OBJECT-ORIENTED SYSTEMS DEVELOPMENT

Analysis and design

Monk Game

An RPG game featuring a monk that fights monsters to get to the treasure. This game is to be designed and developed using object-oriented development with appropriate design patterns.

Harry Wijnschenk

Table of Contents

Requirements gathering

Functionality

- Main character is a “monk”
- At minimum 3 types of rooms:
    - Monster room (fight monsters)
    - Empty room (heal monk)
    - Treasure room (win game)
- Monk must visit each room type before winning
- Main components:
    - Dungeon Generation
    - Exploration
- The design should allow new characteristics to be easily implemented

Dungeon generation

The dungeon will comprise of original room types, and in addition I will add a Boss room type. The boss room will have a tougher opponent than the monster room, however, on winning the battle, the monk will gain an agility point (max of 5) and restore some health back.

Exploration component

This component allows the player to explore the dungeon.

The game begins with settings the monks name and description. By default, the monk has 15hp and 3ap. The monk can move back and forth between rooms. Each room will either have 1 or 2 paths they can choose to visit, or a dead end. There will always be a path to a treasure room, and each time playing will have one and only one treasure room available after visiting each room type.

Game modes will be easy and hard, easy will have no boss rooms and enemies with less health. Hard will have boss rooms and monsters will have more health.

The interface using text will display the current room, actions that can be performed, monk stats and any relevant information such as the monster’s health. The game will end with entering the treasure room, or the monk dies.

Combat system

The combat is a turn-based system, in which either the monk or monster can choose one of 2 possible actions.

1. Attack: This decreases the others health by the number of attack points that entity has.
2. Defend: The entity increases its health by 1.

Each action has a 50% chance of failure.

Design pattern

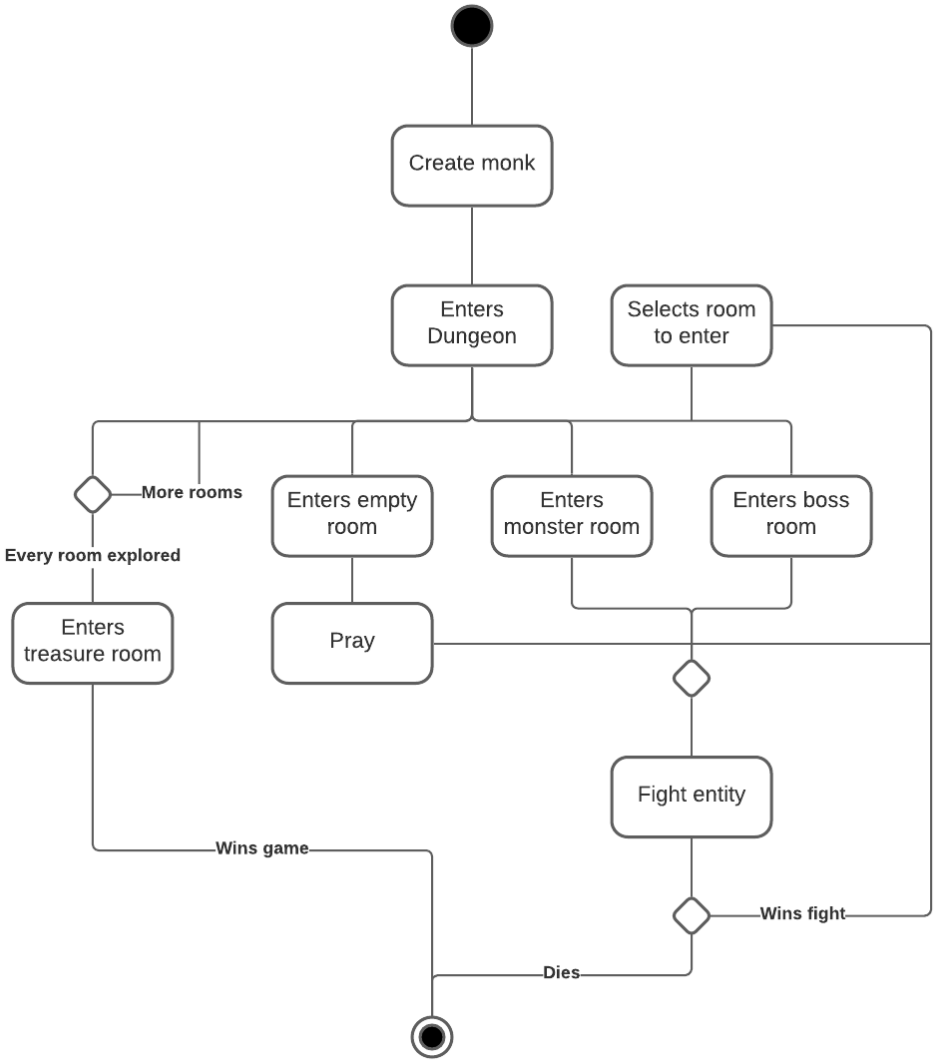
I will be utilizing the creational pattern “Factory Method”. The key points about this design pattern include:

- Defining an objects interface but letting subclasses choose the class to instantiate.
- To define a “virtual” constructor.

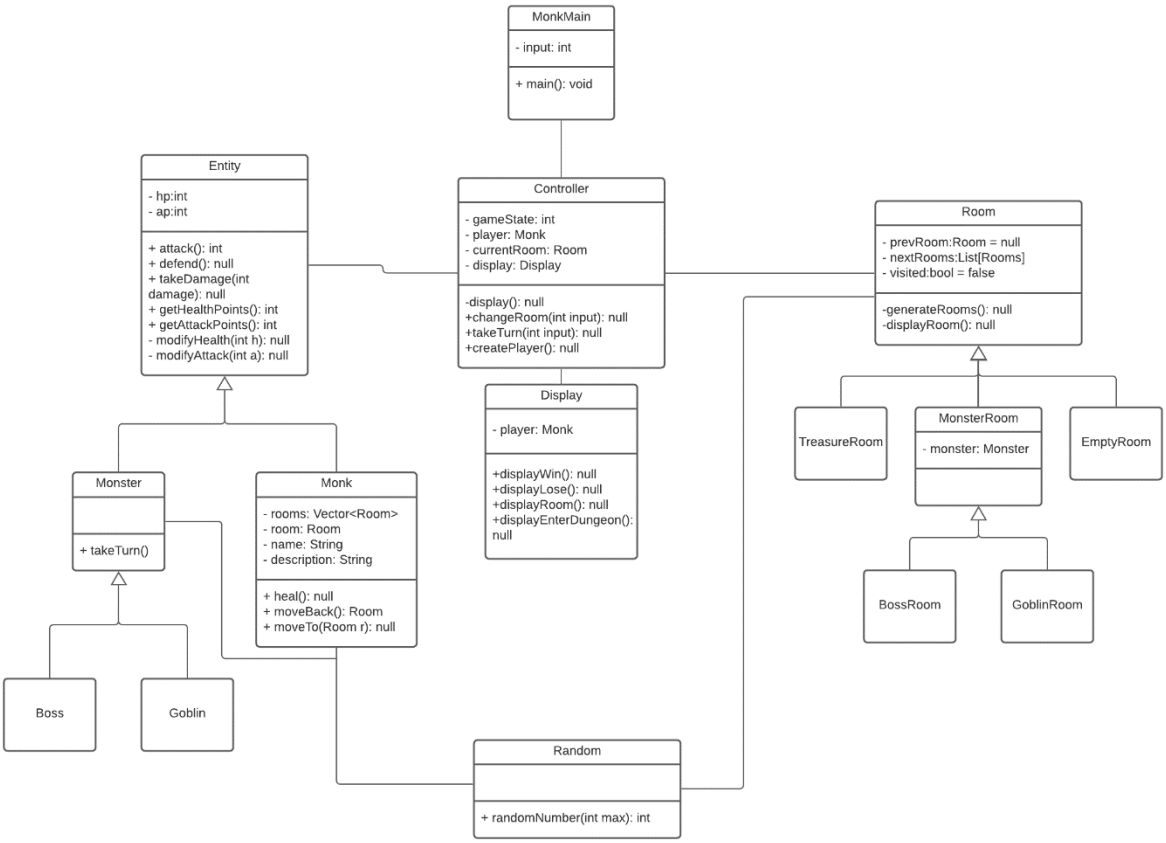
I will be using this pattern to make the code more efficient for implementing new rooms and monsters. I feel this pattern will make the code cleaner and far more maintainable.

