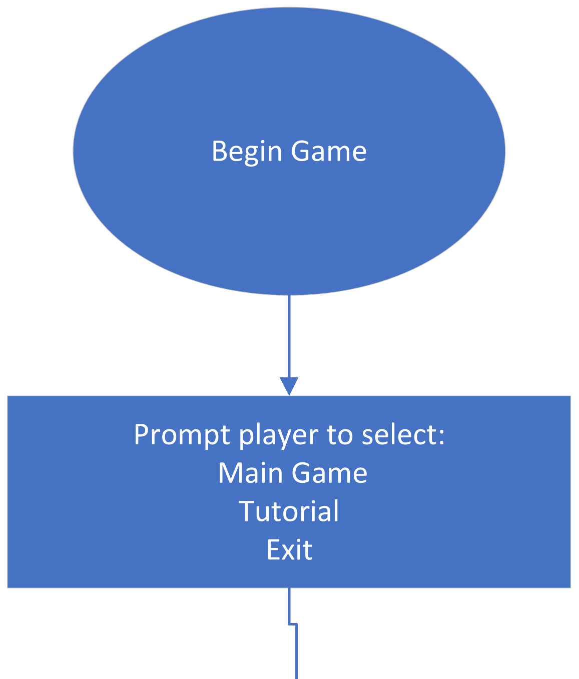
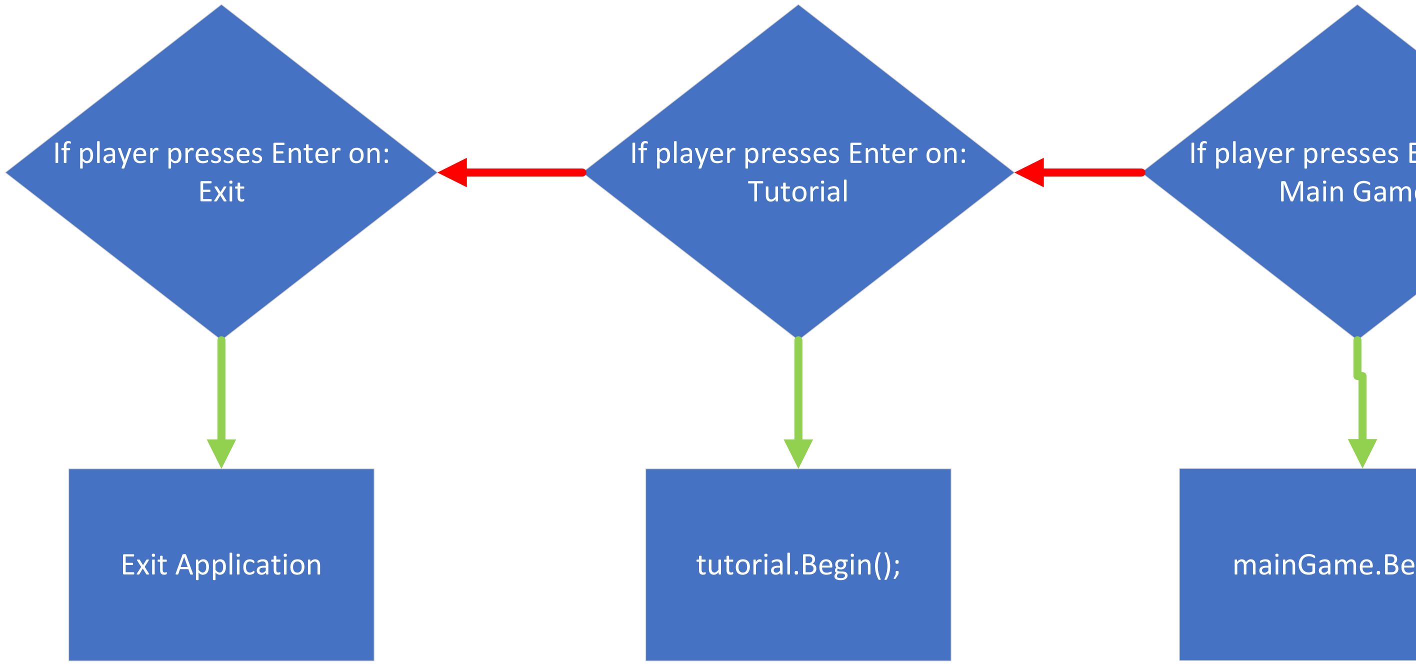


When the application opens, the player will be a given
along with **the title screen**. via **printing options.**
- Choice 1: **Begin** the MainGameNarrative.
- Choice 2: **Begin** the TutorialGameNarrative.

a given **2 choices**,
s.

