Going Down Object Design Document

By: Brendon Serrano, Charneaka Lewis, Jeffrey Graves

# Introduction

## Object design trade-offs

This document provides an overview of the object design of the Going Down game. Included is a breakdown of each subsystem and the classes that they contain.

### We will be using sqlite to offer a self-contained, embedded storage solution.

### Game objects will be created as needed improve performance.

### Variables and objects will have full names only abbreviating general concepts like num for number.

## Interface documentation guidelines

### Classes, methods, and variables MUST be named as singular nouns with descriptive verbs as needed to be able to fully understand the functionality by reading the name.

### Room exits must contain the ID to the connected rooms

# Packages (Descriptions and interactions)

## View

### Going Down Client

The view will contain the logic for the GUI and console that will handle all interactions displaying to the user and accepting user input.

## Controller

### Character subsystem

### The character subsystem contains the Boss, Character, Monster, Player, and Score classes. These classes contain the logic to define, set, and track the values and attributes of the different characters. It also updates the values of the score as the characters interact with each other throughout the game.

### Environment subsystem

The environment subsystem contains the Monastery, Puzzle, and Room classes. These classes contain the logic to set and retrieve the attributes that make up the specifics of the game’s environment.

### Inventory subsystem

The inventory subsystem contains the Armor, Inventory, Item, Potion, and Weapon classes. This subsystem contains the logic to set and provide the attributes that make up tools to be used to defeat the game.

### 

### Game subsystem

The game subsystem contains the Game class which holds the logic that controls the main commands of the game.

## Model

### Going Down Storage

* + 1. Storage contains the databases that store items, monsters, rooms, and puzzles that will be called throughout the game.

# Class interfaces (Descriptions and interactions)

### GoingDownGame

This class gets an instance of the game object and uses the Observer method to keep up with the changes in the game.

### Inventory

This class contains the methods and attributes that allow the player to store their items such as weapons or armor in their inventory. The player will be limited by a maximum weight limit. Each player will be able to carry up to but no more than said weight limit.

### Item

The item class is a parent class to armor, weapon, potions. It will provide all attributes that are shared between child classes. Item class will also itemize all child classes in a numeric list.

### Armor

The armor class is a subclass of the item class and it controls the armor’s damage reduction. When a player uses armor, it will provide a reduction in the damage that the player can obtain.

### Weapon

The weapon class is a subclass of the items class and contains the attributes and methods which control the characteristics and values of each weapon.

### Potion

The potion class is a subclass of the item class and contains the attributes and methods that control the characteristics and values of each potion item. The player can use potions to help heal the damage sustained while fighting monsters in the game.

### Character

The character class contains the attributes and methods that control all the elements of the character. This includes the name, character attacks, character damage, and heath.

### Monster

The monster class is a subclass of the character class. It provides the attributes and methods that control the monsters, their attack types, and their point values.

### Boss

The boss class is a child of the monster class. It inherits all the attributes and methods of the Monster class but multiplies all values in the monster class by 4. This makes the boss harder to defeat.

### Player

The player class contains the attributes and methods that keep track of the player values such as the player’s inventory limit and score.

### Score

The score class will contain a score based on monsters killed and puzzles solved. When the class is called, it will display the scored acquired by the player.

### Monastery

The monastery class contains the attributes and methods that provide a list of the rooms in the game. The methods are made up of getters and setters that display the uncompleted and completed rooms in the game.

### Puzzle

The puzzle class contains the methods that display the puzzles to be solved in the rooms needed to progress in the game.

### Room

The room class control the aspects of the rooms in the game. It is made of the attributes and methods that provide the elements of the room. It will call the monsters, puzzles, and the adjoining rooms.

### Game

The game class handles the interaction for the player. Additionally, it will save the game and resume the game.

### ItemDB

This class handles all database interactions for Items.

### MonsterDB

This class handles all database access for the Monster.

