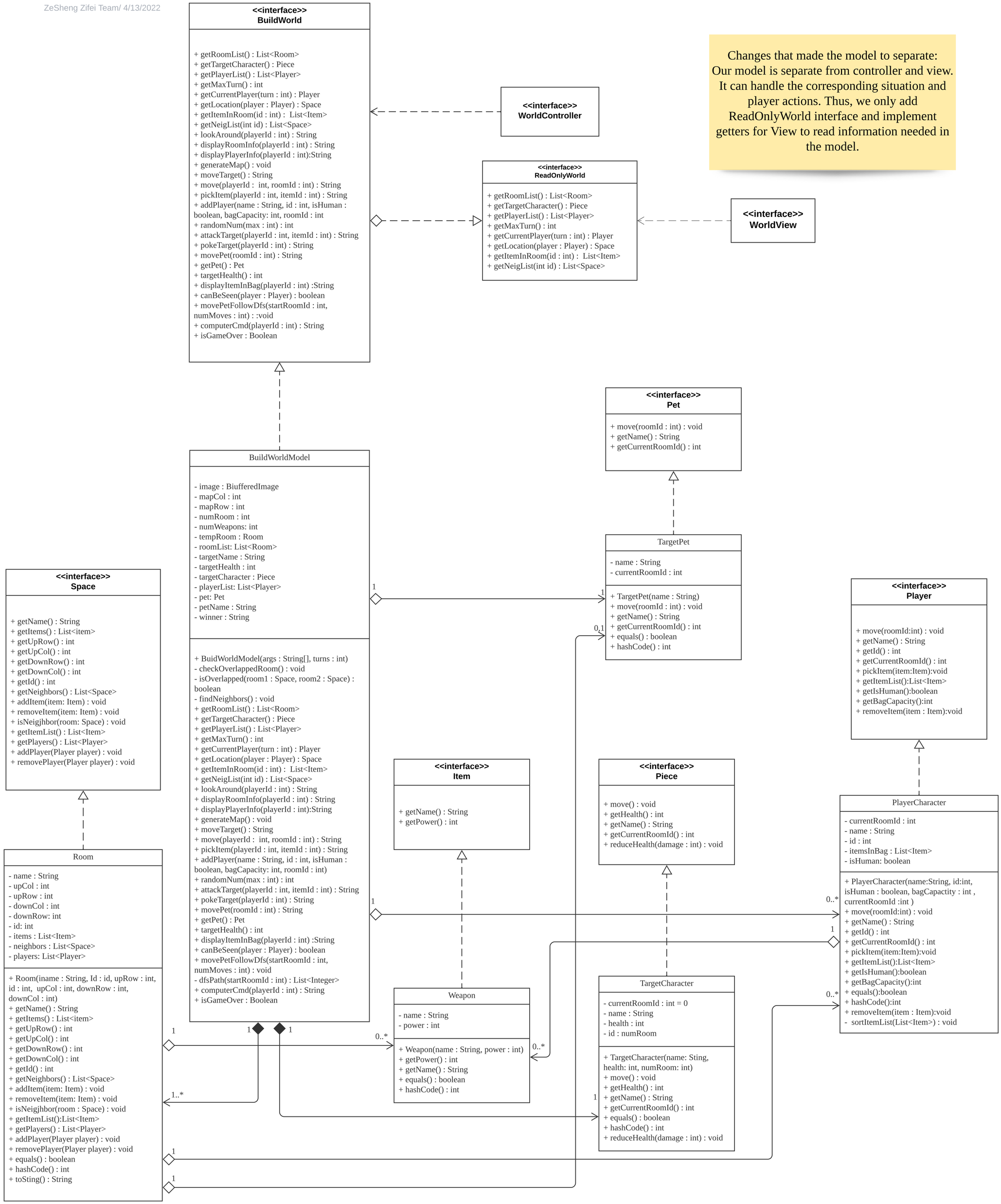
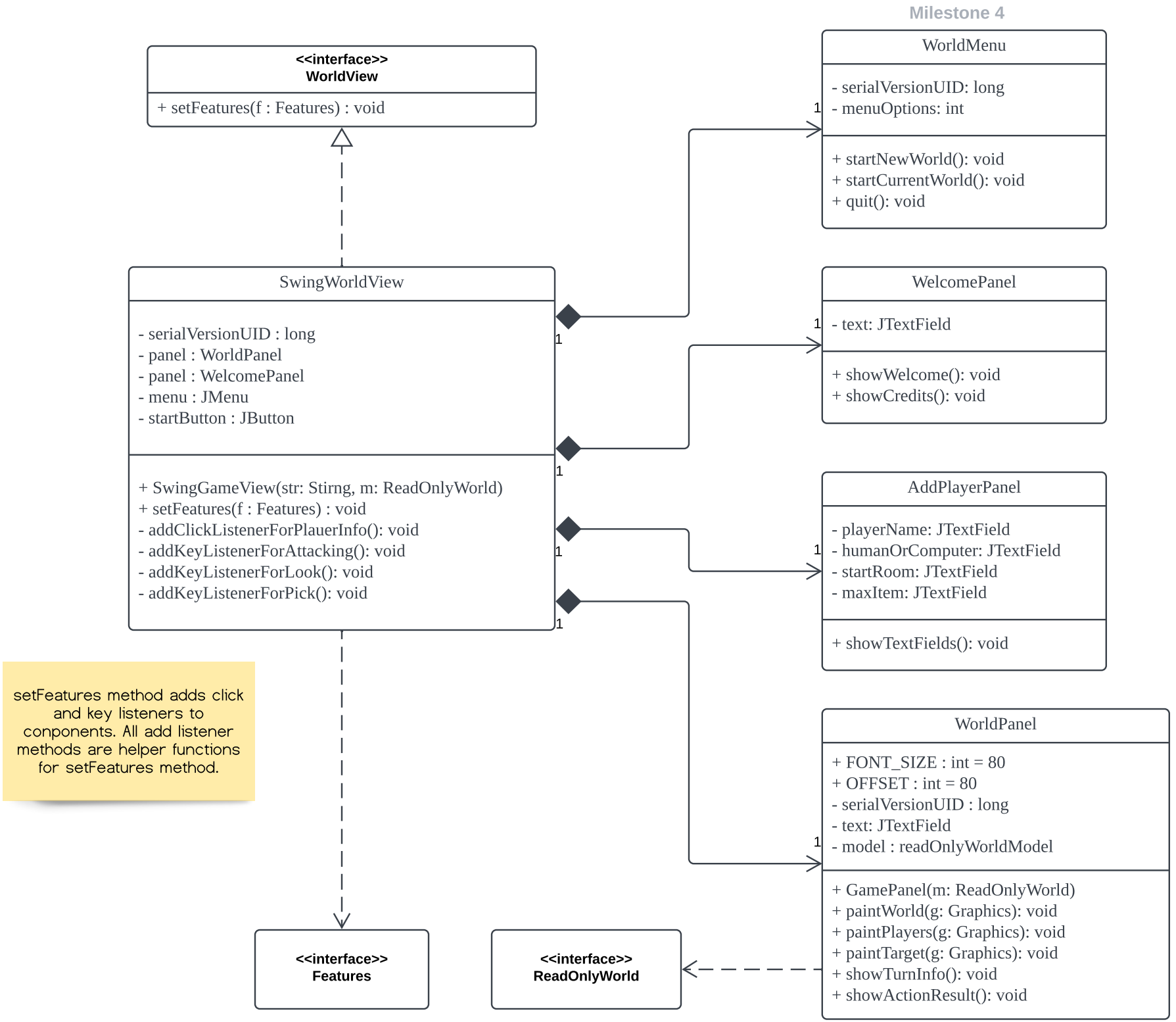
Milestone 4