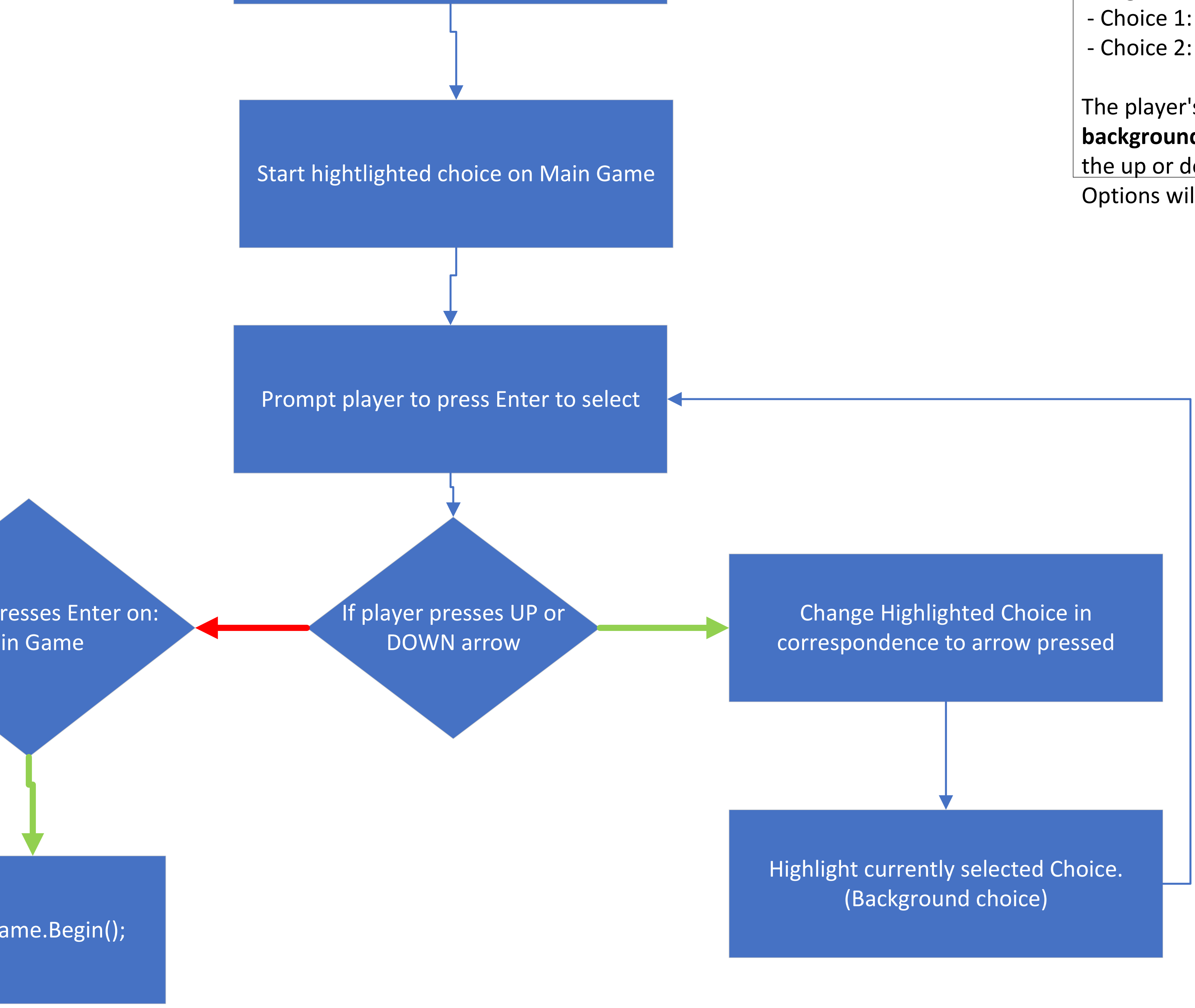
MainMenu

mainGame : MainGameNarrative
tutorial : TutorialGameNarrative
menuPrompts : string[];
title : string;


If player presses Enter on: Exit
If player presses Enter on: Tutorial
If player presses E
Main Gam
Exit Application
tutorial.Begin();
mainGame.Be

- Choice 1: **Begin** the MainGameNarrative.
- Choice 2: **Begin** the TutorialGameNarrative.

The player's **cursor** will be invisible and will be utilizing
background select option. The player can change choi
the up or down arrow key.
Options will be centered on the string.



