Justin Dubin
Chase Foster

Problem: CSCI 1260-001 Project 5 - Zork

List of Inputs, Outputs, and Processing Required:

Inputs:

- Go east
- Go west
- go east
- go west
- GO EAST
- GO WEST
- East
- West
- east
- west
- EAST
- WEST

Outputs:

- Welcome message
- Game overview
- Dungeon layout
- Player's current health
- Direction prompt (east or west)
- Warning if monster is present
- Warning if weapon is present
- Weapon pickup message
- Fight (player vs monster)
- Damage done to monster
- Damage done to player
- Monster death message
- Player death message
- Win message
- Lose message

Processing:

- Amount of damage player does
- Amount of extra damage weapon does once picked up
- Random generation of cells in dungeon(5 through 10)
- Random chance of monster spawning (50%)
- Random chance of weapon spawning (1 in dungeon, sword or stick)
- Random chance of player missing monster (10%)
- Random chance of monster missing player (20%)

Justin Dubin
Chase Foster

Identification of Classes and their Responsibilities:

Class Name: Driver

Responsibilities:

- Displaying Welcome Message
- Displaying Game Overview
- Displaying Dungeon Layout
- Allows player input of direction
- Updating Player on actions in game
- Keeps fight going until Player or Monster dies
- Displaying Win/Lose message at end of game

Class Name: Weapon

Responsibilities:

- Assign base value for weapon damage

Class Name: Stick(Weapon)

Responsibilities:

- Change base value for weapon damage to 1

Class Name: Sword(Weapon)

Responsibilities:

- Change base value for weapon damage to 3

Class Name: Participant

Responsibilities:

- Assign base value for HealthPoints to 0
- Assign base value for DamageDone to 0

Class Name: Player(Participant)

Responsibilities:

- Change base value for HealthPoints to 100
- Change base value for DamageDone to 5
- Allows monster to do damage to player
- Changes damage if player picks up weapon

Class Name: Monster(Participant)

Responsibilities:

- Change base value for HealthPoints to 20
- Change base value for DamageDone to 4
- Allows player to do damage to monster

<u>Class Name: Cell</u>

<u>Responsibilities:</u>

- Creates empty list of weapons, players, and monsters in each cell
- Spawns Monster Randomly in each cell (50%)
- Can create a cell with just a weapon and a player (no monster)
- Creates new cell overriding the start cell once the player leaves
- Can create a cell with just a weapon and a monster (no player)
- Allows Monster and Player to be displayed in dungeon