ZeSheng Zifei Team/ 4/13/2022




<<interface>>
WorldView
+ setFeatures(f : Features) : void
WorldMenu
- serialVersionUID: long
- menuOptions: int
+ startNewWorld(): void
+ startCurrentWorld(): void
+ quit(): void
SwingWorldView
- serialVersionUID : long
- panel : WorldPanel
- panel : WelcomePanel
- menu : JMenu
- startButton : JButton
+ SwingGameView(str: Stirng, m: ReadOnlyWorld)
+ setFeatures(f : Features) : void
- addClickListenerForPlauerInfo(): void
- addKeyListenerForAttacking(): void
- addKeyListenerForLook(): void
- addKeyListenerForPick(): void
WelcomePanel
- text: JTextField
+ showWelcome(): void
+ showCredits(): void
AddPlayerPanel
- playerName: JTextField
- humanOrComputer: JTextField
- startRoom: JTextField
- maxItem: JTextField
+ showTextFields(): void
WorldPanel
+ FONT_SIZE : int = 80
+ OFFSET : int = 80
- serialVersionUID : long
- text: JTextField
- model : readOnlyWorldModel
+ GamePanel(m: ReadOnlyWorld)
+ paintWorld(g: Graphics): void
+ paintPlayers(g: Graphics): void
+ paintTarget(g: Graphics): void
+ showTurnInfo(): void
+ showActionResult(): void
1
setFeatures method adds click and key listeners to conponents. All add listener methods are helper functions for setFeatures method.
<<interface>>
Features
<<interface>>
ReadOnlyWorld

Milestone 4

ZeSheng Zifei Team/ 4/13/2022

**SwingWorldView**

**<<interface>>**
**WorldCommand**

+ execute(m : BuildWorld)
+ showView(view: View)

**<<interface>>**
**WorldController**

+ play(m : BuildWorld, view: View);

**<<interface>>**
**Features**

+ addPlayer(name : String. id : int, isHuman : boolean, bagCapacity : int, roomId : int)
+ displayRoomInfo(roomId : int)
+ displayPlayerInfo(playerId : int)
+ move(playerId : int, roomId : int)
+ PickItem(playerId : int, roomId : int)
+ generateMap()
+ moveTarget()
+ lookAround(playerId : int)
+ attack(playerId : int, itemId : int)
+ movePet(roomId : int)
+ startNewWorld(path: String, maxTurn: int): void
+ startCurrentWorld(maxTurn: int): void
+ quit(): void

**Move**

- roomId : int
- playerId : int

+ Move(playerId : int, roomId : int)
+ execute(m : BuildWorldModel)
+ showView(view: View)

**PickItem**

- roomId : int
- playerId : int

+ PickItem(playerId : int, roomId : int)
+ execute(m : BuildWorldModel)
+ showView(view: View)

**SwingController**

- model: BuildWorld
- view: WorldView

+ SwingController(model, view)
+ play(m : BuildWorld, view: View)
+ setView() : void
+ addPlayer(name : String. id : int, isHuman : boolean, bagCapacity : int, roomId : int)
+ displayRoomInfo(roomId : int)
+ displayPlayerInfo(playerId : int)
+ move(playerId : int, roomId : int)
+ PickItem(playerId : int, roomId : int)
+ generateMap()
+ moveTarget()
+ lookAround(playerId : int)
+ attack(playerId : int, itemId : int)
+ movePet(roomId : int)
+ startNewWorld(): void
+ startCurrentWorld(): void
+ quit(): void

**DisplayRoomInfo**

- roomId : int

+ DisplayRoomInfo(roomId : int)
+ execute(id : player, m : BuildWorldModel)
+ showView(view: View)

**GenerateMap**

+ GenerateMap()
+ execute(m : BuildWorldModel)
+ showView(view: View)

**MoveTarget**

+ MoveTarget()
+ execute(m : BuildWorldModel)
+ showView(view: View)

**DisplayPlayerInfo**

- playerId : int

+ DisplayPlayerInfo(playerId : int)
+ execute(id : player, m : BuildWorldModel)
+ showView(view: View)

**LookAround**

- playerId : int

+ LookAround(playerId : int)
+ execute(id : player, m : BuildWorldModel)
+ showView(view: View)

**AddPlayer**

- name : String
- id : int
- isHuman : boolean
- bagCapacity : int
- roomId : int

+ AddPlayer(name : String. id : int, isHuman : boolean, bagCapacity : int, roomId : int)
+ execute(m : BuildWorldModel)
+ showView(view: View)

**Attack**

- playerId : int
- itemId : int
- withItem : boolean

+ Attack(playerId : int, itemId : int)
+ Attack(playerId : int)
+ execute(m : BuildWorldModel)
+ showView(view: View)

**MovePet**

- playerId : int
- roomId : int

+ MovePet(roomId : int)
+ execute(m : BuildWorldModel)
+ showView(view: View)