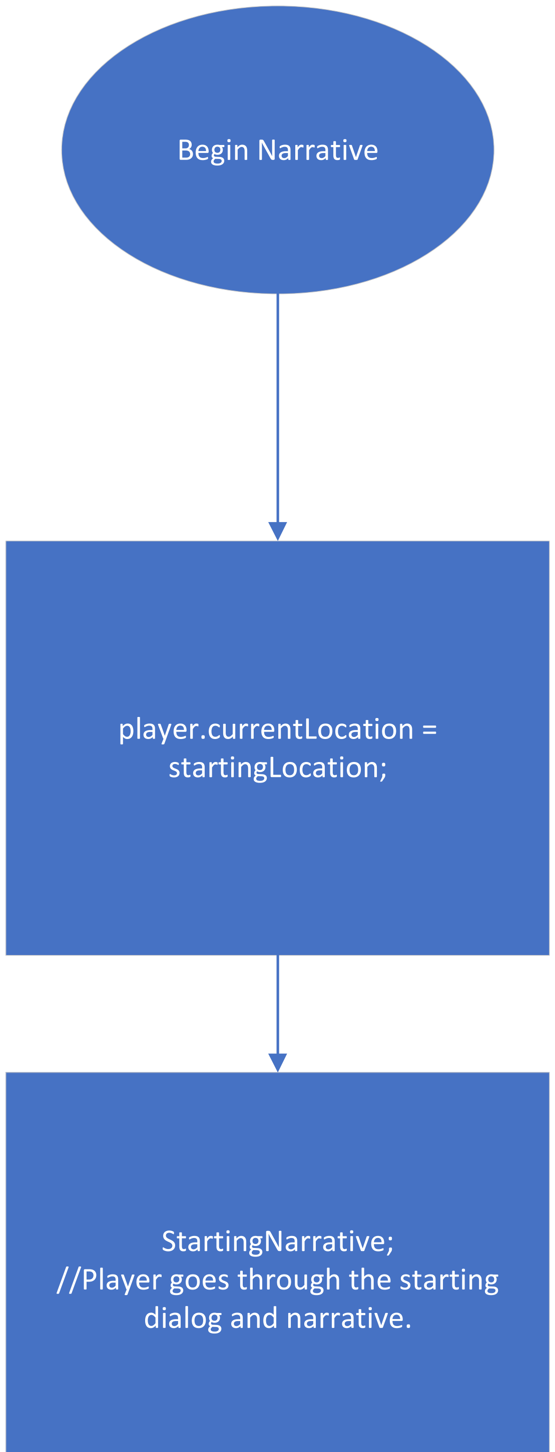
Start hightlighted choice on Main Game
Prompt player to press Enter to select
resses Enter on:
in Game
If player presses UP or DOWN arrow
Change Highlighted Choice in correspondence to arrow pressed
Highlight currently selected Choice. (Background choice)
ame.Begin();

utilizing a highlight /
ge choices by using

title : string;

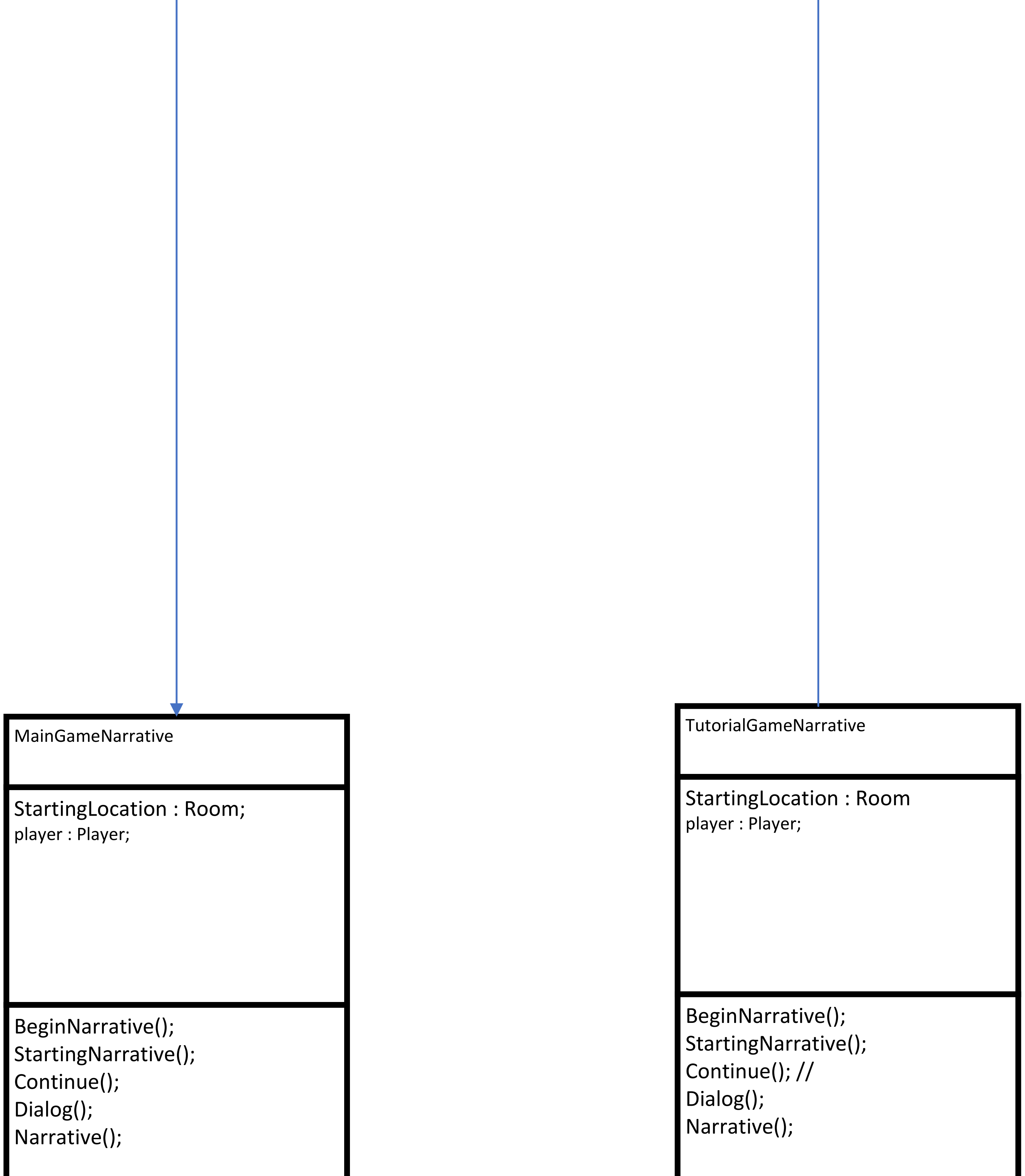
MenuStartup();