Modelling

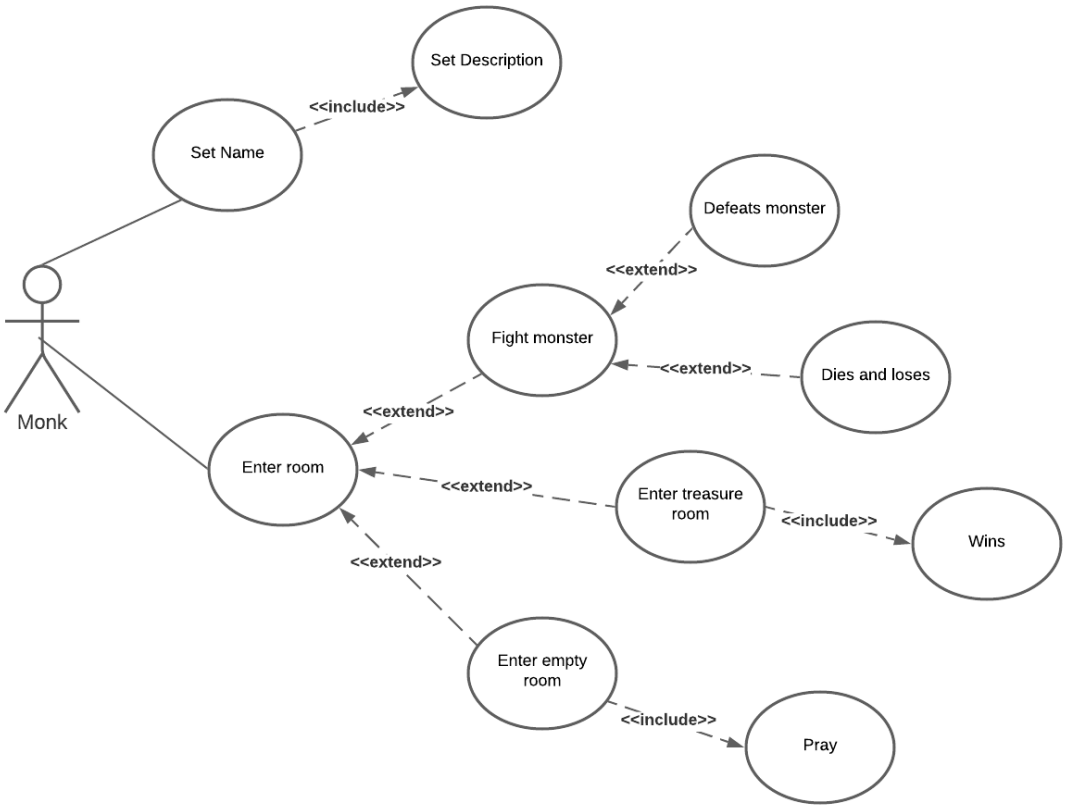
Activity Diagram



Class Diagram



Use case


Set Description
<<include>>
Set Name
Defeats monster
<<extend>>
Fight monster
<<extend>>
Dies and loses
<<extend>>
Monk
Enter room
<<extend>>
Enter treasure room
<<include>>
Wins
<<extend>>
Enter empty room
<<include>>
Pray