Project 5 Final Design Doc

Anthony Vandergriff

# Section 1: Problem

We want to be able to play a simplified text game similar to Zork.

# Section 2: Inputs, Outputs, and Processing

Inputs: (player input)

* Go east
* Go west
* Yes
* No
* Give up

Outputs:

* Introduction
* Map
* Player health
* Monster and/or weapon
* Choice to pick up weapon
* Game exit message (winning/losing)

Processing:

* Calculating chances of weapon and monsters being placed
* Player/monster health during battle
* Player damage when player has a weapon

# Section 3: Classes and Their Responsibilities

**CLASS:** Weapon and subclasses (Sword, Stick, Flail, Gorilla, Mace)

**RESPONSIBILITIES:** adding to player damage upon pickup

**CLASS:** Participant and subclasses (Player, Monster)

**RESONSIBILITIES:** creates framework for interactable characters

**CLASS:** Monster and subclasses (Goblin, Giant Mosquito, Minotaur)

**RESPONSIBILITIES:** uses extension of participant class to allow for creation of individual monster types with their own stats

**CLASS:** Dungeon

**RESPONSIBLITIES:**

* Creates an array of Rooms with a random length of 5-10 to create the dungeon
* Getting and setting dungeon rooms
* Setting a Player object to index 0 in the dungeon array
* Creates toString of all rooms in dungeon
* Returning monster at player room number

**CLASS:** Game

**RESPONSIBILITIES:**

* Initializes a new Player and Dungeon
* Getting and setting dungeon field and player
* Moving player forwards and backwards through the array
* Calculating hit and miss percentages with respective outputs
* Facilitating player/monster combat

**CLASS:** Room

**RESPONSIBILITIES:**

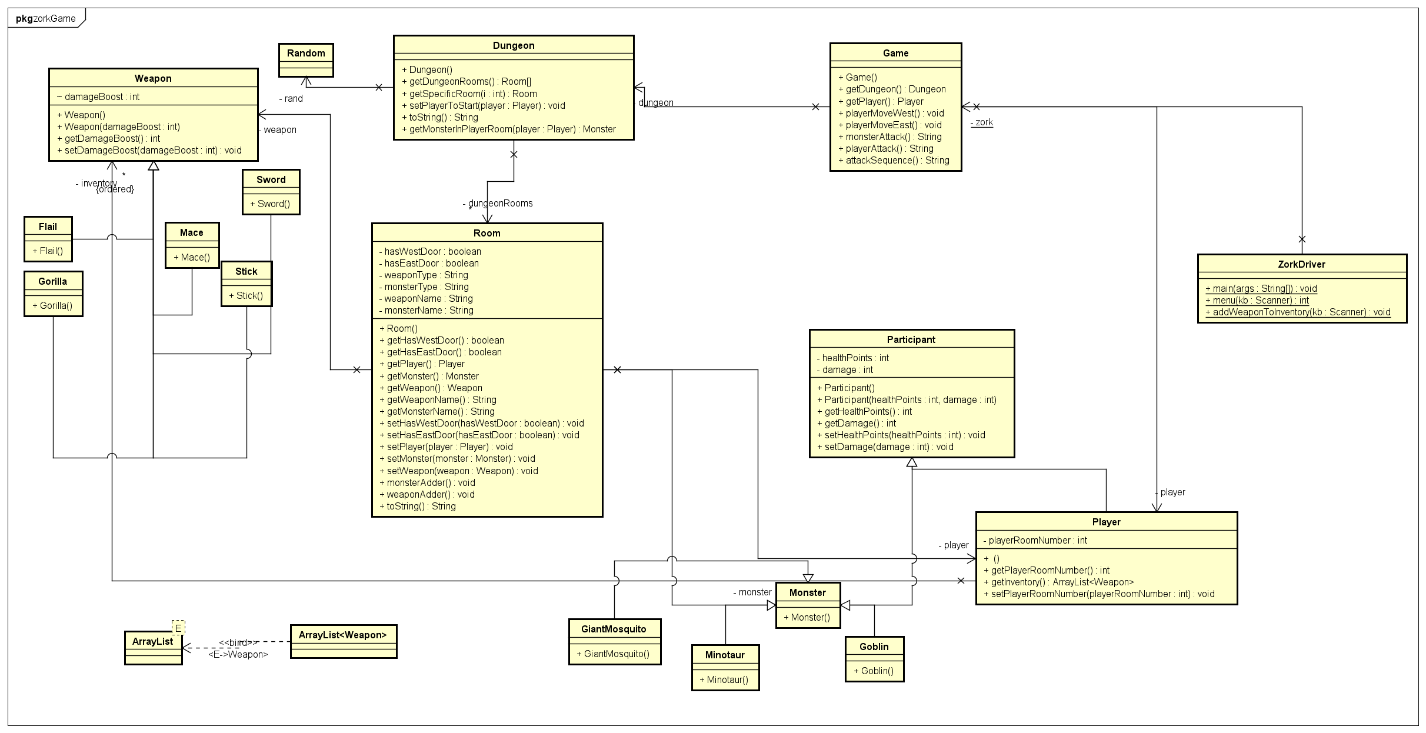
* creating a Room and setting field values
* getting and setting Room field values (hasEastDoor, hasWestDoor, Monster, Weapon, player, weaponName, monsterName)
* Calculating chance of adding types of monsters to that rooms monster field
* Calculating chance of adding types of weapons to that rooms weapon field
* Formatting Room object for display

**CLASS:** ZorkDriver

**RESPONSIBILITIES:**

* Running program
* Displaying menu
* input from user
* Prompting player to add a weapon to inventory

# Section 4: UML Diagram



# Section 5: Algorithms

playerMoveWest():

moves Player backwards in the array by setting the player field in their current room to null, and then setting the player field for the room “behind” them to the Player object

playerMoveEast():

moves Player forwards in the array by setting the player field in their current room to null, and then setting the player field for the room “in front” of them to the Player object

monsterAttack():

calculates probability of monster missing the player using a random number generator, if the monster hits, monster subtracts points from player’s health

playerAttack():

calculates probability of player missing the player using a random number generator, if the monster hits, player subtracts points from monster’s health

attackSequence():

calls playerAttack() and monsterAttack() until player or monster is at 0 HP

# Section 6: Test Cases

|  |
| --- |
| Test Input Values |

|  |  |
| --- | --- |
| Player Input | Result |
| Go east | |P\_\_\_||\_\_\_\_||\_\_\_\_||\_\_\_\_||\_\_\_\_|🡪|\_\_\_\_||P\_\_\_||\_\_\_\_||\_\_\_\_||\_\_\_\_| |
| Go west | |\_\_\_\_||P\_\_\_||\_\_\_\_||\_\_\_\_||\_\_\_\_|🡪|P\_\_\_||\_\_\_\_||\_\_\_\_||\_\_\_\_||\_\_\_\_| |
| Yes (on being prompted to pick up sword) | Picked up sword!  Player Hit Points left: 100  Please Enter Your Choice Below. You May Go East, Go West, Or Give Up. |
| No (on being prompted to pick up sword) | You left the sword on the ground  Player Hit Points left: 100  Please Enter Your Choice Below. You May Go East, Go West, Or Give Up. |
| Give up | Goodbye, and thanks for playing. |
| Go west (with no room to the west) | Sorry, I cant go that way  No monsters in here…  Player Hit Points left: 100 |
| Hello | I’m sorry, I don’t know what you mean |