Math Warrior

Analysis Model

Math\_Warrior\_Analysis\_Model.doc

Draft 1.0

10/13/2014

[Team 2: Matthew Berger, Quan Tran, Brock Bearchell]

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| Draft 1.0 | Matthew Berger  Quan Tran  Brock Bearchell | This is the first version of the Analysis Model. We are detailing all abstract models of the system defined by the requirements elicitation. | 10/5/2014 |
| Draft 1.1 | Matthew Berger  Quan Tran  Brock Bearchell | We added lots of game logic missing for the model. Furthermore, lots of data for the game was added as well with descriptions showing all functions of the | 10/102014 |
| Draft 1.2 | Matthew Berger  Quan Tran  Brock Bearchell | More diagram descriptions with more examples of game logic including Rooms, Monsters and Puzzles. | 10/13/2014 |

Table of Contents

[Revisions 2](#_Toc400990756)

[Introduction 4](#_Toc400990757)

[Current System 4](#_Toc400990758)

[Proposed System 4](#_Toc400990759)

[Overview 4](#_Toc400990760)

[Functional Requirements 5](#_Toc400990761)

[Non-Functional Requirements 10](#_Toc400990762)

[System Model 11](#_Toc400990763)

[Use Case Model 11](#_Toc400990764)

[Object Model 13](#_Toc400990765)

[Dynamic Model 16](#_Toc400990766)

[Glossary 20](#_Toc400990767)

# Introduction

The purpose of this document is to abstractly model requirements for a text based game software application. Furthermore, the functional model, dynamic model and object model are detailed with explicit descriptions and diagrams to show accurate representations of the system. The intended audience for this document is a development team of software engineers/architects who will be able to build, test, and implement all aspects of this documents design.

Math Warriors is a text based role playing game that allows a player controlled by the user to travel through dungeon rooms battling monsters, solving puzzles, and exploring and finding items until they can escape the dungeon. The player will be able to enter text based commands as functional play that will have cause and effect on the game. This is a standalone java software application that interacts with a database system to read in data for the game. There are no other related systems such as servers or web front end utilities that the game interacts with. This game will be independent of operating systems or network configurations due it being a standalone application.

The setting of this text based game is an abandoned high school. The main character is a math nerd that wakes up in a room inside of the school and has no idea where he is or what is going on. The school is abandoned and infested with traps (puzzles) and zombies of people who attended the school (monsters). The math nerd has a gigantic textbook as a default weapon and has to find the exit in the school in order to escape.

# Current System

There is no current system in use for this software application. The required specifications that have been abstractly modeled for this text based role playing game represent the first version of the system. The system modeled in this document will be the first version of this application.

# Proposed System

## Overview

The rest of this document will go into detail of the System Model of this application. The system model is made up of the functional model depicting functional and non-functional requirements, the object model depicting the class diagrams of important entities in the system, and the dynamic model that simulates how external factors affect the system through sequence and state machine diagrams.

## Functional Requirements

* Commands
  + The game will feature a list of commands that the player can enter in order to play the game. These commands will be types in as text and then handled accordingly after being inputted by the player.

|  |  |
| --- | --- |
| TEXT COMMANDS | COMMAND DESCRIPTIONS |
| move up | Move player up on the game map |
| move down | Move player down on the game map |
| move left | Move player left on the game map |
| move right | Move player right on the game map |
| save game | Save the players game for later. |
| quit game | Will end the game |
| help | Will show the entire command list |
| show stats | Displays the players stats at that moment in the game |
| show inventory | Displays the player’s inventory of items, weapons, and armor |
| use weapon | The player attacks with their weapon against a monster. This command will only work if battling a monster. |
| use item \_\_\_\_\_(item name) | This will allow the player to use an item. |
| equip item \_\_\_\_\_(item name) | This will allow the player to equip an item that is armor or a weapon |
| drop item \_\_\_\_\_(item name) | This will allow the player to drop an item from inventory. |
| edit game | This command will allow a creation system to be implemented. |

* + There will be some commands tied to certain monsters and puzzles that can only be used at that moment in the game and are not permanent commands in the command list. These commands would be described as hint messages to the player as they are tied to the monster or puzzle. For example:

“You enter the room. This room is the great hall of kings where dwarves feast upon mead and gnomish bread. However, there is no light. You do not know of the dangers ahead and cannot continue without light. You notice a torch on the ground, what will you do?” The player could enter “light torch” and then be notified by “The room was lit by the torch. You can now continue on your adventure.” If the player entered an invalid command, they would be notified by “That command is not valid, reference your help menu by typing “help” or read the description for the room again. “

* + If the player enters a command that is not a valid command, they are to be notified that it is not a valid command.
* Game Map Contents
  + The game contains a minimum of 50 rooms for the player to venture through. Each room can contain the possibility of a puzzle, a monster to fight, an item to obtain, or nothing resembling an empty room. There must be a minimum of 10 monsters and 10 puzzles in the entire game map. Puzzles, items, and monsters do not have to be in the same room.
  + Puzzles are mainly descriptions of rooms that you have to solve that contain a riddle. The player will have to type in a text command that solves the puzzle to allow them to earn more points on their score. Puzzles will have descriptions that hint to what the player should type in order for them to be solved. Monsters are different than puzzles. When a player enters a room and a Monster is there, they can choose to initiate combat or move to another room. If the player enters a room and initiates combat with a Monster, then a turn based battle simulation between the player and the monster in a room starts. The player will be notified who is attacking, the results of the attack, and all battle information as it is simulated. When the players turn starts, they can type “use weapon” or “use item potion” to heal their health points. They can also type any other command as well, but these are related to the battle and are used to win the battle. When a puzzle is presented, a description of the puzzle and an acceptable solution to that puzzle must be provided for the player to progress further in the game.
  + Room Examples:

|  |  |
| --- | --- |
| Gymnasium | You enter the school gym where all the sports are practiced. You notice cracked concrete floors and spider webs covering the windows. All of the gym equipment is scattered and creaking groans are heard in the distance. |
| Cafeteria | You enter the cafeteria where shattered plates of fried chicken and decomposing goo decorate the floor, walls, and destroyed tables. You notice the horrid smell coming from the goo. |
| Principles Office | You determinedly enter the office of the school principle. Items are scattered across the room and important papers are lying around torn and burned asunder. You have heard stories of how terrifying this room is. |
| Locker Room | As you enter the locker room, you sense the abandonment of cleanliness as football gear is scattered everywhere, even in the showers. Most of the lockers are empty but anything remaining looks broken and useless. |
| Counseling Office | You enter the counseling office. This room has guidance books and bullying pamphlets neatly stacked in piles. It looks like they were getting ready for a school wide seminar. |

* + Rooms: Science room, janitors closet, locker room, teachers office, principles office, weight room, cafeteria, halls(A-E), Classrooms, Gym, wrestling room, basketball court, football field, teacher break room.(More examples of Rooms)
  + Monster Examples:

|  |  |  |
| --- | --- | --- |
| Cheerleader Captain | The captain of the cheerleading squad. Beware her ferocious pom poms. | Health: 10  Strength: 8 |
| Principle | Principle of the school. The tall and stocky zombie glows at you with green eyes and sharp teeth. Beware those teeth! | Health: 15  Strength: 10 |
| Substitute Teacher | The substitute teacher. Her once golden hair has now turned ghostly white and her skin is a yellowish hue. She is holding an enormous kickball glowing with green goo, look out! | Health: 12  Strength: 7 |

* + More Monster Names: Ghost Bully, Sprinting zombie jock, Sassy cheerleader ghost, Rampaging math tutor, Ghosts of the Math Club (club members 1-5), Head of the math teacher, DEAD LUNCH LADIES, Wise old Janitor, The mad Science teacher, Mystery meat monster! (These are ideas for more monsters in the game)
  + Puzzle Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| There is no light and you cannot see. You cannot pass without knowing the dangers ahead. There is a lantern lying on the ground with matches next too it covered in goo. Maybe the matches can be placed in the lantern? | light matches | Try using “light” with what you are trying to light. | The batteries are placed in the flashlight and it lights up. You can now see the way! The room is now clear to pass through. |
| As you enter the room, your foot triggers a rope trap and are snagged up into the air. You are swinging side to side and cannot swing yourself up. Scissors are laying on the ground below and you think you can reach them with your fingertips. | grab scissors | Try using “grab” with what you are trying to grab from the floor. | You grabbed the scissors and cut the rope. You fall with a steady landing but seem ok. The room is now cleared! |

