MAIN

Driver

+ main(args: String[])

CONTROLLER

<<interface>>
Controller

+ start(): String

+ setView(view: View)

ControllerImpl

- world: World
- view: View
- addPlayerPieceCommand: AddPlayerPiece
- playerPickItem: PlayerPickItem
- createGraphicalWorld: CreateGraphicalWorld

+ ControllerImpl(world: World)

+ start(): String

+ setView(view: View)

+ notifyScreenChange()

+ notifyAddPlayerPiece(): String

+ notifyClick(int x, int y): String

+ notifyPickItem(): String

+ notifyPlayerAttack(): String

+ notifyMovePet(): String

+ exitGame()

These are the features available to the view!

<<interface>>
Features

+ notifyScreenChange()

+ notifyAddPlayerPiece(): String

+ notifyPickItem(): boolean

+ notifyClick(int x, int y): String

+ notifyPlayerAttack(): String

+ notifyMovePet(): String

+ exitGame()

uses

VIEW

View interface is how the Controller will see the view.

<<interface>>
View

+ setFeatures(feature: Feature)

+ renderOverlayPices()

+ getCurrentFrameNumber(): int

+ switchFrames()

+ getFileFromFileChooser(): String

+ showScreen()

+ showItemChooser() String

+ showRoomChooser() String

+ updateModel(world:ReadOnlyWorld)

+ extractAndSavePlayerDetails()

+ getPlayerDetails(): List<String>

+ showAttackItemChooser(): String

+ updateRoomNameClicked(roomName: String)

+ getRoomNameClicked(): String

+ getGamePlayPanelSize(): Dimension

+ refreshGamePlayScreen()

+ notifyViewGameOver()

renderOverlayPieces will get coordinates of all pieces of board from the controller. The view will simply render the pices at the co-ordinates mentioned by the read only model.

CurrentEventName will be sent to the Controller. On the basis of this name, the controller will decide if it needs additional data for the event (Eg. item name for attack command). For doing this; a pop up dialogue box will be rendered with a drop down menu of item list.

ViewImpl

- currentEventName: String
- moveResult: String
- welcomeFrame: WelcomeFrame
- addPlayerFrame: AddPlayerFrame
- gamePlayFrame: GamePlayFrame
- currentFrameNumber: int
- roomNameClicked: String
- model: ReadOnlyWorld
- features: Features

+ ViewImpl(world: ReadOnlyWorld)

+ setFeatures(feature: Features)

- updateCurrentEventName()

+ getCurrentFrameNumber(): int

+ showScreen()

+ renderOverlayPices()

+ switchFrames()

+ showItemChooser(): String

+ showRoomChooser() String

+ updateModel(world:ReadOnlyWorld)

+ getPlayerDetails(): List<String>

+ updateRoomNameClicked(roomName: String)

+ getRoomNameClicked(): String

+ extractAndSavePlayerDetails()

+ refreshGamePlayScreen()

+ showAttackItemChooser(): String

+ getGamePlayPanelSize(): Dimension

+ notifyViewGameOver()

<<interface>>
GameOverBox

+ renderGameResult()

GameOverBoxImpl

- exitButton : JButton
- restartButton : JButton
- gameResultMessage : JLabel

+ GameOverBoxImpl()
- getGameResult()
+ renderGameResult()
- closePopup(actionEvent: ActionEvent)

All these display methods will be called from the controller and view will decide how to display a component by appropriately defining these methods.

WelcomeTextPanel

- welcomeMessage: JLabel
- developerTitle: JLabel
- developerOne: JLabel
- developerTwo: JLabel

+ WelcomeTextPanel()

<<interface>>
WelcomeFrame

+ setFeatures(features: Features)

+ display()

+ setDisplay(display: Boolean)

+ updateModel(world:ReadOnlyWorld)

+ dispatchEventClose()

WelcomeFrameImpl

- features: Features
- model: ReadOnlyWorld
- welcomeTextPanel: WelcomeTextPanel
- fileChooser: JFileChooser
- existingSpectem: JMenuItem
- newSpectem: JMenuItem
- exitItem: JMenuItem

+ WelcomeFrameImpl(world: ReadOnlyWorld)

- createMenuBar: JMenuBar

+ setFeatures(features: Features)

+ getGile

+ display()

+ setDisplay(display: Boolean)

+ updateModel(world:ReadOnlyWorld)

+ dispatchEventClose()

<<interface>>
AddPlayerFrame

+ setFeatures(features: Features)

+ display()

+ setDisplay(display: Boolean)

+ updateModel(world:ReadOnlyWorld)

+ getPlayerDetails: List<String>

+ savePlayerDetails()

