Game	
• List of deck as a field • List of solution a field • Map of Players a field • Starts: asks how many players wish to participate. • Initialises game and sets up the game to be played. • The solution of the game is chosen and put in the envelope. • Remaining cards are shuffled and distributed to the players. • Adds cards to each player's hand. • Rolls two dices for each player's turn. • Allows players to make Suggestion or Accusation • If player chooses Suggestion, allow other players to refute • If player chooses Accusation, check if game is won or not	• Board • Player • Position • Card • Hand • Square • RoomSquare • StartingSquare • PlainSquare • Move

Board	
• Constructor: Create the game board • Returns the type of Square using getSquare(String name) method • Returns list of starting squares in order of play • toString() method for string representation of the board • Update the board after each move	• Square • PlainSquare • Position • RoomSquare • StartingSquare

Abstract Square PlainSquare, RoomSquare, StartingSquare	
• Stores name of Square as a String field • Stores name of Player as a field • Has isOccupied as a boolean field to check if room is occupied • Has isAccessible as a boolean field to check if room is accessible • toString() method • getPlayer() method that return the current player occupying the current square • Setter method: setOccupied(boolean flag, Player p) indicates whether the current square is already occupied by another player and isAccesscible or not	• Position

PlainSquare Square	
• Extends from Square class • Represents the squares that are inaccessible (door and X squares) • Has boolean method isAccessible() that return is Square is accessible or not	

RoomSquare	Square
• Has enum Room with all room types(Kitchen, Ballroom, Conservatory, Dining room, Billiard room, Library, Lounge, Hall, Study) • Boolean field hasPlayer • Static field Map<String, Room> rooms • Private Room field name room • Constructor for the Room Square • A private method fillMap() that links the RoomSquares to the corresponding Rooms • A getRoom() method that should return the Room • An isAccessible() method that return true or false depending on if the player is on the square • A setOccupied() method that takes in two arguments and stores the information if a player is occupying a square • A setWeapon() method that sets a square that will have the weapon	

StartingSquare	Square
• Has a private field String playerLabel • A constructor called StartingSquare that takes in two parameters String name and String playerLabel and calls the super constructor • Has boolean method isAccessible() that return true or false depending upon if the name of the player does not match the name stored by this square then it isn't accessible • Another method getPlayerLabel() that return the player label •	

Player	
• Has enum PlayerToken that stores the player names (MISS_SCARLETT, COLONEL_MUSTARD, MRS_WHITE, MR_GREEN, MRS_PEACOCK, PROFESSOR_PLUM) • Has a static method getNext that takes in PlayerToken as a parameter and returns the next token clockwise from this PlayerToken • Has name field which is String type • Has PlayerToken token field • Has Square location field • Has Hand type hand field • Has a Player constructor that takes in two arguments: String name and Square startLocation • A method named getName() that should return player's name as entered in the beginning • A method named getSquare() that should return Square occupied by the player currently • A getHand() method that return the hand of cards for this player • A toString() method that returns the string representation of the player • A move() method that takes in newLocation of the Square type as an argument and moves the player to new location ensuring the new location is unoccupied and accessible and is not null	• Square • Hand • Position

Position	
• Has two int type fields x and y • Position constructor that takes in two parameters and sets x and y • Two getter methods getX() and getY() returns x and y positions	

Card	
• Has enum type Type that has the type of card i.e PlayerCard, WeaponCard, RoomCard • Has enum type Name for character names, weapon names and room names • Has field names name of type Name • Has field of type Type called type • Has field of boolean type thats called isSolution • Has cards as static Map type that hold Type as Key and List of Name and Value • Has a constructor Card to create a new card • has Getter methods: getCardType() and getCardName() that return card type and card name respectively • A method isSolution() of type boolean that should return if the card is a solution card or not • Static getDeck() method that returns the deck of cards in the order how its stores in the map • A static getSolution() method that holds the solution	

Die	Move
• Constructor takes in the number of dice to use • Field is 2D array of size that is the number of dice, containing an array of dice values • Should have a static final variable DICE_VALUES with 6 as constant value • Should have a roll() method • Roll the dice(get random values from array)	

Hand	
• Should have a variable that holds list of Cards • A constructor to create hand • A method to add cards in Hand	• Card

Move	
• Should have two variables: one of int type to store diceRoll, other a string type for move sequence • A move constructor that takes in two arguments for moveSequence and diceRoll • A method that checks Validity of a move • A method that return boolean and checks if there are still any moves left • A method to get position from current player position • Method to get the first move sequence	