* + Item Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| Mountain Dew | Use this item to heal your character. | Amount: 15 | ITEM |
| Heavy Chemistry Book | Equip this item to increase your characters strength in battle. | Amount: 3 | WEAPON |
| Trash Can Lid | Equip this item to increase your hp to endure more hits in battle. | Amount: 10 | ARMOR |

* Data Storage – Resuming Games and New Games
  + The player must be able to save their game at any point. The text command “save game” should allow the player to save their data including room status and contents as well as their player’s stats. The game must recognize different users and when they return to the game, allow them to resume where they left off. When the game start is initiated, the player must enter their username in order to obtain the data for that player. If the player has not played before and must create a new user, they must enter their username in order to create their player.
* Creation System
  + An adventure creation system must be provided to allow the user to create and edit the current game information in the database. The information allowed to be changed could be the player adding a monster to the game, a puzzle to the game, a room to the game, or any aspect regarding the game map that doesn’t interfere with the determined game logic and processing. This is up for the development team to decide which option the player can add: monster, puzzle, room, or item.
* Player Functionality – Inventory and Operations
  + Players will be able to solve puzzles, fight monsters, take and inflict damage, and equip /use items including armor, potions, and weapons. This will be important for the combat mechanics of the game Players have operations tied to commands like attack, use item, equip armor, etc. Furthermore, the character will have equipment items that will add bonuses to base stats like strength and health points. These items that the player can equip and use in the game will be held in an inventory system that has a maximum capacity of storage which is 5. The player will be limited on how many items they can equip and carry at one time by only allowing FIVE items in inventory. Items that increase base health are an ARMOR TYPE. Items that improve strength are a WEAPON TYPE. Items that restore health points to the player are an ITEM TYPE. The player at any time could have 5 weapons, 5 armors, 5 items, or a mix of the bunch.
  + The Player will have health points, a name, score, strength and inventory of items. The player will use strength to attack monsters. Each WEAPON ITEM adds to the strength value. The player will lose health points if they are attacked by a monster. To improve health points, equip ARMOR ITEMS. To restore health points, USE ITEMS with the ITEM TYPE.
  + The Player will be able to add items, drop items, equip items, use weapon, and display stats in regards to game mechanics outside of saving, exiting, navigating and so forth.
* Navigation Through Game Map
  + The player will type text commands to navigate the map by choosing to move up, down, left, or right. For Example: “You have defeated the monster. The room is now clear and there is nothing of interest here.” The player enters “move up “.”The player moves north”. If they have already been in this room, they will be notified by “This room looks very familiar and there is nothing of interest here.” If the player decides to go to a room that is out of the bounds of the map, they should be redirected to another room on the other side of the boundary map. So if they move left and that was the leftmost room on that row in the map, then the rightmost room on that same row is where they will end up.
  + The Game Map is set up to resemble a table of rows and columns where each cell contains a Game Room. Each Game Room has its own contents as well as description and so forth.
  + Map Demo Example: M = Monster, P = Puzzle, I = Item, E = Exit, S = Player Start

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * + M | * + S | * + M |  | * + P |  | * + M | * + P | * + P |  |
| * + I |  | * + P |  |  | * + M | * + P |  | * + I | * + M |
| * + P | * + M | * + I |  | * + I | * + P |  | * + P | * + M |  |
|  | * + I | * + P |  | * + M |  | * + I | * + M |  | * + I |
| * + M |  | * + M |  | * + P |  | * + E |  | * + P |  |

* Tracking Player
  + The game should be able to track the player’s score and rooms that have been visited. If a player has solved a puzzle or defeated a planned monster, they should not have to do this again. If a player enters a room they have already solved, they should be notified that the room is safe and empty. The player’s score will be updated based on how many monsters they defeat, how many puzzles they solve, and how many items they obtain in their play through. Each time they achieve one of these three categories, they can be rewarded a score point. At the end of the game, the player needs to be notified of their score.
* Game Win
  + In order to win the game, the player must find the exit room in the map. When they find the exit room, they win the game and are notified of their score based on how many puzzles they solved and how many monsters they defeated. The player must have obtained enough points in order for the exit room to allow winning the game. The exit room will appear as a regular room with no content until enough score points have been obtained by the player.
* Default Weapon
  + The player will be assigned an item that is a weapon type at the start of the game. The default weapon is a gigantic textbook and will be used to help boost attack strength at the beginning of the game.