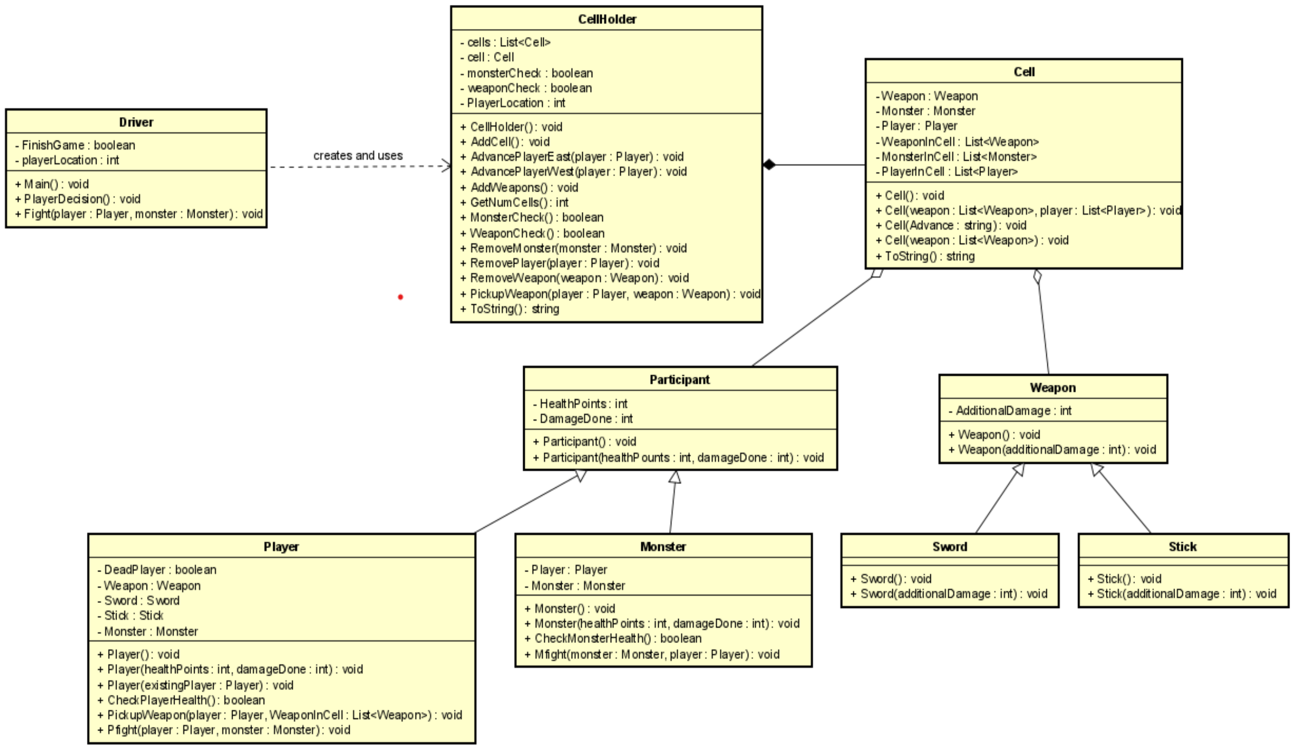
<u>Class Name: CellHolder</u>

<u>Responsibilities:</u>

- Randomly generates Certain Number of Cells (5-10)
- Spawn weapon randomly in a single cell (50% Sword/50%Shield)
- Ensures player spawns in first cell/Ensures monster cannot
- Allows Player to Move east or west
- Checks if player and monster are in same cell
- Checks if player and weapon are in same cell
- Can remove monster from cell if monster dies
- Can remove player from cell if player dies or moves to different cell
- Can remove weapon if picked up by player
- Increases players' damage if weapon is picked up
- Allows entire dungeon to be displayed

UML Class Diagram:

This diagram may omit common methods that essentially all classes have such as constructors and getters/setters, but they must be present in the code


Driver
- FinishGame : boolean
- playerLocation : int
+ Main() : void
+ PlayerDecision() : void
+ Fight(player : Player, monster : Monster) : void
creates and uses
CellHolder
- cells : List<Cell>
- cell : Cell
- monsterCheck : boolean
- weaponCheck : boolean
- PlayerLocation : int
+ CellHolder() : void
+ AddCell() : void
+ AdvancePlayerEast(player : Player) : void
+ AdvancePlayerWest(player : Player) : void
+ AddWeapons() : void
+ GetNumCells() : int
+ MonsterCheck() : boolean
+ WeaponCheck() : boolean
+ RemoveMonster(monster : Monster) : void
+ RemovePlayer(player : Player) : void
+ RemoveWeapon(weapon : Weapon) : void
+ PickupWeapon(player : Player, weapon : Weapon) : void
+ ToString() : string
Cell
- Weapon : Weapon
- Monster : Monster
- Player : Player
- WeaponInCell : List<Weapon>
- MonsterInCell : List<Monster>
- PlayerInCell : List<Player>
+ Cell() : void
+ Cell(weapon : List<Weapon>, player : List<Player>) : void
+ Cell(Advance : string) : void
+ Cell(weapon : List<Weapon>) : void
+ ToString() : string
Participant
- HealthPoints : int
- DamageDone : int
+ Participant() : void
+ Participant(healthPounts : int, damageDone : int) : void
Weapon
- AdditionalDamage : int
+ Weapon() : void
+ Weapon(additionalDamage : int) : void
Player
- DeadPlayer : boolean
- Weapon : Weapon
- Sword : Sword
- Stick : Stick
- Monster : Monster
+ Player() : void
+ Player(healthPoints : int, damageDone : int) : void
+ Player(existingPlayer : Player) : void
+ CheckPlayerHealth() : boolean
+ PickupWeapon(player : Player, WeaponInCell : List<Weapon>) : void
+ Pfight(player : Player, monster : Monster) : void
Monster
- Player : Player
- Monster : Monster
+ Monster() : void
+ Monster(healthPoints : int, damageDone : int) : void
+ CheckMonsterHealth() : boolean
+ Mfight(monster : Monster, player : Player) : void
Sword
+ Sword() : void
+ Sword(additionalDamage : int) : void
Stick
+ Stick() : void
+ Stick(additionalDamage : int) : void