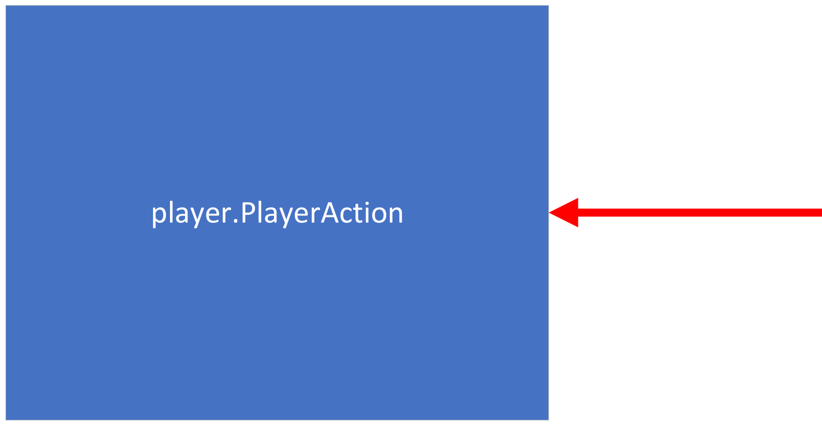
Menu();
GetSelectedMenuChoice();
CenterString(); //Center to screen