+ dispatchEventClose()

AddPlayerFrameImpl

- c: Container
- model: ReadOnlyWorld
- title: JLabel
- statusMessage: JLabel
- playerNameLabel: JLabel
- bagLimitLabel: JLabel
- playerNameInputField: JTextField
- roomNameLabel: JLabel
- playerTypeLabel: JLabel
- humanRadioButton: JRadioButton
- computerRadioButton: JRadioButton
- buttonGroup: JButtonGroup
- submitButton: JButton
- startGameButton: JButto
- bagLimitInputField: JTextField
- roomDropDown: JComboBox<String>
- savedPlayerDeatils: List<String>

+ AddPlayerFrameImpl(world: ReadOnlyWorld)

+ setFeatures(features: Features)

+ display()

+ setDisplay(display: Boolean)

+ updateModel(world:ReadOnlyWorld)

+ savePlayerDetails()

+ getPlayerDetails(): List<String>

+ dispatchEventClose()

GraphicWorldPanel

+ GraphicWorldPanel()

TurnInfoAndResultPanel

- turnInfoPanel: TurnInfoPanel
- turnResultPanel: TurnResultPanel
- scrollableTurnInfoPanel: JScrollPane
- scrollableTurnResultPanel: JScrollPane

+ TurnInfoAndResultPanel()

+ setTurnResultText(textResult: String)

+ setTurnInfoText(textInfo: String)

TurnResultPanel

- turnResultText: JLabel
- turnResultTitle: JLabel

+ TurnResultPanel()

+ setTurnResultText(textResult: String)

TurnInfoPanel

- turnInfoText: JLabel
- turnInfoTitle: JLabel

+ TurnInfoPanel()

+ setTurnInfoText(textInfo: String)

<<interface>>
GamePlayFrame

+ setFeatures(features: Features)

+ display()

+ setDisplay(display: Boolean)

+ updateModel(world:ReadOnlyWorld)

+ showItemChooser(): String

+ showRoomChooser(): String

+ showAttackItemChooser(): String

+ refreshGamePlayScreen()

+ dispatchEventClose()

GamePlayFrameImpl

- model: ReadOnlyWorld
- features: Features
- c: Container
- graphicalWorldPanel: GraphicalWorldPanel
- turnInfoAndResultPanel: TurnInfoAndResultPanel
- gamePlayerContainer: Container
- baseImage: BufferedImage
- picLabel: JLabel
- rootPanel: JPanel
- graphicalWorldWithScroll: JScrollPane

+ GamePlayFrame(world: ReadOnlyWorld)

+ refreshGamePlayScreen()

+ setFeatures(features: Features)

+ display()

+ setDisplay(display: Boolean)

+ showItemChooser(): String

+ showRoomChooser(): String

+ updateModel(world:ReadOnlyWorld)

+ showAttackItemChooser(): String

+ dispatchEventClose()

<<interface>>
GameOverBox

+ renderGameResult()

GameOverBoxImpl

- exitButton : JButton
- restartButton : JButton
- gameResultMessage : JLabel

+ GameOverBoxImpl()
- getGameResult()
+ renderGameResult()
- closePopup(actionEvent: ActionEvent)

MODEL

<<interface>>
ReadOnlyWorld

+ getTurnInformation(): String

+ getBaseImage(): BufferedImage

+ getPlayerDescription(): String

+ getRoomNames(): List<String>

+ getItemNamesInRoom(): List<String>

+ getItemNamesInBag(): List<String>

<<interface>>
World

+ addPlayerPiece(roomName: String, roomName: String, itemBagLimit: Integer, isHuman: boolean): String

+ movePlayerPiece(roomName: String): String

+ playerPickUpItem(itemName: String): String

+ playerLookAround(): String

+ getPlayerDescription(playerName: String: String

+ isCurrentTurnHuman(): boolean

+ getPlayerList(): List<Player>

+ getMaxTurns(): Integer

+ getRows(): Integer

+ getColumns(): Integer

+ getNumberOfTurns(): Integer

+ displayRoomInformation(roomName: String): String

+ playComputerPlayer(option: Integer, flagUseRandom: boolean): String

+ playerAttack(itemName: String): String

+ movePet(roomName: String): String

+ updateMoveResult()

+ updateInputSpecsFilePath(filePath: String)

+ generateBaseImage(scaleUpRow: int, scaleUpCol: int)

+ getTargetPiece(): Target

+ getPlayerList(): List<Player>

+ getRoomsList(): List<Room>

+ getItemNamesInBag(): List<String>

+ generateOverlayPieceCoordinates()