1 — 1

**<<interface>>**
**WorldView**

1 — 1

**<<interface>>**
**BuildWorld**

WelcomePage

Menu

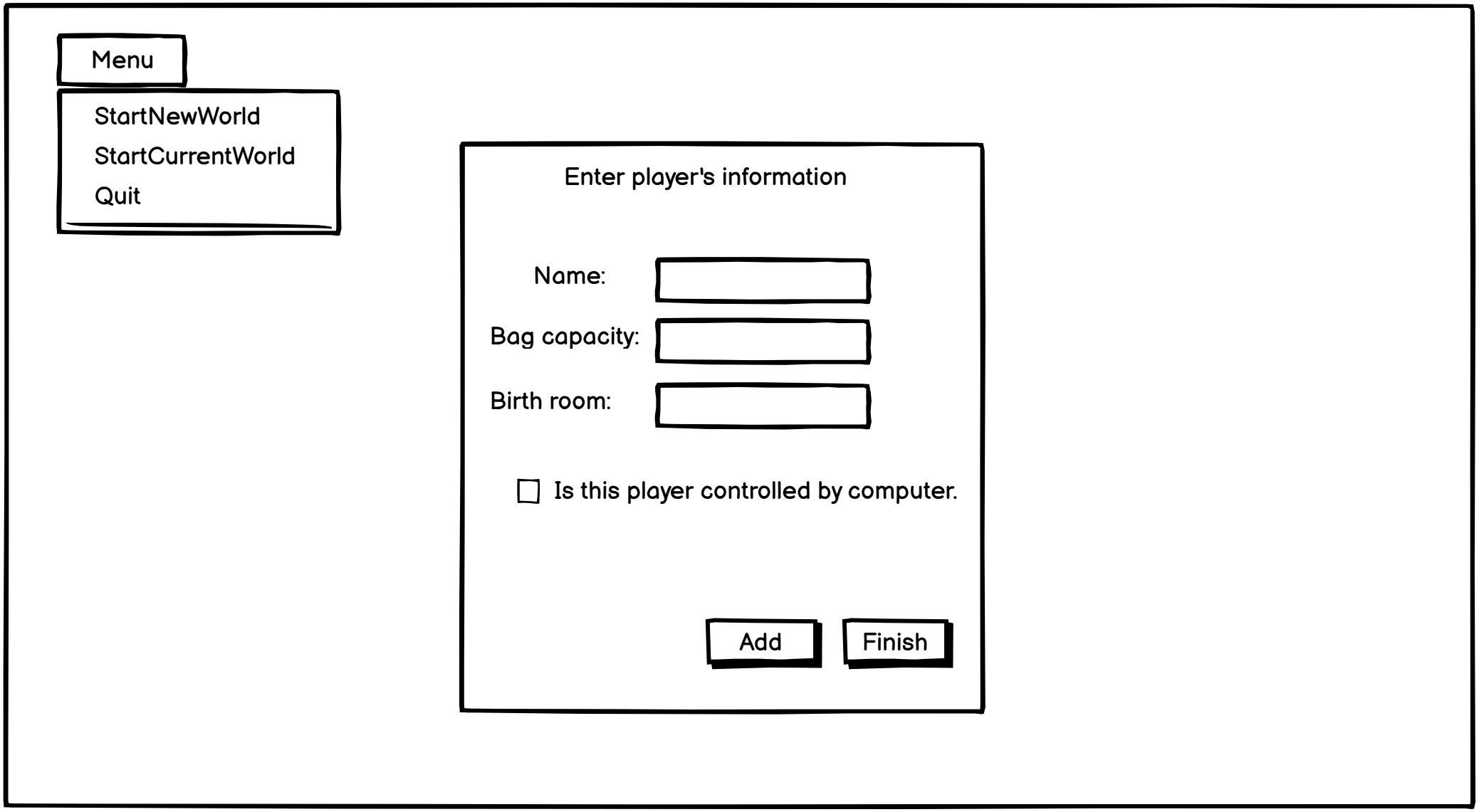
StartNewWorld

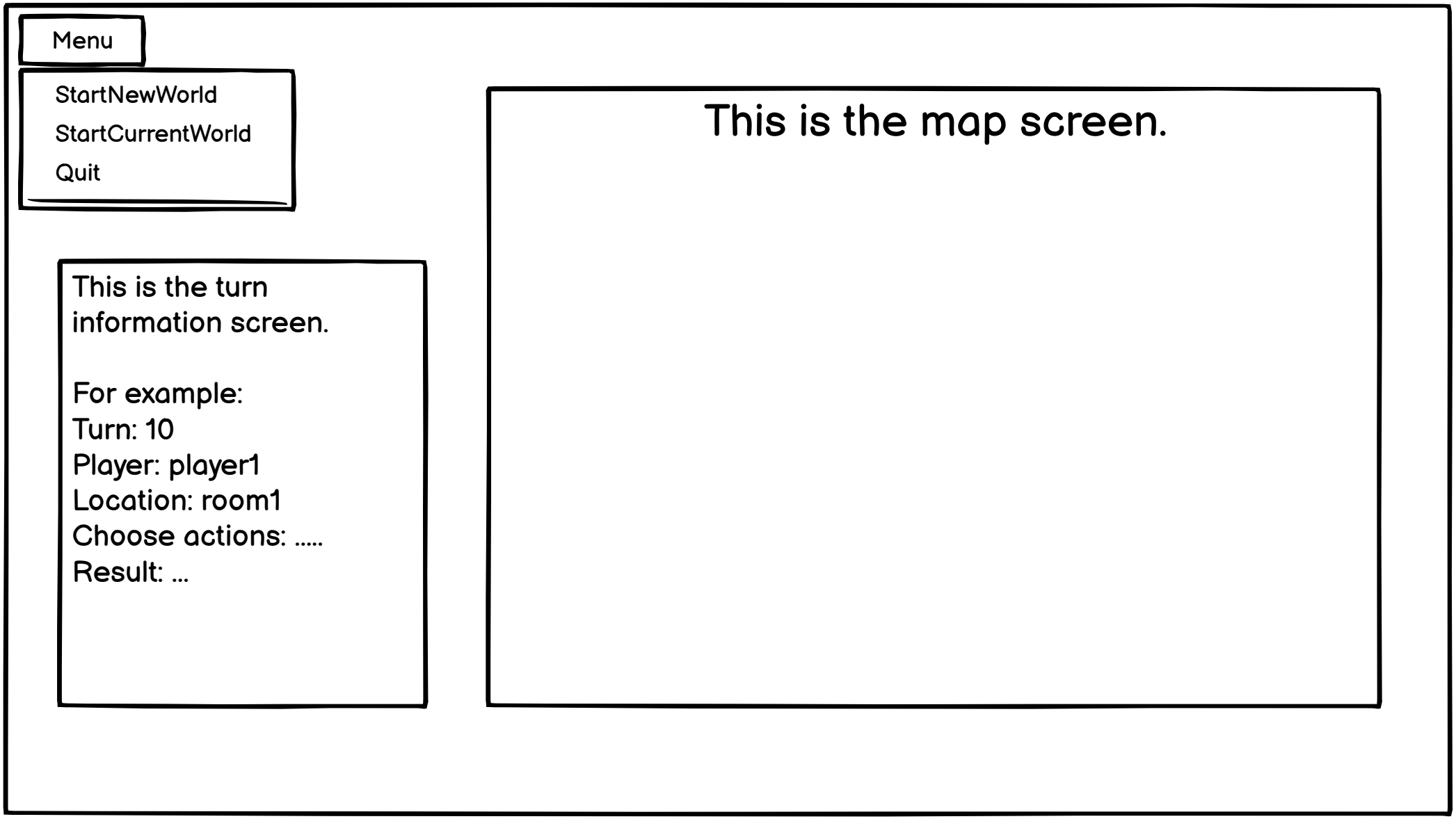
StartCurrentWorld

Quit

# Welcome to Kill Doctor Lucky Game!

Credits:
Zifei Song
Zesheng Li


Menu
StartNewWorld
StartCurrentWorld
Quit
Enter player's information
Name:
Bag capacity:
Birth room:
Is this player controlled by computer.
Add
Finish


Menu
StartNewWorld
StartCurrentWorld
Quit
This is the turn information screen.
For example:
Turn: 10
Player: player1
Location: room1
Choose actions: .....
Result: ...
This is the map screen.

Team members:
ZeSheng Li
Zifei Song

Milestone 4 - Preliminary Design Test Plan

Test controller using mock view and mock model

| Test case | Input | expected |
|---|---|---|
| Test move with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new Move(1,0);<br>//execute the command | log -> “Model: player 1 moves to room 0”<br>“View: player 1 moves to room 0 on the map” |
| Test look around with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new LookAround(1);<br>//execute the command | log -> “Model: player 1 is looking around”<br>“View: player 1 is looking around on the map” |
| Test attack with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new Attack(1,knife);<br>//execute the command | log -> “Model: player 1 makes attempt on target’s life with item knife”<br>“View: Model: player 1 makes attempt on target’s life with item knife on the map” |
| Test poke target with mock | //mock view<br>WorldView mockView = new | log -> “Model: player 1 tries to poke target in eyes” |