## Non-Functional Requirements

##### Performance:

The game is meant for only one user to play at a time and they can interact with only one command terminal to type in text commands to play the game. When the user types in a command, the game should run the command and simulate within a second displaying the text notification on screen or to the console. On startup for the application, the user should login or create a new user and start the game within 10-15 seconds maximum. There is not a lot of data and records to be read from the database so processing time should be short.

##### Security:

The user can only make changes in the database with limited commands in the creation system. The user will be limited to the changes they can make protecting the game data from getting wiped or changed dramatically for malicious intent by the user.

To play the game the user must register with a user name and password. If the user does not have a user name and password or they have forgotten their user name and/or password they must create a new user name and password.

USER NAME:

· It must be unique in the database.

· It must be at least 5 characters long.

· It is not case sensitive.

· It must start with a letter but the remaining characters can be any combination of letters and numbers only.

PASSWORD:

· It must be at least 6 characters long.

· It is case sensitive.

· It must begin with a letter and the remaining characters may be any combination of letters and numbers only.

· It does not need to be unique to the database.

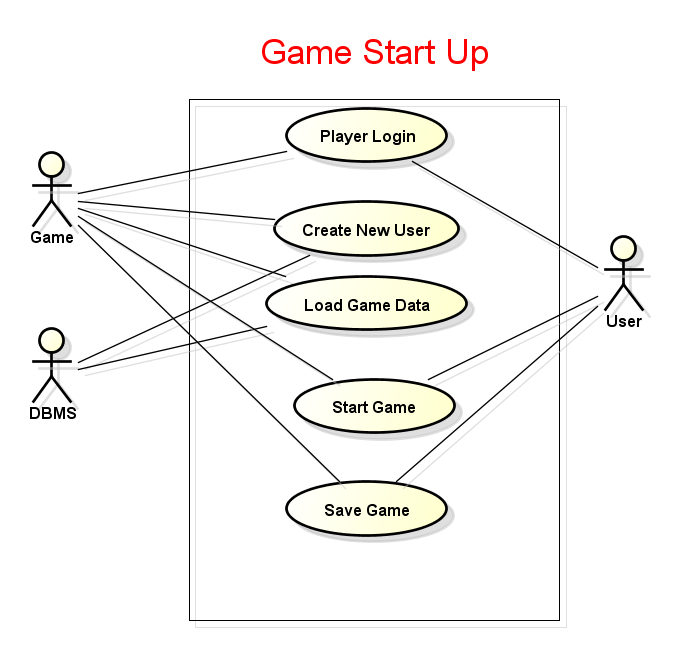
##### Maintainability

The game will be maintained easily through the DBMS system implemented in the game. All changes to the game for data regarding the monsters, rooms, map, items, etc. can be made in the DBMS system. That way the game functionality is independent and doesn’t have data hard coded into source code. All source code must be well documented and organized following coding standards.