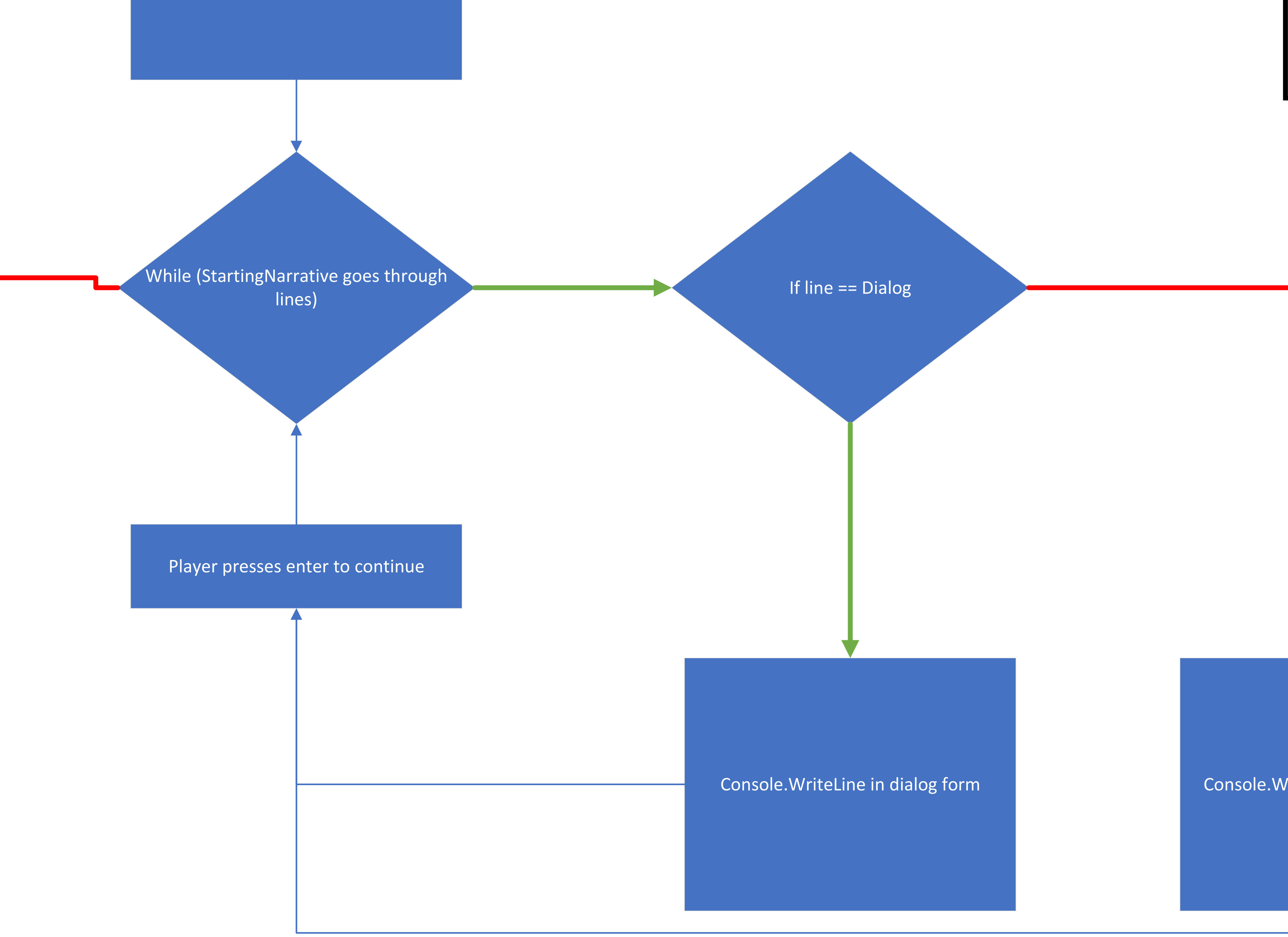

Begin Narrative
player.currentLocation = startingLocation;
StartingNarrative; //Player goes through the starting dialog and narrative.

- The game **begins**.
- The game places the player in a **starting location.**
- The **starting narrative** pops up on the screen, beginning a short narrative for the player to read as they begin the game.

    -This can be in the form of **dialog** or **narrative**.
- As the game plays, there will be several different **checkers** in the class, checking and alternating the narrative as the player plays.


MainGameNarrative
StartingLocation : Room;
player : Player;
BeginNarrative();
StartingNarrative();
Continue();
Dialog();
Narrative();
TutorialGameNarrative
StartingLocation : Room
player : Player;
BeginNarrative();
StartingNarrative();
Continue(); //
Dialog();
Narrative();


player.PlayerAction


While (StartingNarrative goes through lines)
If line == Dialog
Player presses enter to continue
Console.WriteLine in dialog form
Console.W

Dialog();
Narrative();

Narrative();

Suspect:
- The suspect is created via the policeVan.

Sus

nan

evic

nsole.WriteLine in narrative form

Character

name : string;

currentState : string;

AllCha

target
directo
InfoFil

Suspect

name : string;

evidence : List<Clue>;

PoliceVan

name : string;

currentState : string;
description : string;
location : Room;

suspects : List<Suspect>
containedItems : List<Item>

AddSuspect();
RemoveSuspect();

AddEvidence();
RemoveEvidence();

AddItem();
RemoveItem();

The policeVan:
The police van will act as an interface for the player to choose who to accuse and to associate clues or items to the accused.
- Player uses the van to accuse suspects (Add a character to the suspect list).
- The van holds a list of all the characters the player has accused.
- The player uses the van to un-accuse accused suspects.

- The player uses the van to associate clues to the suspects
- The player uses the van to disassociate clues from the suspects.

- The player can add or remove items from the van, like a container.

AllCharacterObjects

targetFolder : string;
directoryPath : string;
InfoFiles : string[];

- Suspects will have a name and a general description.

- Suspects will have a current state, meaning what they are doing at the moment.

to

pect

- Questions will have a question prompt.
- Questions will have an answer.
- Some answers will require an item. If the item is not present in the player’s inventory, the question will be Unavailable.
- If the original question requires another separate question to be answered, the

name : string;

currentState : string;
description : string;
location : Room;

questionList : List<Question>;
availableQuestions : List<Question>;

CreateListOfQuestions();
RevealListOfQuestions();
GiveDescription();

targetI
directo
InfoFil
allChar

SetDir
Create
Create

Game

jazzNoir : SoundPlayer;
player : Player
allRoomObjects : AllRoomObjects;
allCharacterObjects : AllCharacterObjects;
allItemObjects : AllItemObjects;
allDoorObjects : AllDoorObjects;

PlayMusic();
GameStartUp();

```
targetFolder : string;
directoryPath : string;
InfoFiles : string[];
allCharacters : static List<Character>
-----------------------------------
SetDirectory();
CreateCharacters();
CreateQuestions();
```

- Suspects will have a current state, meaning what they are doing at the moment.

- Suspects will have their own list of questions that the player can ask.

- Suspects will be in a location.

- When the player talks to the suspect, the suspect reveals their list of questions

 - When the game starts, suspects create their questions

```
Questio
-----------------------------------
questior
questior
answer

requirec
requirec
requirec
requirec

clue : Cl
clueSum
-----------------------------------
IsQuesti
Display(
GiveAns
IsRequir
IsRequir
```

```
ObjectColors
-----------------------------------

-----------------------------------
ItemColor();
RoomColor();
PersonColor();
DoorColor();
```

```
Clue
-----------------------------------
clueSummary : string;
clueName : string;
locationFound : string;
personStatementBy : string;

clueType : string;
-----------------------------------
RevealClueInfo_Item();
RevealClueInfo_Character();
```

the question will be Unavailable.