| | | |
|---|---|---|
| | SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new Attack(1);<br>//execute the command | "View: player 1 tries to poke target in eyes on the map" |
| Test pick item with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new Pick(1,knife);<br>//execute the command | log -> "Model: player 1 picks up the item knife"<br>"View: player 1 picks up the item knife on the map" |
| Test movePet with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new MovePet(1,15);<br>//execute the command | log -> "Model: player 1 moves pet to room 15"<br>"View: player 1 moves pet to room 15 on the map" |
| Test addPlayer with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new AddPlayer(A, 0, 1, 5, 15);<br>//execute the command | log -> "Model: player name: A, player id: 0, player type: human, player bag size: 5, birth room id: 15"<br>"View: player name: A, player id: 0, player type: human, player bag size: 5, birth room id: 15, player is added into the map" |
| Test generate map with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model | log -> "Model: buffedImage map is generated"<br>"View: buffedImage map |

| | | |
|---|---|---|
| | BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new GenerateMap();<br>//execute the command | is generated” |
| Test target move each round with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new MoveTarget();<br>//execute the command | log -> “Model: target moves to next room”<br>“View: target moves to next room on the map” |
| Test display player info with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new DisplayPlayerInfo(1);<br>//execute the command | log -> “Model: player 1’s info displayed”<br>“View: player 1’s info displayed on the panel” |
| Test display room info with mock | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>cmd = new DisplayRoomInfo(1);<br>//execute the command | log -> “Model: display the info of player 1’s located room”<br>“View: User clicks on player on the map” |
| Test quit game feature in JMenu | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15); | log -> “View: User clicks on game quit.” |

| | | |
|---|---|---|
| | WorldController c = new SwingController (mockModel, mockView)<br>c.quit(); | |
| Test start new model in JMenu | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>// "./mansion.txt" is the path of the file, and 15 is the max turn.<br>c.startNewWorld("./mansion.txt, 15"); | log -> "View: User clicks on start new world. Model: User gives the path is 'mainsion.txt', the max turn is 15." |
| Test start current model in JMenu | //mock view<br>WorldView mockView = new SwingWorldView(log, 15);<br>//mock model<br>BuildWorld mockModel = new MockModel(log, 15);<br>WorldController c = new SwingController (mockModel, mockView)<br>// 15 is the max turn.<br>c.startCurrentWorld(15); | log -> "View: User clicks on start current world. Model: User gives the max turn is 15." |

Model:

Pet interface:
TargetPet Class:

| Test case | Input | expected |
|---|---|---|
| Valid constructor get name | TargetPet("cat) | "cat" |
| Constructor with null String | TargetPet(null) | IAE |
| Test valid single move | Pet cat = new TargetPet("cat")<br>cat.move(2) | room Id: 0 -> 2 |

| | | |
|---|---|---|
| Test valid moves | Pet cat = new TargetPet(“cat”)<br>cat.move(2);<br>cat.move(3); | room Id: 0 -> 2<br>2 -> 3 |
| Test move to a room with negative id | Pet cat = new TargetPet(“cat”)<br>cat.move(-10); | IAE |
| Test move pet to a room with id exceed max room id | cat.move(10000) | IAE |
| Test if pet is born in space 0 | cat.getCurrentRoomId(); | 0 |
| Test if pet is moving automatically according to the algorithm<br>// assume correct order is 0 -> 9 -> 15 | Pet cat = new TargetPet(“cat”)<br>turn 1:<br>cat.getCurrentRoomId();<br>turn 2:<br>cat.getCurrentRoomId();<br>turn 3:<br>cat.getCurrentRoomId(); | 0<br>9<br>15 |

BuildWorld interface
BuildWorldModel class:
Setup:
BuildWord m = new BuildWorldModel(args);

| Test case | Input | expected |
|---|---|---|
| Test displayTarget position | m.displayTargetPosition(); | “Target position: room 0.” |
| Test displayTarget position after move target | m.displayTargetPosition();<br>m.moveTarget();<br>m.displayTargetPosition(); | “Target position: room 0.”<br>“Target position: room 1.” |
| Test attack target with item and no other players can see current player | targetHealth = 50<br>knife = new Weapon(“knife”, 4)<br>m.attack(1, knife);<br>target.getHealth(); | target health:<br>50 -> 46 |
| Test attack target with no item(poke target in eye) and no other players can see current player | targetHealth = 50<br>m.attack(1, null);<br>target.getHealth(); | target health:<br>50 -> 49 |

| | | |
|---|---|---|
| Test attack target with item but other players can see current player | targetHealth = 50<br>knife = new Weapon(“knife”, 4)<br>m.attack(1, knife);<br>target.getHealth(); | target health:<br>50 -> 50<br>“Attack failed” |
| Test attack target with no item(poke target in eye) but other players can see current player | targetHealth = 50<br>m.attack(1, null);<br>target.getHealth(); | target health:<br>50 -> 50<br>“Attack failed” |
| Test attack target when target is killed | targetHealth = 1<br>m.attack(1, null);<br>m.attack(1, null); | IAE |
| Test player move pet to a room | player id -> 1<br>pet room id -> 0<br>neighbor list of current room:<br>{0: bedroom (id = 0), 1: bathroom (id = 1)}<br>m.movePet(1, 0);<br>pet.getCurrentRoomId(); | 1 |
| Test player tries to move pet to a non-existing room | player id -> 1<br>pet room id -> 0<br>neighbor list of current room:<br>{0: bedroom (id = 0), 1: bathroom (id = 1)}<br>m.movePet(1, 1000); | IAE |
| Test move pet to a room with negative id | player id -> 1<br>pet room id -> 0<br>neighbor list of current room:<br>{0: bedroom (id = 0), 1: bathroom (id = 1)}<br>m.movePet(1, -1); | IAE |
| Test player lookAround when pet is in a neighbor room | player a’s id -> 1<br>current room -> bedroom<br>item in room{gun (damage: 3)}<br>neighbor list {bathroom, playroom}<br>players in bedroom{a}<br>players in bathroom{}<br>players in playroom{}<br>pet position -> bedroom<br>m.lookAround(1); | “a is looking around.<br>a is in the bedroom<br>neighbor rooms: 0: bathroom 1: playroom<br>a can see these items in bedroom: 0: gun(damage:3)<br>These players are in the bedroom: a<br>**Pet blocks player's vision when looking through bathroom**<br>No players are in the playroom<br>No items are in the playroom<br>Player doesn't see the target.” |