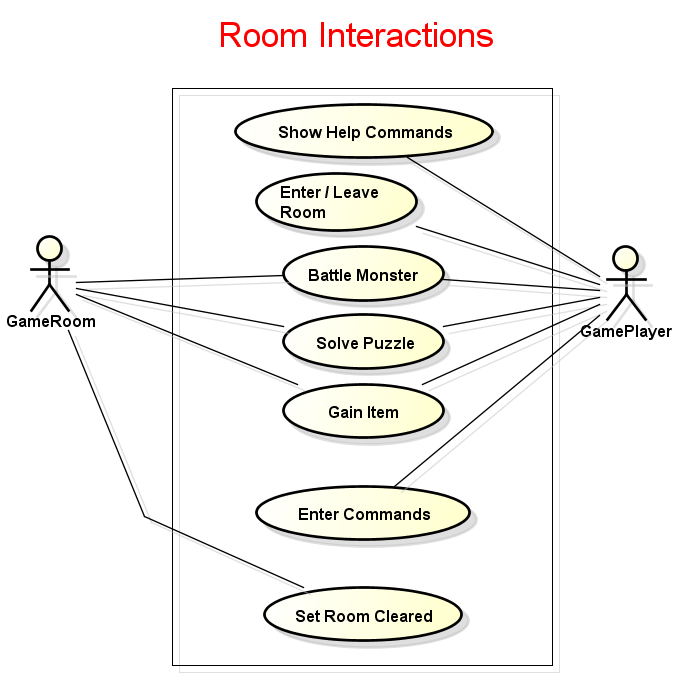
## System Model

### Use Case Model

The first use case model shows interactions with the game start up between the actors of the player, game, and database. This models the login process as well as saving/loading game data and starting the game.



This use case diagram models the different operations a player can perform in a room as well as how the player interacts with rooms in the game map. This shows how the player can interact with room functions like solving a puzzle or battling a monster, but can also leave that room or enter commands for different functionality like saving or exiting the game. The room determines if it is solved on its own without the player’s interaction based on the solved values of its room contents.



### Object Model

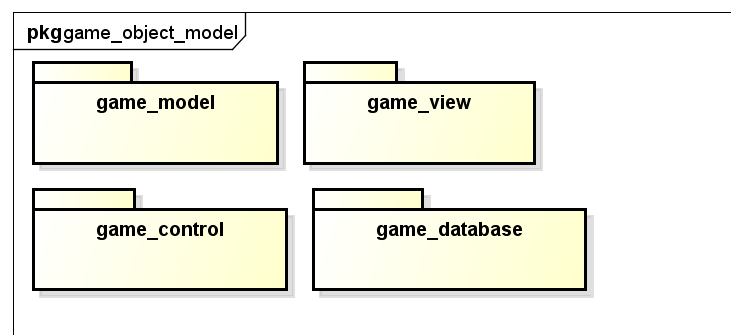
The object model is categorized into four design packages representing the model, view, control, and database of the game.

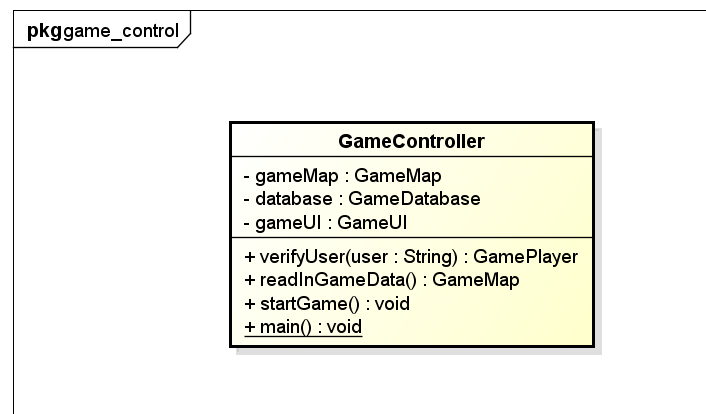
The model is consisted of entities that represent game components like the Player, Monster, Puzzles, Monsters, Rooms, and Items. Their important attributes and operations are defined within each entity diagram along with the relationships they have with each other. Both the Game Player and Monsters in the game are children of the Game Character entity as they share a lot of functionality. Items have a type attribute that depicts which kind of Item it is, Weapon, Item or Armor. Rooms contain Monsters, Puzzles, and Items, and the Game Map contains a collection of Game Rooms. Furthermore, the Game Player can contain a set amount of Game Items. Attributes and operations are listed in the diagrams below.

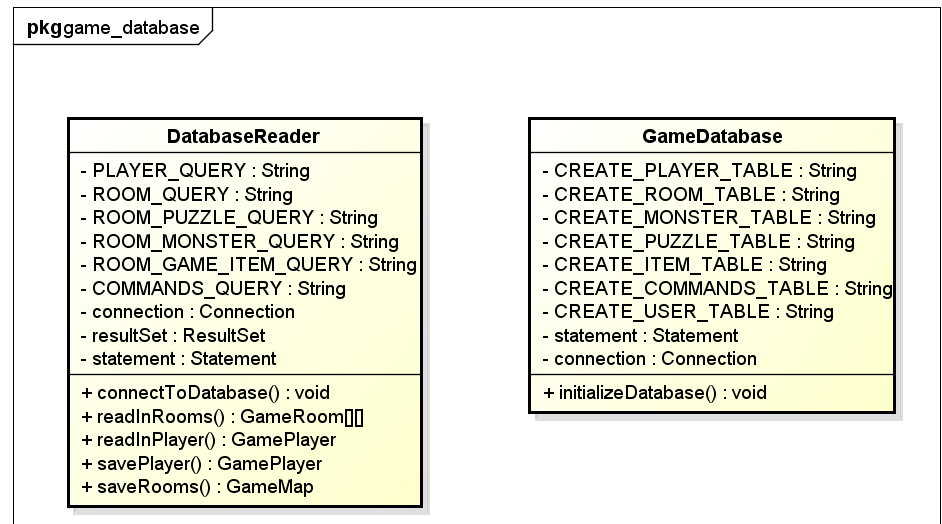
The view package models a JavaFX based UI that will be implemented as an optional UI choice for the game. The view diagram can be easier seen with the UI demo shown later in the document in the UI section.

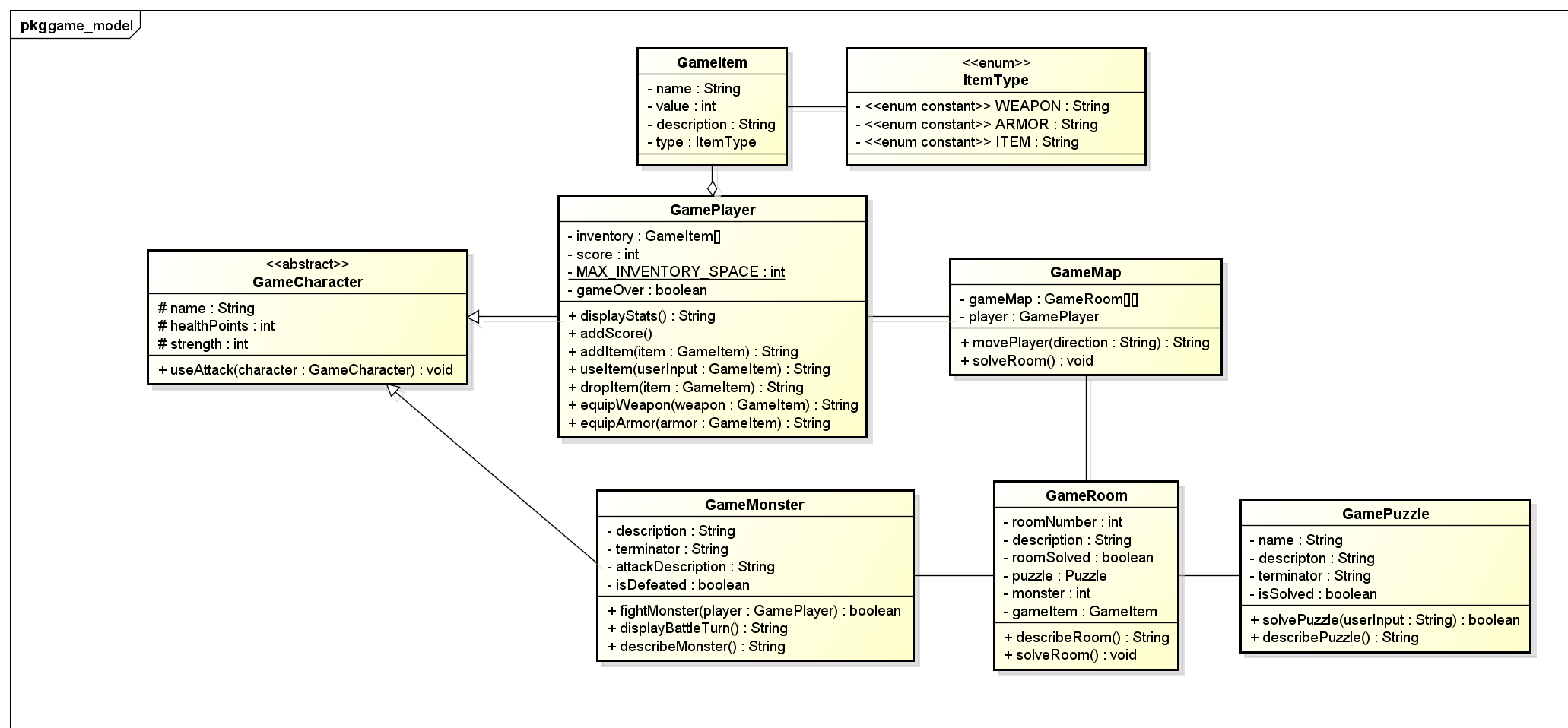
The database package will handle all saving and reading data to and from the database. This package will create the database as well as handle changes and updates. Important statements for creating the database and important operations the database will handle are listed such as reading and saving.