Justin Dubin
Chase Foster

Algorithms:

Class: Player.cs
Operation: Pfight(Player player, Monster monster)

```
START
        int pHealthPoints = player.HealthPoints;
        Random r = new Random();
        if (r.Next(10000) < 2000)
        {
           Console.WriteLine("The monster missed the player!");
        }
        else
        {
           HealthPoints = pHealthPoints - monster.DamageDone;
           Console.WriteLine("The monster hit the player! Player health: " +
player.HealthPoints);
        }
END
```

Class: Monster.cs
Operation: Mfight(Monster monster, Player player)

```
START
        Player Player = player;
        int mHealthPoints = monster.HealthPoints;
        Random r = new Random();
        if (r.Next(10000) < 1000)
        {
           Console.WriteLine("The player missed!");
        }
        else
        {
           HealthPoints = mHealthPoints - Player.DamageDone;
           Console.WriteLine("The player hit the monster! Monster Health: " +
monster.HealthPoints);
           if(HealthPoints <= 0)
           {
              Console.WriteLine("The monster is dead!");
           }
        }
END
```

Justin Dubin
Chase Foster

Test Cases:

Table 1: Direction

Test Input Value:	**Output:**
Value before user input:	|_P_| |___| |___|
Go east/east:	|___| |_P_| |___|
Go west/west:	|_P_| |___| |___|
Go west/west	“Sorry I can’t go in that direction. . .”
Any input other than east/west:	“Sorry, I don’t know what you mean. . .”
Go east/east:	|___| |_P_| |___|
Go east/east:	|___| |___| |_P_|
	You have beaten the dungeon and all of its monsters - congratulations!

Table 2: Fight

Test Input Value:	**Output:**
Value before user input:	|_P_| |_M_| |___|
Go east/east:	A monster is here:
	A Fight has started!
	The monster hit the player! Player health: 96
	The player hit the monster! Monster Health: 15
	The monster hit the player! Player health: 92
	The player hit the monster! Monster Health: 10
	The monster hit the player! Player health: 88
	The player hit the monster! Monster Health: 5
	The monster missed the player!
	The player hit the monster! Monster Health: 0
	The monster is dead!
	|___| |_P_| |___|

Justin Dubin
Chase Foster

Table 3: Weapon (stick) Pickup

Test Input Value:	**Output:**
Value before user input:	|___| |_P_| |_St_| |___|
Go east/east:	You Picked Up a STICK! +1 Damage!
	|___| |___| |_P_| |___|

Table 4: Weapon (sword) Pickup

Test Input Value:	**Output:**
Value before user input:	|___| |_P_| |_Sw_| |___|
Go east/east:	You Picked Up a SWORD! +3 Damage!
	|___| |___| |_P_| |___|

Table 5: Player Death (Player starts this test with low amount of health)

Test Input Value:	**Output:**
Value before user input:	|_P_| |_M_| |___|
Go east/east:	A monster is here:
	A Fight has started!
	The monster hit the player! Player health: 8
	The player hit the monster! Monster Health: 15
	The monster hit the player! Player health: 4
	The player hit the monster! Monster Health: 10
	The monster missed the player!
	The player hit the monster! Monster Health: 5
	The monster hit the player! Player Health: 0
	The player is dead! The game is over

Table 6: Fight with Stick

Test Input Value:	**Output:**
Value before user input:	|_P_| |_M_| |___|
Go east/east:	A monster is here:
	A Fight has started!
	The monster hit the player! Player health: 96
	The player hit the monster! Monster Health: 14
	The monster hit the player! Player health: 92
	The player hit the monster! Monster Health: 8
	The monster hit the player! Player health: 88
	The player hit the monster! Monster Health: 2
	The monster missed the player!
	The player hit the monster! Monster Health: -4
	The monster is dead!
	|___| |_P_| |___|

Table 7: Fight with Sword

Test Input Value:	Output:
Value before user input:	|_P_| |_M_| |___|
Go east/east:	A monster is here:
	A Fight has started!
	The monster hit the player! Player health: 96
	The player hit the monster! Monster Health: 12
	The monster hit the player! Player health: 92
	The player hit the monster! Monster Health: 4
	The monster hit the player! Player health: 88
	The player hit the monster! Monster Health: -4
	The monster is dead!
	|___| |_P_| |___|