+ getCurrentPlayerRoomName(): String

+ getRoomNameFromCoordinates(): String

WorldInitializer

- fileReader: Readable
- worldInitializer: WorldInitializer
- area51: World
- maxTurns: int

- WorldInitializer(fileReader: Readable, maxTurns: int)

+ getInstance(fileReader: Readable, maxTurns: int): WorldInitializer

+ generateWorld(): World

+ updateFileReader(fileReader: Readable)

<<interface>>
Room

+ calculateNeighborsAndSave(roomsList: List<Room>)

+ getNeighbors(): List<Room>

+ getItemList(): List<Item>

+ getLeftTopRow(): int

+ getLeftTopCol(): int

+ getRoom(): int

+ getRightTopRow(): int

+ getRightTopCol(): int

+ getName(): String

+ addPlayerToRoom(Player player)

+ removePlayerFromRoom(Player player)

+ removeItemFromRoom(Item: Item)

+ getPlayersInRoom(): List<Player>

+ statusChangePetInRoom(isInRoom: boolean)

+ getIsPetPresent(): boolean

RoomImpl

- number: int
- name: String
- leftTopRow: int
- leftTopCol: int
- rightBottomRow: int
- rightBottomCol: int
- neighborRooms: List<Room>
- itemsList: List<Item>
- globalItemsList: List<Item>
- globalRoomsList: List<Room>
- playersInRoom: List<Player>
- isPetPresent: boolean

+ Room(number: int, name: String, leftTopRow: int, leftTopCol: int, rightBottomRow: int, rightBottomCol: int, itemList: List<Item>)

+ calculateNeighborsAndSave(roomsList: List<Room>)

- generateItemList(itemList: List<Item>)

+ getNeighbors(): List<Room>

+ getItemList(): List<Item>

+ getLeftTopRow(): int

+ getLeftTopCol(): int

+ getRoom(): int

+ getRightTopRow(): int

+ getRightTopCol(): int

+ getName(): String

+ equals(o: Object): boolean

+ hashCode(): int

+ toString(): String

+ addPlayerToRoom(Player player)

+ removePlayerFromRoom(Player player)

+ removeItemFromRoom(item: Item)

+ getPlayersInRoom(): List<Player>

+ statusChangePetInRoom(isInRoom: boolean)

+ getIsPetPresent(): boolean

<<interface>>
Command

+ execute(world: World): String

CreateGraphicalWorld

- view: View
- newWorld: World

+ CreateGraphicalWorld(view: View)
+ getNewWorld(): World
+ execute(world: World): String

AddPlayerPiece

- view: View

+ AddPlayerPiece(view: View)

+ execute(world: World): String

Attack

- view: View

+ PlayerAttack(view: View)

+ execute(world: World): String

PlayerMove

- view: View

+ PlayerMove(view: View)

+ execute(world: World): String

MovePet

- view: View

+ MovePet(view: View)

+ execute(world: World): String

DisplayPlayerDescription

- view: View

+ DisplayPlayerDescription(view: View)

+ execute(world: World): String

PlayerLookAround

- view: View

+ PlayerLookAround(view: View)

+ execute(world: World): String

PlayerPickItem

- view: View

+ PlayerPickItem(view: View)

+ execute(world: World): String

<<interface>>
Pet

+ getCurrentRoom(): Room

+ getName(): String

+ movePet(room: Room): String

+ movePetDFS(): Room

PetImpl

- name: String
- currentRoom: Room
- roomsList: List<Room>

+ PetImpl(name: String, currentRoom: Room, roomsList: List<Room>)

+ getCurrentRoom(): Room

+ getName(): String

+ movePet(room: Room): String

+ movePetDFS(): Room

WorldImpl

- rows: Integer
- columns: Integer
- worldName: String
- roomsList: List<Rooms>
- itemList: List<Item>
- playersList: List<Player>
- targetPiece: Player
- maxTurns: Integer
- numberOfTurns: Integer
- currentTurnLisIndex: Integer
- pet: Pet
- imageDisplayer: ImageDisplayer
- overlayPieceCoordinates: List<List<Integer>>
- moveResult: String
- inputSpecsFilePath: String
- baseImage: BufferedImage
- scaleUpRow: int
- scaleUpCol: int

+ World(rows:Integer, columns:Integer, worldName:String, roomsList:List<Room>, itemList: List<Item>, targetName:String, targetHealth:Integer, petName: String, maxTurns: Integer)

- doRoomsOverlap(room1: Room, room2: Room): boolean

- areItemsRoomIndicesValid(): boolean

- allRoomsLesserDimThanWorld(): boolean

- paintPlayerOnBaseImage()