| | | |
|---|---|---|
| Test player lookAround when pet is in the same room with player | player a's id -> 1<br>current room -> bedroom<br>item in room{gun (damage: 3)}<br>neighbor list {bathroom, playroom}<br>players in bedroom{a}<br>players in bathroom{}<br>players in playroom{}<br>pet position -> bedroom<br>m.lookAround(1); | "a is looking around.<br>a is in the bedroom<br>neighbor rooms: 0: bathroom 1: playroom<br>a can see these items in bedroom: 0: gun(damage:3)<br>These players are in the bedroom: a<br>No players are in the bathroom<br>No items are in the bathroom<br>No players are in the playroom<br>No items are in the playroom<br>Player doesn't see the target." |
| Test getPet() | Pet cat = new Pet("cat");<br>m.getPet(); | cat |

| | | |
|---|---|---|
| test Add Player | bwm.addPlayer("a", 0, true, 5, 0);<br>bwm.addPlayer("b", 1, false, 3, 0); | assertEquals(new PlayerCharacter("a", 0, true, 5, 0), bwm.getPlayerList().get(0));<br>assertEquals(new PlayerCharacter("b", 1, false, 3, 0), bwm.getPlayerList().get(1)); |
| test Duplicate Player Name | bwm.addPlayer("Bob", 0, true, 5, 0);<br>bwm.addPlayer("Bob", 1, false, 3, 0); | IAE |
| test Player RoomId Out Of Range | bwm.addPlayer("Bob", 1, false, 3, 35); | IAE |
| test Negative Player Bag Capacity | bwm.addPlayer("Bob", 1, false, -3, 5); | IAE |
| test Pick Item | bwm.addPlayer("Bob", 0, true, 5, 0);<br>Player a = bwm.getPlayerList().get(0);<br>Item item = bwm.getItemInRoom(0).get(0);<br>String str = bwm.pickItem(0, 0); | "Player picks up Shower nozzle\n" |
| test Pick Item if item is removed from room | bwm.addPlayer("Bob", 0, true, 5, 0);<br>Player a = bwm.getPlayerList().get(0);<br>Item item = bwm.getItemInRoom(0).get(0);<br>String str = bwm.pickItem(0, 0); | assertFalse(bwm.getRoomList().get(0).getItems().contains(item)); |
| test Pick Item When Bag Is Full | //player bag size is 1, make player pick two items<br>bwm.addPlayer("Bob", 0, true, 1, 0);<br>//player picks up number 0 item in room's item list<br>bwm.pickItem(0, 0); | IAE |

| | | |
|---|---|---|
| | bwm.move(0, 1);<br>//player tries to pick up number 0 item in room's item list<br>bwm.pickItem(0, 0); | |
| test Pick Item When No Items In Room | //player bag size is 1, make player pick two items<br>bwm.addPlayer("Bob", 0, true, 1, 0);<br>//player picks the only item in space 0<br>bwm.pickItem(0, 0);<br>//no item in space and call pickItem again<br>bwm.pickItem(0, 0); | IAE |
| test Pick Item That Does Not Exist In Room | bwm.addPlayer("Bob", 0, true, 1, 0);<br>//player is in space 0 that only contains 1 item.<br>//try to pick item with id 5 in the item list of the room which does not exist.<br>bwm.pickItem(0, 5); | IAE |
| test Move Player | bwm.addPlayer("Bob", 0, true, 5, 0);<br>Player a = bwm.getPlayerList().get(0);<br>//move player from room 0 to room 1, room 0 and room 1 are next to each other<br>String str = bwm.move(0, 1);<br>assertEquals(1, a.getCurrentRoomId());<br>assertEquals("Bob moves to Bedroom\n", str);<br>//test if player is added to new room<br>assertTrue(bwm.getRoomList().get(1).getPlayers().contains(a));<br>//test if player is removed from previous room | assertFalse(bwm.getRoomList().get(0).getPlayers().contains(a));<br>); |
| test Invalid Move | bwm.addPlayer("Bob", 0, true, 5, 0);<br>//try to move player to a room that is not a neighbor of current room<br>bwm.move(0, 3); | IAE |
| test Invalid Negative Move | bwm.addPlayer("Bob", 0, true, 5, 0);<br>//try to move player to a room with negative id<br>bwm.move(0, -3); | IAE |
| test Invalid Move Exceed Size | bwm.addPlayer("Bob", 0, true, 5, 0);<br>//try to move player to a room that does not exist.<br>bwm.move(0, 100); | IAE |
| test Look Around | bwm.addPlayer("Bob", 0, true, 5, 0); | Bob is looking around.\n" |

