Ryan Shupe
CSCI 1260-002
23/04/18

Driver Class

Driver class is going to create a text-based or menu-driven similar to the old Zork (http://en.wikipedia.org/wiki/Zork) games.
The game is played in a dungeon that has between 5 and 10 cells or rooms.
The objective of the game is to exit the dungeon while still alive.

+ main(String[] : args) : void

PROCESSING:

Display welcome message
Ask for the name of the player
Display the games toString and the player name and health
Ask which direction the would wish to go
Call game.move and pass the user input
Check the player position to see if they have beat the dungeon
If a Victory Exception is thrown:
Print a victory message and end the program
If a MonsterException is thrown:
Display message and call the game.fight method
Look for an Item in the same room
If ItemException is thrown:
Display message and call the pickup weapon method from game.
If a DeadPlayerException is thrown:
Display message and end the program.
If ItemException is thrown:
Display message and call the pickup weapon method from game.
If a Health Exception is thrown
Add 10 units to the player's health
Disaply the updated health

+ Welcome() : String

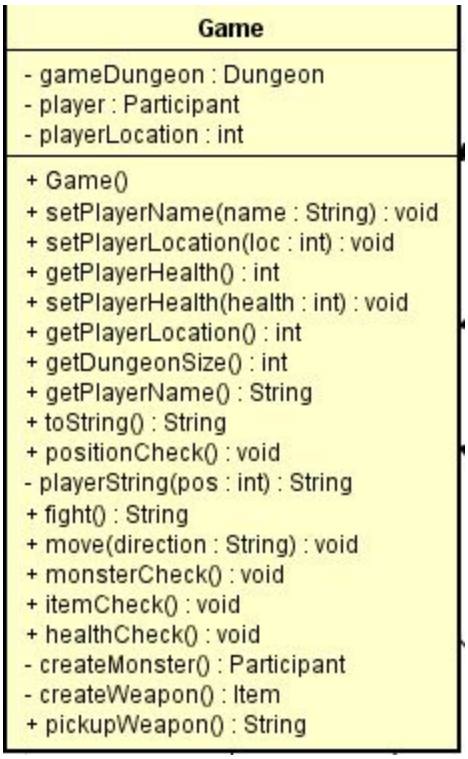
PROCESSING:

Create a String containing a welcome message and details about the game.

Return the string.

Game Class

The game class is going to be how the driver controls the game. Basically this is the brain of the whole thing.



+ Game()

PROCESSING:

Set the player location to 0

+ setPlayerName(name : String) : void

PROCESSING:

Set the name to the string passed in

+ setPlayerLocation(loc : int) : void

PROCESSING:

Set the location to the int passed in

+ getPlayerHealth() : int
    PROCESSING:
        Return the player health

+ setPlayerHealth(health : int) : void
    PROCESSING:
        Set the player health to the int passed in

+ getPlayerLocation() : int
    PROCESSING:
        Return the player location int

+ getDungeonSize() : int
    PROCESSING:
        Return the dungeon size int from the dungeon class
        using the dungeon object created.

+ getPlayerName() : String
    PROCESSING:
        Return the player name String

+ toString() : String
    PROCESSING:
        If the player location is less than the dungeon size
            Make output = call the playerString method and pass the player location.

+ positionCheck() : void
    PROCESSING:
        Check to see if the player location is greater than the size of the dungeon.
        If it is then throw a new VictoryException

- playerString(pos : int) : String
    PROCESSING:
        Take the dungeon String and format it to contain the Player “P”
        The P moves based on player location

+ fight() : String
    PROCESSING:
        Create a boolean called dead
        If monster lands an attack

Add to string that the player was hit then subtract the damage taken from the players health

If the player lands an attack

Add to string that the monster was hit then subtract the damage taken from the monsters health

Loop through this until someones health is <= 0

Set dead to true to get out of loop

If playerhealth is <= 0 throw new DeadPlayerException

If monsters health is <= 0 add to string monster has been killed.

+ move(direction : String) : void

PROCESSING:

Set the string passed in to lowercase

If the player tries to go west and the position is 0 throw Exception

If the player tries to go east increase player location, check for monsters, items, and health pots.

