpcAnswer to npcAnswers
defaultNpc: Human = new Human("defaultHuman", "defaultDescription", npcInventory, npcAnswers, false)
defaultRoomPeople: Human[] = []
add defaultNpc to defaultRommPeople
adjacentRooms: <Direction, Room>
defaultRoom: Room = new Room("defaultRoom", "defaultRoomDescription", roomInventory, defaultRoomPeople, adjacentRooms)
secondAdjacentRooms: <Direction, Room>
secondAdjacentRooms add (Direction.WEST, defaultRoom)
secondDefaultRoom: Room = new Room("secondDefaultRoom", "defaultRoomDescription", defaultRoomInventory, defaultRoomPeople, secondAdjacentRooms)
adjacentRooms add (Direction.EAST, secondDefaultRoom)
update adjacentRooms of defaultRoom
defaultPlayer: Player = new Player("defaultPlayer", "defaultDescrition", defaultRoom, Language.English, playerInventory)
roomsOfHouse: Room[] =
defaultHouse: House = new House("defaultHouse",
currentDate: Date = new Date()


[defaultRoom, secondDefaultRoom]
roomsOfHouse)
defaultGameState: GameState = new GameState("defaultGameState", "defaultDescription", currentDate, defaultPlayer, defaultHouse)
return defaultGameState

constructor(_name: string, _description: string, _saveDate: Date, _player: Player, _house: House): GameState