| | | |
|---|---|---|
| With Items | String str = bwm.lookAround(0); | Bob is in the Bathroom\n"<br>Bob can see these items in room: 0: Shower nozzle(damage:2)\n"<br>These players are in the room: Bob\n"<br>"Bob can see these rooms: 0: Bedroom 1: Gym \n"<br>Target is in the same room with player.\n" |
| test Look Around With No Items | bwm.addPlayer("Bob", 0, true, 5, 6);<br>String str = bwm.lookAround(0); | "Bob is looking around.\n"<br>Bob is in the Doll room\n"<br>Bob can see these items in room: There is no items in this room.\n"<br>These players are in the room: Bob\n"<br>Bob can see these rooms: 0: Baby's room 1: Living room 2: Foyer \n"<br>Player doesn't see target in current room.\n" |
| test Look Around With Players | bwm.addPlayer("Bob", 0, true, 5, 6);<br>bwm.addPlayer("Amy", 1, true, 5, 6);<br>String str = bwm.lookAround(0); | "Bob is looking around.\n"<br>Bob is in the Doll room\n"<br>Bob can see these items in room: There is no items in this room.\n"<br>These players are in the room: Bob Amy\n"<br>Bob can see these rooms: 0: Baby's room 1: Living room 2: Foyer \n"<br>Player doesn't see target in current room.\n" |
| test Look Around With Target | bwm.addPlayer("Bob", 0, true, 5, 0);<br>String str = bwm.lookAround(0); | "Bob is looking around.\n"<br>Bob is in the Bathroom\n"<br>Bob can see these items in room: 0: Shower nozzle(damage:2)\n"<br>These players are in the room: Bob\n"<br>Bob can see these rooms: 0: Bedroom 1: Gym \n"<br>Target is in the same room with player.\n" |
| test Invlid Look | bwm.lookAround(-1); | IAE |

| | | |
|---|---|---|
| Around Negative Id | | |
| test Invlid Look Around Id Exceed Size | bwm.addPlayer("Bob", 0, true, 5, 0);<br>bwm.lookAround(2); | IAE |
| test Display Player Info | //player with no item<br>bwm.addPlayer("Bob", 0, true, 5, 15);<br>Player player = bwm.getPlayerList().get(0);<br>String str = bwm.displayPlayerInfo(0); | "Player name: Bob\n"<br>Player's position: Guest bathroom\n"<br>Items carried: This player has no item.\n" |
| test Invalid Display Player Info | bwm.displayPlayerInfo(-1); | IAE |
| test Display Room Info | bwm.addPlayer("Bob", 0, true, 5, 15);<br>String str = bwm.displayRoomInfo(0); | Room: Guest bathroom\n"<br>Neighbors: 0: Secret room, 1: Guest bedroom\n"<br>Items in room: There is no items in this room.\n"<br>Players in room: Bob\n |
| test Invalid Display Room Info Negative Id | bwm.displayRoomInfo(-1); | IAE |
| test Invalid Display Room Exceed Size | bwm.addPlayer("Bob", 0, true, 5, 15);<br>bwm.displayRoomInfo(2); | IAE |
| Test computer player always attack with item deal most damage | bwm.computerCmd(player id); | “attack”<br>// with item 0 in the list (order based on damage) |
| Test game over when reach max turn | //make game one turn, call isGameOver after one turn<br>bwm.isGameOver(); | true |
| Test game over when target killed | //make target 1 health<br>bwm.poke(1);<br>bwm.isGameOver(); | true |
| Test Random | //Test random with seed<br>Random random = new Random(10); | assertEquals(3, bwm.randomNum(10, random)); |
| Test switch player each round | //turn 1<br>bwm.getCurrentPlayer();<br>//turn2<br>bwm.getCurrentPlayer(); | player a<br>player b |

Player Interface
PlayerCharacter class
Setup:
Player player = new PlayerCharacter((“A”, 0, 1, 5, 15)
);

//attack method in this class is more like a remove item method, it will only cause item removed from player’s item list

| Test case | Input | expected |
|---|---|---|
| Check if item is removed from player’s item list when player attack target successfully | player.attack(item);<br>player.ItemInBag.contains(item); | false // item removed |
| Check if item is removed from player’s item list when player attack target unsuccessfully | add another player into the current room.<br>player.attack(item);<br>player.ItemInBag.contains(item); | false // item removed |
| Check call attack method with an item not in player’s item list | player.attack(item); | IAE |
| Check call attack method with an item not existing | player.attack(item); | IAE |

| | | |
|---|---|---|
| Test getName | player.getName() | A |
| Test getId | player.getId() | 0 |
| Test bag size | player.getBagCapacity() | 5 |
| Test get room id | player.getCurrentRoomId() | 15 |
| test Move | //assume room 1 and 15 are neighbors<br>player.move(1) | 1 |
| test MoveNegativeRoomId | player.move(-1) | IAE |
| test PlayerWithNullName | new PlayerCharacter(null, 0, true, 3, 0) | IAE |
| test PlayerWithNegId | new PlayerCharacter(“A”, -1, true, 3, 0) | IAE |
| test PlayerWithNegBagSize | new PlayerCharacter(“A”, -1, true, -3, 0) | IAE |
| test PickItem | tem item = new Weapon("knife", 4);<br>player.pickItem(item) | assertTrue(player.getItemList().contains(item)) |

| | | |
|---|---|---|
| test PickItemWhenBagIsFull | player.pickItem(new Weapon("knife", 4));<br>player.pickItem(new Weapon("gun", 2));<br>player.pickItem(new Weapon("spoon", 1));<br><br>//pick item when bag is full<br>player.pickItem(new Weapon("torch", 4)); | IAE |
| test GetItemList | player.pickItem(new Weapon("knife", 4));<br>player.pickItem(new Weapon("gun", 2));<br>player.pickItem(new Weapon("spoon", 1)); | List<Item> expected = new ArrayList<>();<br>expected.add(new Weapon("knife", 4));<br>expected.add(new Weapon("gun", 2));<br>expected.add(new Weapon("spoon", 1));<br>assertEquals(expected, player.getItemList()); |
| test Equals | Player temp = pl("Amy", 0, true, 3, 0); | assertTrue(player.equals(temp)); |
| test HashCode | Player temp = pl("Amy", 0, true, 3, 0); | assertEquals(temp.hashCode(), player.hashCode()) |