+ monsterCheck() : void

PROCESSING:

Check to see if the dugeon string has a P and M in the same cell

If it does then throw new MonsterException

+ itemCheck() : void

PROCESSING:

Check to see if the dugeon string has a P and I in the same cell

If it does then throw new ItemException

+ healthCheck() : void

PROCESSING:

Check to see if the dugeon string has a P and H in the same cell

If it does then throw new HealthException

- createMonster() : Participant

PROCESSING:

Create a random number

If the number is < 8

Create a Dragon

If the number is < 30

Create a Ogre

If the number is < 50

Create a Cyclops

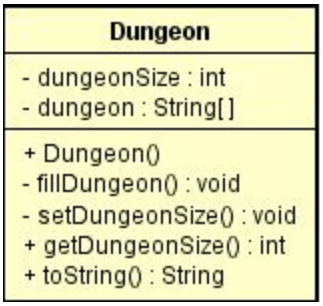
Return the monster

- createWeapon() : Item
    PROCESSING:
        Create a random number
            If the number is == 0
                Create a Stick
            If the number is == 1
                Create a Stone
            If the number is == 2
                Create a Sword
            Else
                Create a dagger
        Return the Item

+ pickupWeapon() : String
    PROCESSING:
        Call create Weapon
        Set damage to the playerdamage + the item damage
            Create a string containing the name and the updated player damage
        Return the string

Dungeon Class

Dungeon Class creates a random size dungeon from 5-10 rooms and puts items, monsters ets in them at random



+ Dungeon()
    PROCESSING:
        Call setDungeonSize
        Create an array of Strings that is the size of the dugeon size
        Call fill dungeon

- fillDungeon() : void
    PROCESSING:
        Fill the array with Strings that reseble a cell
        Have one of the Strings contain an I for item

50% chance the string will contain a monster M
25% chance the string will contain a health pot H

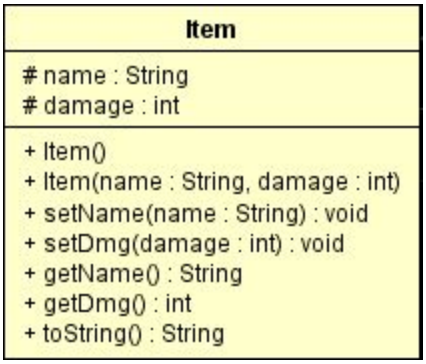
- setDungeonSize() : void
    PROCESSING:
        Set the dungeon size variable to a number between 5 and 10

+ getDungeonSize() : int
    PROCESSING:
        Return the dungeon size

+ toString() : String
    PROCESSING:
        Create a string and fill it by going through the array of strings

Item Class

To create type item that has damage and a name that can be used in the game class.



+ Item()
    PROCESSING:
        setName to Unknown
        setDmg to 0

+ Item(name : String, damage : int)
    PROCESSING:
        setName to the string passed in
        setDmg to the int passed in
+ setName(name : String) : void
    PROCESSING:
        Set the name to the String passed in

+ setDmg(damage : int) : void

PROCESSING:

Set the damage to the int passed in

+ getName() : String

PROCESSING:

Return the string Name

+ getDmg() : int

PROCESSING:

Return the int damage

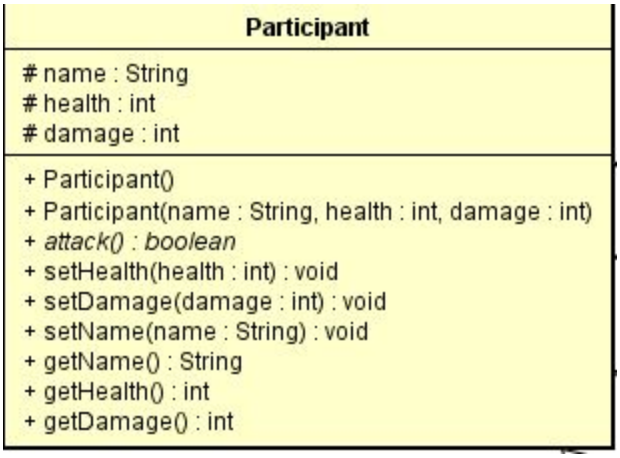
+ toString() : String

PROCESSING:

Return a String that has the name and the damage neatly formatted

Participant Abstract Class

To create an abstract class that makes up all participants in the game including monsters and the player



+ Participant()

PROCESSING:

setName to a default name

+ Participant(name : String, health : int, damage : int)

PROCESSING:

Call SetName to the string passed in

Call setHealth to the int passed in

Call set Damage to the int passed in

+ *attack() : boolean*

+ setHealth(health : int) : void
    PROCESSING:
        Set health to the int passed in

+ setDamage(damage : int) : void
    PROCESSING:
        Set damage to the int passed in

+ setName(name : String) : void
    PROCESSING:
        Set Name to the string passed in

+ getName() : String
    PROCESSING:
        Return the name
+ getHealth() : int
    PROCESSING:
        Return health

+ getDamage() : int
    PROCESSING:
        Return damage

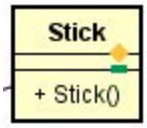
Dagger Class

To create a dagger weapon

+ Dagger()
    PROCESSING:
        Call the super constructor to create a participant and pass it 2 variables. The name Dagger and the Damage it adds 6.