+ getTargetPiece() : Target

+ getRows(): Integer

+ getRoomsList(): List<Rooms>

+ getCols(): Integer

+ getWorldName(): String

+ toString(): String

+ updateMoveResult()

+ updateInputSpecsFilePath(filepath: String)

+ getTurnInformation(): String

+ addPlayerPiece(name: String, roomName: String, itemBagLimit: int, isHuman: boolean): String

+ movePlayerPiece(roomName: String): String

+ playerPickUpItem(itemName: String): String

+ playerLookAround(): String

+ displayRoomInformation(roomName: String): String

+ playComputerPlayer(option: Integer, flagUseRandom: boolean) : String

+ getPlayerDescription(playerName: String): String

+ getPlayerList(): List<Player>

+ isCurrentTurnHuman(): boolean

+ getMaxTurns(): Integer

+ getNumberOfTurns(): Integer

+ playerAttack(itemName: String): String

- canSeePlayer(player1: Player, player2: Player): Boolean

+ movePet(roomName: String): String

+ generateBaseImage(scaleUpRow: int, scaleUpCol: int)

+ getBaseImage(): BufferedImage

+ getRoomNames(): List<String>

+ getItemNamesInBag(): List<String>

+ getItemNamesInRoom(): List<String>

+ generateOverlayPieceCoordinates()

+ getCurrentPlayerRoomName(): String

+ getRoomNameFromCoordinates(): String

<<interface>>
Item

+ getContainingRoomIndex(): int

+ getName(): String

+ getId(): int

+ getDamage(): int

ItemImpl

- id: int
- name: String
- damage: int
- containingRoomIndex: int

+ Item(id: int, name: String, damage: int, containingRoomIndex: int)

+ getContainingRoomIndex(): int

+ getName(): String

+ getId(): int

+ getDamage(): int

+ toString(): String

<<interface>>
Player

+ getCurrentRoom(): Room

+ getName(): String

+ getNumber(): int

+ getIsHuman(): boolean

+ getItemList(): List<Item>

+ pickItem(item: Item): boolean

+ getSpiteLevel(): int

+ getItemBagLimit(): int

+ lookAround(): String

+ move(room: Room): String

+ removeItemFromBag(item: Item): boolean

PlayerImpl

- itemList: List<Item>
- currentRoom: Room
- spiteLevel: int
- itemBagLimit: int
- isHuman: boolean
- name: String
- number: int

+ Player(number: int, name: String, currentRoom: Room, itemBagLimit: int, isHuman: boolean)

+ getItemList(): List<Item>

+ getCurrentRoom(): Room

+ pickItem(item: Item): boolean

+ getSpiteLevel(): int

+ lookAround(): String

+ getIsHuman(): boolean

+ getItemBagLimit(): int

+ move(room: Room): String

+ getName(): String

+ getNumber(): int

+ toString(): String

+ removeItemFromBag(item: Item): boolean

<<interface>>
Target

+ getCurrentRoom(): Room
+ getName(): String
+ getNumber(): int
+ getHealth(): int
+ move()
+ receiveAttack(damage: int)

TargetImpl

- name: String
- number: int
- currentRoom: Room
- roomList: List<Room>
- health: int

+ Target(number: int, name: String, health: int, roomsList: List<Room>)

+ getHealth(): int

+ getCurrentRoom(): Room

+ move()

+ getName(): String

+ getNumber(): int

+ toString(): String

+ receiveAttack(damage: int)

ImageDisplayer

- bufferedImage: BufferedImage
- graphics: Graphics
- scaleUpRow: int
- scaleUpCol: int
- world: World
- roomsList: List<Room>

+ ImageDisplayer(world: World, scaleUpRow: int, scaleUpCol: int)

- paintComponent(g: Graphics, scaleUpRow: int, scaleUpCol: int)

+ draw(scaleUpRow: int, scaleUpCol: int): BufferedImage

GenerateRandom

- randomInt: Integer
- random: Random

+ GenerateRandom()
+ GenerateRandom(randomInt: int)
+ getRandomInt(range: int)

contains

initializes

uses

Changes to the model to separate it from the text based controller.

Our model was highly decoupled from the controller. The entire game logic was encapsulated in the model itself.

So, when we separated the model from the console based text controller; no changes were required in the model.

Validation that no changes were required: If we plug in the console based text controller to the model using the previous interface; the program will still work as before.

However it is noteworthy to mention that we were required to add some methods to the model in order to integrate it with the new controller. For example a method to retrieve a room from the click coordinates received from the view.

Thus lowest possible coupling between model and controller helped us separate the model with no changes to existing code base.