### RoomDB

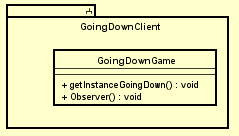
This class handles all of the database interactions for Rooms.

### PuzzleDB

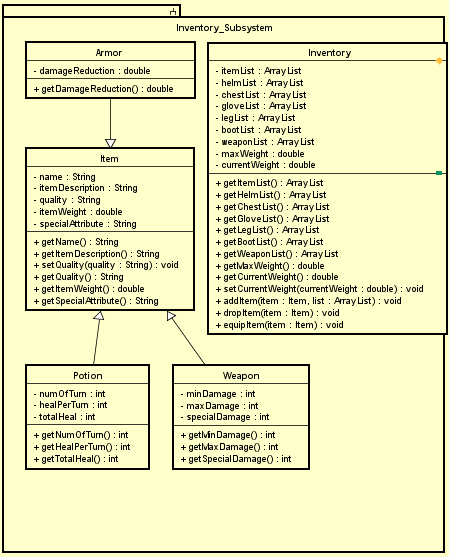
This class handles all the database interactions for the Puzzles class.

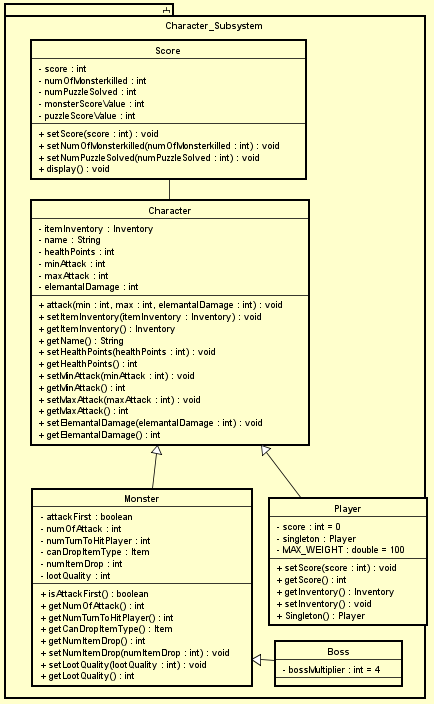
# 4. Object Diagram

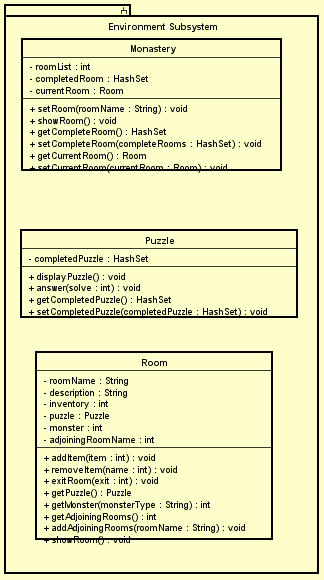
View Subsystem:

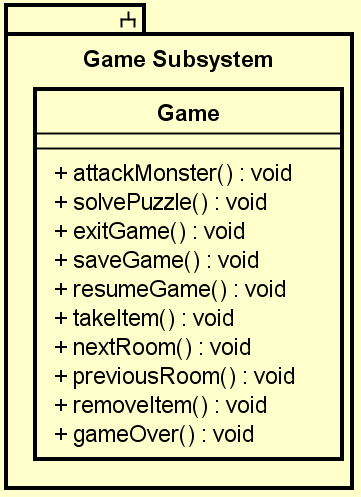


Controller Subsystem:

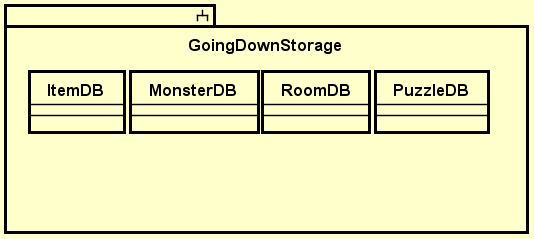








Model Subsystem:



Complete Object Diagram:

