

#	From	To	Type	Multiplicity (From → To)	Description
1	Deadwood	Player	Composition	1 → *	Deadwood owns a list of players.
2	Deadwood	Board	Association	1 → 1	Deadwood uses one Board.
3	Deadwood	Actions	Association	1 → 1	Deadwood uses one Actions.
4	Board	Room	Composition	1 → *	Board is composed of many rooms.
5	Board	Scene	Composition	1 → *	Board owns the scene deck.
6	Board	ParseXML	Dependency	1 → 1	Board depends on ParseXML to load data.
7	Room	Scene	Association	1 → 0..1	A room may have a current scene.
8	Room	Role	Composition	1 → *	A room composes its extra roles.
9	Room	Player	Association	1 → *	A room contains zero or more players.
10	Room	Room	Association	1 → *	Self-association: adjacent rooms.
11	Scene	Role	Composition	1 → *	A scene composes its starring roles.
12	Player	Room	Association	1 → 0..1	Player knows their current room.
13	Player	Role	Association	1 → 0..1	Player may have a current role.
14	Role	Player	Association	1 → 0..1	Role knows the assigned player.
15	Actions	Board	Association	1 → 1	Actions uses Board.
16	Actions	CastingOffice	Association	1 → 1	Actions uses CastingOffice.
17	Actions	Dice	Association	1 → 1	Actions uses Dice.
18	CastingOffice	PaymentType	Dependency	1 → 1	CastingOffice contains the PaymentType enum.

Room
Methods -String roomName -int totalShots -int remainingShots -Scene currentScene -List <Role> extraroles -List <Player> playersInRoom -List <Player> adjacentRooms -boolean isSet Attributes +Room(String, boolean) +Room(String, int, boolean) +getRoomName() :: String +isSet() :: boolean +getRemainingCounters() :: int +removeShotCounter() :: void +isSceneComplete() :: boolean +getCurrentScene(Scene) :: void +hasActiveScene() :: boolean +removeScene() :: void +addExtraRole(Role) :: void +getExtraRole() :: List<Role> +getAvaliableExtraRoles() :: List<Role> +getAllRoles() :: List<Role> +getAvailableRoles() :: List<Role> +getPlayersInRoom() :: List<Player> +addPlayer(Player) :: void +removePlayer(Player) :: void +getAdjacentRooms() :: List <Room> +isAdjacent(Room) :: boolean +addAdjacentRooms(Room) :: void +displayRoleOption(int) :: void +displaySetInfo() :: void +displayNeighbors() :: void

Role
Methods -String line -int requiredRank -String roleName -boolean isStar -Player assignedPlayer Attributes +Role(String, String, int, boolean) +getRoleName() :: String +getRequiredRank() :: int +isStarringRole() :: boolean +getAssignedPlayer() :: Player +setAssignedPlayer(Player) :: void +isOccupied() :: boolean +removePlayerFromRole() :: void +displayRole() :: void

Player
Methods -int credits -int credits -int dollars -int rank -int rehearsalTokens -Room currentRoom -Role currentRole -String playerName Attributes +Player(String, int, int, int, Room) +getPlayerName() :: String +getCredits() :: int +getDollars() :: int +getRank() :: int +getRehearsalTokens() :: int +getCurrentRoom() :: Room +getCurrentRole() :: Role +setCredits(int) :: void +setDollars(int) :: void +setRank(int) :: void +setRehearsalTokens(int) :: void +setCurrentRoom(Room) :: void +setCurrentRole(Role) :: void +isWorking() :: boolean +leaveRole() :: void +addRehearsalToken() :: void +resetTokens() :: void +addDollars(int) :: void +addCredits(int) :: void +playerScore() :: int +displayPlayerInfo() :: void

ParseXML
+readBoardData(Document): Map<String, Room> +readCardData(Document): List<Scene> +getDocFromFile(String) : Document

Deadwood
Methods -ArrayList <Player> players -Board board -Actions action -int playerIndex -int totalDays -int currentDay -Player currentPlayer -Room currentRoom -boolean gameOver -Scanner input Attributes +main(String[]): : void +setupGame: : void +runDeadwood() :: void +displayAllPlayersLocations() :: void +displayPlayerOptions(Players) :: void +playerNotworking() :: void +playerWorking() :: void +endOfDay() :: void +endOfGame() :: void

Board
Methods -Room Trailers -Room castingOffice -int activeScenes -int dayCount -int totalDays -Map<String, Room> rooms -List<Scene> sceneDeck Attributes +Board() +createBoard() :: void +shuffleDealScene() :: void +isDayOver :: boolean +isGameOver :: boolean +removeScene(Room) :: void +clearAllScene() :: void +setUpNewDay(List<Player>) :: void +moveToTrailer(List<Player>) :: void +moveToOffice(List<Player>) :: void +displayScore(List<Player>) :: void +getRoom(String) :: Room +getTrailers() :: Room +getCastingOffice() :: Room

Actions
Methods -Board board -CastingOffice castingOffice -Dice dice -Room currentRoom Attributes +Actions(board) +Move(Player, String) :: boolean +takeRole(Player, String) :: boolean +validateTakeRole(Player, Role) :: boolean +validateAct(Player) :: boolean +validateRehearse(Player) :: boolean +validateCanUpgrade(Player, int) :: boolean +rollDice() :: int +rollBonusDice(int) :: int[] +act(Player) :: void +Rehearse(Player) :: void +displayUpgradeOptions(Player) :: void +upgradeRank(Player, String, String) :: void +wrapScene(Room) :: void +payOut(Room, Scene) :: void

Dice
Methods -int value -int totalVal -int[] lastRolls -Random random Attributes +Dice(int) +roll() :: int +addRehearsalBonus(int, int) :: int +bonusDice(int) :: int[] +sortDesc(int[]) :: int[] +bonusDiceSorted(int) :: int[] +displaySortedRolls(int[]) :: void

Scene
Methods -String sceneDescription -int movieBudget -int sceneNumber -List<Role> starRoles -String sceneName Attributes +Scene(String, String, int, int, ArrayList<Role>) +getSceneName() :: String +getSceneDescription() :: String +getMovieBudget() :: int +getStarRoles() :: List<Role> +addStarRole(Role) :: void +resetScene() :: void +getAvailableStarRoles() :: List<Role> +displaySceneInfo() :: void

CastingOffice
Methods -int[] upgradeRankDollars -int[] upgradeRankCredits Attributes +CastingOffice() +canUpgrade(Player, int, PaymentType) :: void +upgradePlayer(Player, int, PaymentType) :: void +displayCosts() :: void +getDollarCost(int) :: int +getCreditCost(int) :: int +validateCanUpgrade(Player, int) :: boolean