Bot Logic

If the bot does not know the location of the murder{

- Move to the nearest room which we have no info on(i.e whether it was the murder room or not).
- Ask a question.
- If we do not know the weapon and/or suspect ask with a random weapon and/or suspect we have no info on(i.e whether it was the murder weapon or not/whether they were the murderer or not).
- If we do know the weapon and/or suspect ask with a weapon and/or suspect we were dealt at the start or if no weapon and/or suspect was dealt at the start as with the murder room.
  (The 2 steps above avoid retrieving information from other players that we already know.)

else we know the murder room{

- Go to the murder room or if we were just in the murder room do to the room closest to the murder room.
- Ask a question
- If weapon has been found and suspect not, ask with a weapon we were dealt at the start or the murder weapon.
- If suspect has been found and weapon not, ask with a suspect we were dealt at the start or the murderer.
- If we do not know the weapon and suspect ask with a random weapon and suspect we have no info on(i.e whether it was the murder weapon or not/whether they were the murderer or not).

else we know the contents of the envelope{

- Move to cellar
- Make accusation

Methods:

void pathFInd(Coordinates start, Coordinates end)
This is an A* algorithm that finds the shortest path between the start and the end coordinates through the corridors.

String[] getDirections(Coordinates curr, Coordinates end)
This method returns a string array with a specific ordering of {“l”, “u”, “r”, “d”} where the first two strings are the horizontal and vertical position of the end coordinate relative to the current coordinate. This was to improve pathFind method.

String getDirection(Coordinates prev, Coordinates next)
This method finds the direction between to coordinates (next to each other). Returns a string from the list {“l”, “r”, “u”, “d”}.

boolean containsCoord(ArrayList<Coordinates> list, Coordinates c)
this method checks whether a coordinate is in a list of coordinates, if it is in the list true is returned otherwise it is false. This was due to an issue with the ArrayList contains method that didn’t do exactly what we wanted it to do so we made our own.

int heuristic(Coordinates start, Coordinates end)
returns the Manhattan distance between the start and end coordinates

String getMove()
Returns a string from the list {“l”, “r”, “u”, “d”}. indicating the direction to move the bot. if the path is empty (in case something goes wrong), check to see if we don’t know the answer and don’t want to make an accusation or if we do. Remove the first character from the path string, make it a string and return that string.

int getClosestDoor(Coordinates start, Room endRoom)
This finds the closest door of a room relative to a coordinate. It returns the index of the closest door in the array.

String getDoor()
Returns a string representation of an integer which represents the door of the room that the bot will exit the room by. It finds the nearest room out of all of the rooms that have been unseen. Loop through all of the doors in the room that the bot is currently in finding the door with the shortest distance heuristically between then and getting the shortest out of all of them. Find the path that the bot needs to follow to get to the desired room. Set the pathEndRoom (the nearest room from the list of unseen rooms) to null and return the door number.

String getCommand()
Overall general control of bot actions. If the player can roll the dice will be rolled, if they enter a room they will ask a question, or if they enter the cellar they make an accusation, if they decide they want to take a passage and the room has a passage they will take the passage.

String getSuspect()
Returns the name of the character we want to question/ believe to be the murderer. If we know the murder question a character who we were dealt at the start, therefor no useless information will be told to us about suspects, if we don't have any suspects dealt to us at the start ask with the murderer. Else if the don't know who the murderer is, ask with a random character to find out if they are the murderer.

String getWeapon()
Returns the name of the weapon we want to question with/ believe to be the murder weapon. If we know the weapon question with a weapon which we were dealt at the start, therefor no useless information will be told to us about weapons, if we don't have any weapons dealt to us at the start ask with the murder weapon. Else if the don't know what the murder weapon is, ask with a random weapon to find out if it was the murder weapon.

String getRoom()
Returns the murder room

String getCard(Cards matchingCards)
This is a method that returns a card in the matching card. It has been implemented that if a card in matchingCards has already been shown to a player, it returns that one. This is to show the same card to other players so as to make it harder for them to win the game.

Boolean isRoomKnown()
Returns true if we know the room. To see if we know the room we check if there is only 1 card that is unseen and that must be the murder room. Returns false otherwise.

Boolean isWeaponKnown()
Returns true if we know the weapon. To see if we know the weapon we check if there is only 1 card that is unseen and that must be the murder weapon. Returns false otherwise.

Boolean isSuspectKnown()
Returns true if we know the murderer. To see if we know the murderer we check if there is only 1 card that is unseen and that must be the murderer. Returns false otherwise.

Void updateUnseenWeapons()
Makes sure the weapon cards that have not been seen are up to date

Void updateUnseenSuspects()
Makes sure the suspect cards that have not been seen are up to date

Void updateUnseenRooms()
Makes sure the room cards that have not been seen are up to date

Void initialiseSuspectsDealt()
Initialises an array list of all the suspects we were dealt

Void initialiseRoomsDealt()
Initialises an array list of all the rooms we were dealt

Void initialiseWeaponsDealt()
Initialises an array list of all the weapons we were dealt

Boolean decideToAsk()
If the room we are now in is not the same as the last room we were in then ask a question by returning true. Else return false.

Boolean decideToTakePassage()
Decide if it is worth it for the bot to take the secret passage

Room nearestRoom(Arraylist<Strings> rooms)
Finds the nearest room to the players current location from the passed in list of rooms. It does this using the manhattan distance between the 2 points.