Item interface
Weapon class

setUp:
knife = new Weapon("knife", 4)

| Test case | Input | expected |
|---|---|---|
| test GetItemName | knife.getName() | knife |
| test GetItemDamage | knife.getPower() | 4 |
| test damage less than 0 | new Weapon("knife", -1) | IAE |
| test Equals | Item knife2 = new Weapon("knife", 4) | assertTrue(knife2.equals(knife)); |

| | | |
|---|---|---|
| test HashCode | Item knife2 = new Weapon("knife", 4); | assertEquals(knife2.<br>hashCode(),<br>knife.hashCode()); |
| test Weapon With Null Name | Item knife2 = new Weapon(null, 4); | IAE |

Space Interface
Room class
SetUp:
World world = new BuildWorld(... 30, 30, …) // represents size of world is 30 x 30.

| Test case | Input | expected |
|---|---|---|
| Test room name. | Room(“room1”, 0, 0, 5, 5) | room1<br>// name is room1. |
| Test room id. | Room(“room1”, 0, 0, 5, 5) | 0<br>// id of the room1. |
| Test getItems in the room. | Room(“room1”, 0, 0, 5, 5) | ArrayList<Item>()<br>//empty list. |
| | Room(“room1”, 0, 0, 5, 5)<br>addItem(new Item(“i1”, 3)) | ArrayList<Item>(Item(“i1”, 3))<br>// one item in this room |
| Test getUpRow() | Room(“room1”, 1, 2, 3, 5) | 1 |
| Test getUpCol() | Room(“room1”, 1, 2, 3, 5) | 2 |
| Test getDownRow() | Room(“room1”, 1, 2, 3, 5) | 3 |
| Test getDownCol() | Room(“room1”, 1, 2, 3, 5) | 5 |
| Test getNeighbors() | Room(“room1”, 1, 2, 3, 5)<br>// assume this room has neighbors are “roomA(id:1), roomB(id:2), roomC(id:3)”. | ArrayList<Integer>(Arrays.as List(1, 2, 3)) |
| Test for valid coordinates | Room(“room1”, -1, -1, 3, 5) | IAE<br>// rows and columns should be greater than or equal to 0 and not exceed |
| | Room(“room1”, 3, 3, 1, 5) | IAE<br>// Up rows and up columns |

| | | |
|---|---|---|
| | | should be less than down rows and columns. |
| | Room(“room1”, 3, 3, 5, 1) | IAE<br>// Up rows and up columns should be less than down rows and columns. |
| | Room(“room1”, 3, 3, 31, 8) | IAE<br>// Down rows and down columns should be less than World size. |
| | Room(“room1”, 3, 3, 5, 31) | IAE<br>// Down rows and down columns should be less than World size. |
| Test add item into room | Item item = new Weapon("Knife", 4);<br>bedroom.addItem(item); | assertTrue(bedroom.getItems().contains(item)) |
| Test remove item from room | bedroom.addItem(item);<br>bedroom.removeItem(item); | assertFalse(bedroom.getItems().contains(item)); |
| Test get item list | Item item1 = new Weapon("Knife", 4);<br>Item item2 = new Weapon("gun", 4);<br>bedroom.addItem(item1);<br>bedroom.addItem(item2); | List<Item> expected = new ArrayList<>();<br>expected.add(item1);<br>expected.add(item2);<br>assertEquals(expected, bedroom.getItems()); |
| Test add player | Player player = new PlayerCharacter("A", 0, false, 1, 0);<br>bedroom.addPlayer(player); | assertTrue(bedroom.getPlayers().contains(player)) |
| Test remove player | Player player = new PlayerCharacter("A", 0, false, 1, 0);<br>bedroom.addPlayer(player);<br>bedroom.removePlayer(player); | assertFalse(bedroom.getPlayers().contains(player)) |
| Test get player list | Player player1 = new PlayerCharacter("A", 0, false, 1, 0);<br>Player player2 = new PlayerCharacter("B", 1, true, 1, | List<Player> expected = new ArrayList<>();<br>expected.add(player1);<br>expected.add(player2);<br>assertEquals(expected, |

| | | |
|---|---|---|
| | 0);<br>bedroom.addPlayer(player1);<br>bedroom.addPlayer(player2); | bedroom.getPlayers()); |
| Test equals. | Space r1 = Room(“room1” 5, 5, 10, 12)<br>Space r2 = Room(“room2” 10, 10, 15, 17) | FALSE |
| | Space r1 = Room(“room1” 5, 5, 10, 12)<br>Space r2 = Room(“room1” 5, 5, 10, 12) | TRUE |
| Test neighboring name is in the neighbor list | Space bathroom = new Room("Bathroom", 2, 0, 0, 5, 8);<br>bedroom.isNeighbor(bathroom); | assertTrue(bedroom.getNeighbors().contains(bathroom)) |
| Test get neighbors list | Space bathroom = new Room("Bathroom", 2, 0, 0, 5, 8); | List<Space> expected = new ArrayList<>();<br>expected.add(bathroom);<br>bedroom.isNeighbor(bathroom);<br>assertEquals(expected, bedroom.getNeighbors()); |
| Test toString | Space bathroom = new Room("Bathroom", 2, 0, 0, 5, 8); | "Room's name: Bathroom\n"<br>+ "Neighbors: \n"<br>+ "Items in room: There is no items in this room.\n"<br>+ "Players in room: There is no players in this room.\n" |