Stick Class

To create a stick weapon

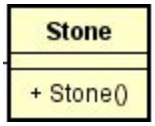


+ Stick()
    PROCESSING:

Call the super constructor to create a participant and pass it 2 variables. The name Stick and the Damage it adds 1.

Stone Class

To create a stone weapon



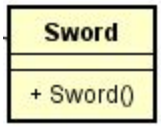
+ Stone()

PROCESSING:

Call the super constructor to create a participant and pass it 2 variables. The name Stone and the Damage it adds 3.

Sword Class

To create a sword weapon



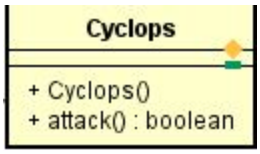
+Sword()

PROCESSING:

Call the super constructor to create a participant and pass it 2 variables. The name Sword and the Damage it adds 10.

Cyclops Class

To create a type of monster cyclops to add variety



+ Cyclops()

PROCESSING:

Call the super constructor to create a participant and pass it 3 variables

The name Cyclops, health 20, and damage 8

+ attack() : boolean

PROCESSING:

Create a new random number variable 1-10 & boolean hit

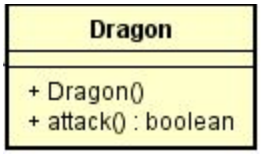
If the number is less than or equal to 2 set the boolean hit to false

(has a 20% chance of missing)

Return hit

Dragon Class

To create a type of monster Dragon to add variety



+ Dragon()

PROCESSING:

Call the super constructor to create a participant and pass it 3 variables

The name Dragon, health 50, and damage 15

+ attack() : boolean

PROCESSING:

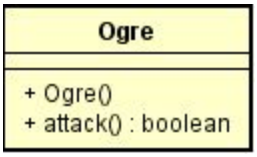
Create a new random number variable 1-10 & boolean hit

If the number is less than or equal to 3 set the boolean hit to false (has a 30% chance of missing)

Return hit

Ogre Class

To create a type of monster Ogre to add variety



+ Ogre()

PROCESSING:

Call the super constructor to create a participant and pass it 3 variables

The name Ogre, health 20, and damage 8

+ attack() : boolean

PROCESSING:

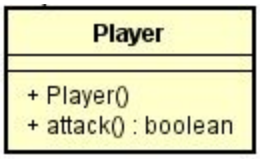
Create a new random number variable 1-10 & boolean hit

If the number is less than or equal to 2 set the boolean hit to false (has a 20% chance of missing)

Return hit

Player Class

Create a player type because that's necessary



+ Player()

PROCESSING:

Call the super constructor to create a participant
Set the health to 100 because it is a player
Set has Weapon to false
Set default damage to 5

+ attack() : boolean

PROCESSING:

Create a new random number variable & boolean hit
If the number is equal to 1 set the boolean hit to false
(has a 10% chance of missing)
Return hit

HealthException Class

Create own HealthException so it can be thrown/caught when the player finds a healing pot

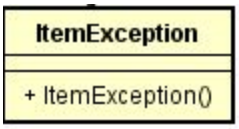
+ HealthException()

PROCESSING:

Call the super constructor from Exception class and pass a String message.

ItemException Class

Create own ItemException so it can be thrown/caught when the player finds an item



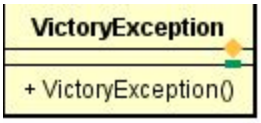
+ ItemException()

PROCESSING:

Call the super constructor from Exception class and pass a String message.

VictoryException Class

Create own VictoryException so it can be thrown/caught when the player wins the game



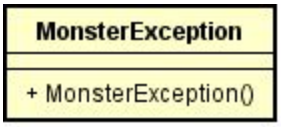
+ VictoryException()

PROCESSING:

Call the super constructor from Exception class and pass a String message.

MonsterException Class

Create own MonsterException so it can be thrown/caught when the player finds a monster



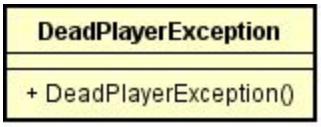
+ MonsterException()

PROCESSING:

Call the super constructor from Exception class and pass a String message.

DeadPlayerException Class

Create own DeadPlayerException so it can be thrown/caught when the player dies



+ DeadPlayerException()

PROCESSING:

Call the super constructor from Exception class and pass a String message.