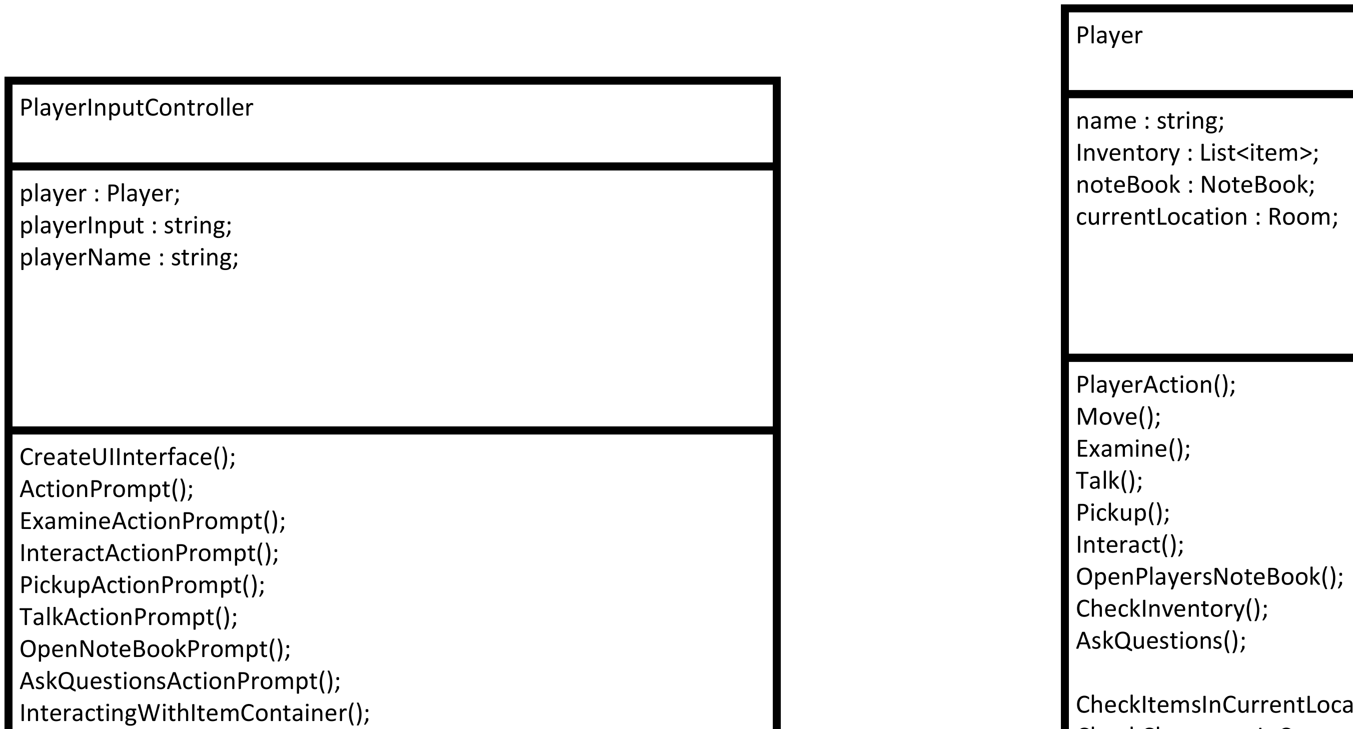
- If the original question requires another separate question to be answered, the original question will be Unavailable.
- Include the separate question object if the original question requires it’s answer.
- If the original question is a required question for a follow-up question, the original question will change the follow-up question’s unavailable to available. The separate.
- If a question was already answered, highlight the questionPrompt in a separate color.

Question

questionID : string
questionPrompt : string;
answer : string;
 	
requiredQuestionID : string;
requiredClueID : string;
requiredQuestion : Question;
requiredClue : Clue;
 	
clue : Clue;
clueSummary : string;

IsQuestionAvailable();
DisplayQuestionPrompt();
GiveAnswer();
IsRequiredClueLogged();
IsRequiredQuestionAnswered();

```
- Suspect reveals questions

Check to see if question requires an item
{
            - If item is logged in notebook, playerHasClue = true;
           - else if item is not logged in notebook, playerHasClue = false;
}

isQuestionAvailable()
{
If playerHasClue = true and isRequiredQuestionAnswered = true;
Then isAvailable = true;
}

DisplayQuestionPrompt()
{
             If isAvailable = true;
            - Display questionPrompt.
}



- Player selects an available question

DisplayAnswer
{
Write answer;
isAnswered = true;
If question has a followUpQuestion
            followUpQuestion.isAvailable = true;
}
AnsweredColor()
{
```


PlayerInputController
player : Player;
playerInput : string;
playerName : string;
CreateUIInterface();
ActionPrompt();
ExamineActionPrompt();
InteractActionPrompt();
PickupActionPrompt();
TalkActionPrompt();
OpenNoteBookPrompt();
AskQuestionsActionPrompt();
InteractingWithItemContainer();
Player
name : string;
Inventory : List<item>;
noteBook : NoteBook;
currentLocation : Room;
PlayerAction();
Move();
Examine();
Talk();
Pickup();
Interact();
OpenPlayersNoteBook();
CheckInventory();
AskQuestions();
CheckItemsInCurrentLoca

tem>;
Book;
: Room;

eBook();
);

rentLocation();

- Items that have been marke
notebook will appear once t
opened.
- (1. Clue1)
- (2. Clue2)
-
- The player's inventory will a
book.
-
- Note can open to reveal opt

n marked in the
r once the notebook is

)
)

ry will appear in the note

veal options.

NoteBook
clues : List<Clue>; statementClues : List<Clue>; observationClues : List<Clue>; playersInventory : List<item>

AddClue();
OrganizeStatements();
OrganizeOberservationClues();

- The player can examine each room for items.

```
}
AnsweredColor()
{
If isAnswered = true, change color.
}
```

AllRoomObjects

```
targetFolder : string;
directoryPath : string;
InfoFiles : string[];

rooms : List<Room>;
```

```
SetDirectory();
CreateRooms();
GiveRoom();
```

Room

```
roomID : string;
name : string;
description : string;

northRoomID : string;
southRoomID : string;
eastRoomID : string;
westRoomID : string;

northDoorID : string;
southDoorID : string;
eastDoorID : string;
westDoorID : string;

northRoom : Room;
southRoom : Room;
eastRoom : Room;
westRoom : Room;
northDoor : Door;
southDoor : Door;
eastDoor : Door;
westDoor : Door;
```

- Rooms will have a north, south, east, west side of it.
- Each room with have a unique description to each one.
-
- Rooms will have a list of items and characters in it.
-
- If there is a room in any direction, the player can go to that room, UNLESS there is a door object that is locked.

```
OpenNoteBookPrompt();
AskQuestionsActionPrompt();
InteractingWithItemContainer();
MarkAsClue();
InteractingWithDoor();

HighlightSelectedChoice();
ColorRequiredInputPrompt();
ColorCharacterName();
CheckForCommandShortcut();
AskForInput();
GetInput();
Dialog();
ValidatePlayerInput();
CreateInterfaceBorder();
InvalidInputWarning();

InterfaceCompass();
CompassRoomColor();
CompassDrawDoorWithColor();

CenterString();
```

```
AskQuestions();

CheckItemsInCurrentLoca
CheckCharactersInCurren
CheckDoorsInCurrentLoca
```

```
rrentLocation();
InCurrentLocation();
rrentLocation();
```

Identifier

Attributes

Methods

- The player can examine each room for items.
- The player can pickup items as long as the item is not too heavy.
- The player can talk and ask questions to each suspect.
-
- The player can interact with items, and can also inspect an item for details.
- - The item will give a specific string when it's interacted with.
-
-
- Interacted items or picked up items that are clues will be marked in the notebook.
- If the player feels like they have a suspect, they can accuse a suspect.
- The player can review items they have come across in their notebook
-
- Picked up items will be stored in the Player’s inventory

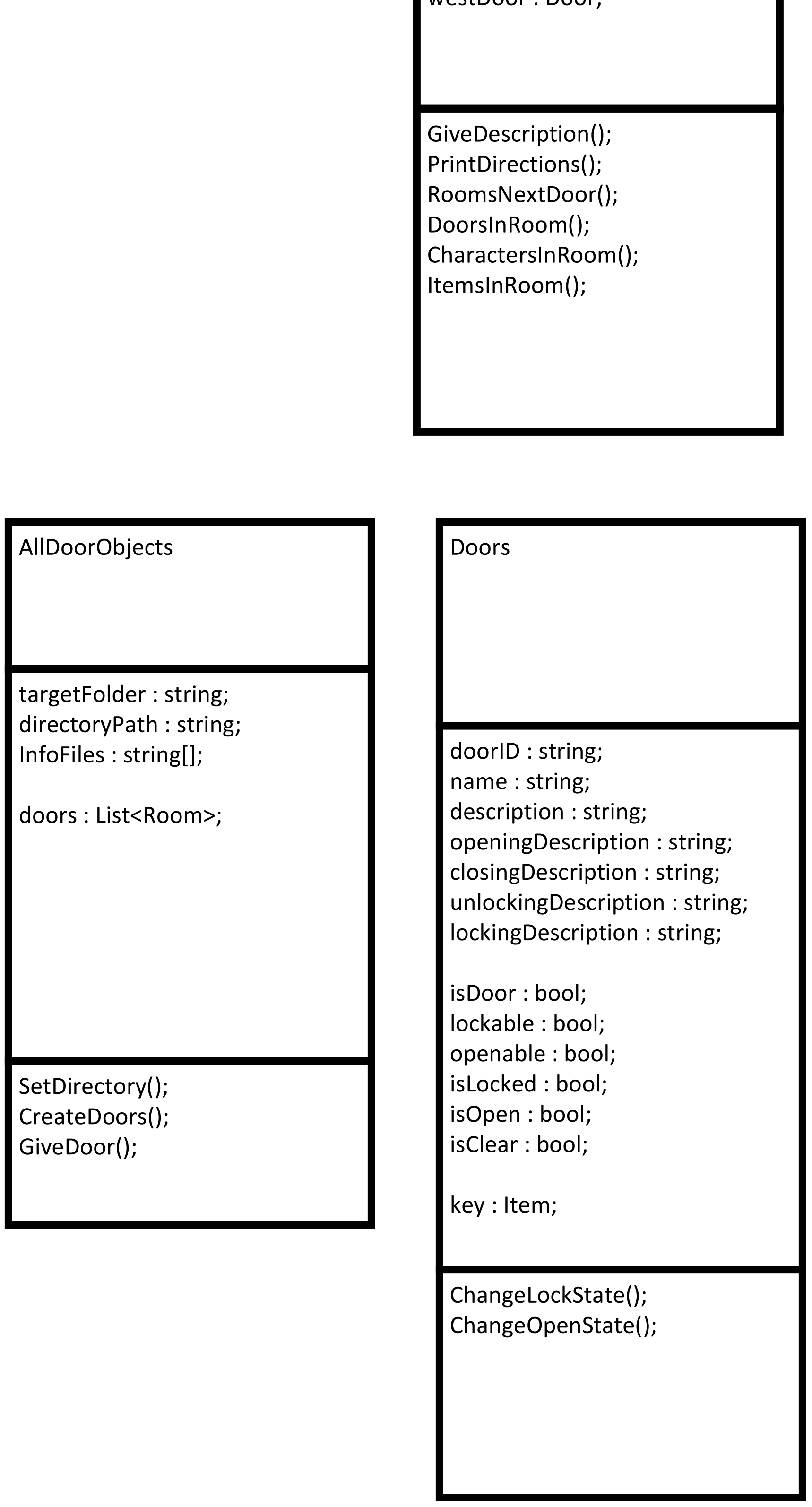
AllItemObjects

targetFolder : string;
directoryPath : string;
InfoFiles : string[];
 	
items : List<Item>;

SetDirectory();
CreateItems();
SetContainedItemsForContainerItems();
SetKeyItemsForContainerItems();
GiveItem();

Item

location : Room;
clue : Clue;
 	
itemID : string;
name : string;
currentState : string;
description : string;
interactDescription : string;
 	
size : int;
canPickUp : bool;
 	
isContainer : bool;



westDoor : Door;

GiveDescription();
PrintDirections();
RoomsNextDoor();
DoorsInRoom();
CharactersInRoom();
ItemsInRoom();

AllDoorObjects

targetFolder : string;
directoryPath : string;
InfoFiles : string[];

doors : List<Room>;

SetDirectory();
CreateDoors();
GiveDoor();

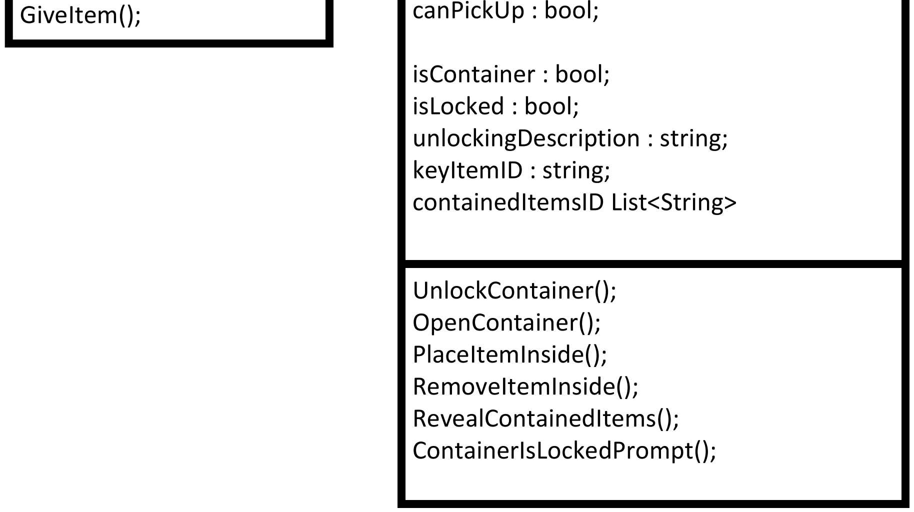
Doors

doorID : string;
name : string;
description : string;
openingDescription : string;
closingDescription : string;
unlockingDescription : string;
lockingDescription : string;

isDoor : bool;
lockable : bool;
openable : bool;
isLocked : bool;
isOpen : bool;
isClear : bool;

key : Item;

ChangeLockState();
ChangeOpenState();


GiveItem();
canPickUp : bool;
isContainer : bool;
isLocked : bool;
unlockingDescription : string;
keyItemID : string;
containedItemsID List<String>
UnlockContainer();
OpenContainer();
PlaceItemInside();
RemoveItemInside();
RevealContainedItems();
ContainerIsLockedPrompt();