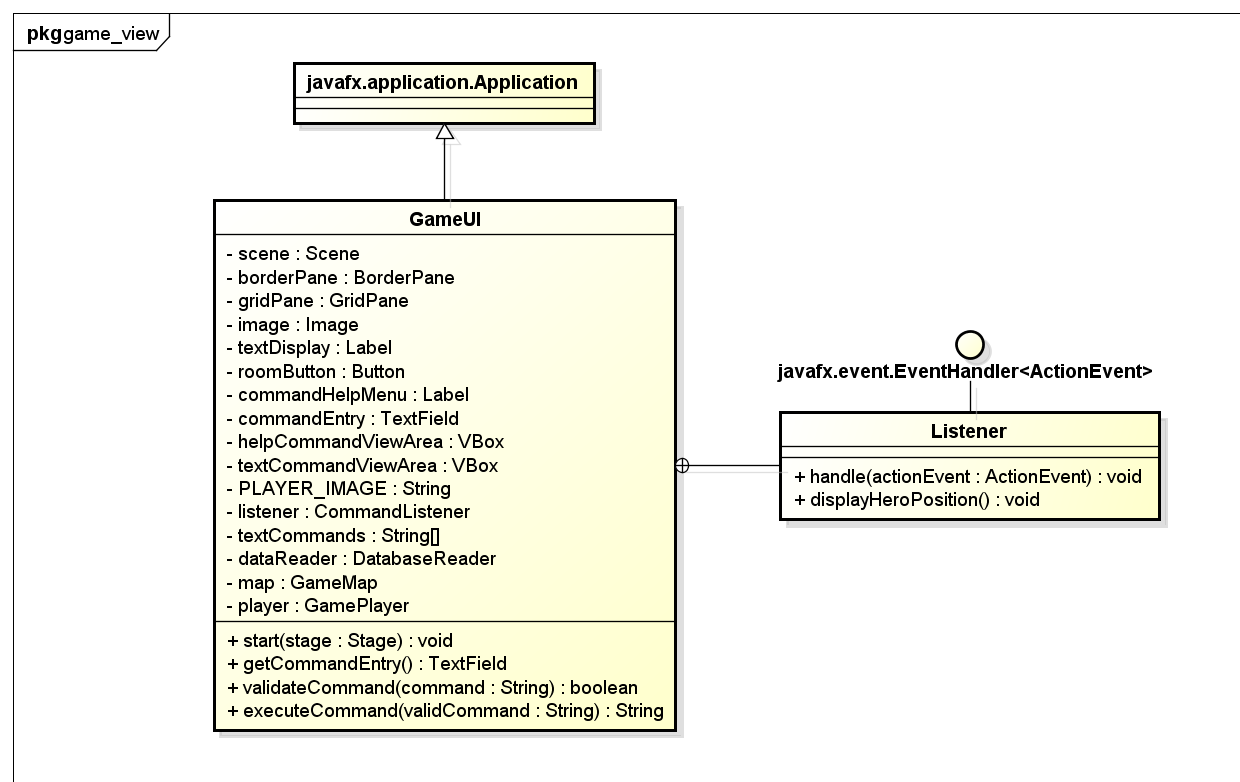
The control package will be the driver that controls all packages and simulates them appropriately. This package is responsible for handling all components of the game as they are designed in order to work together.







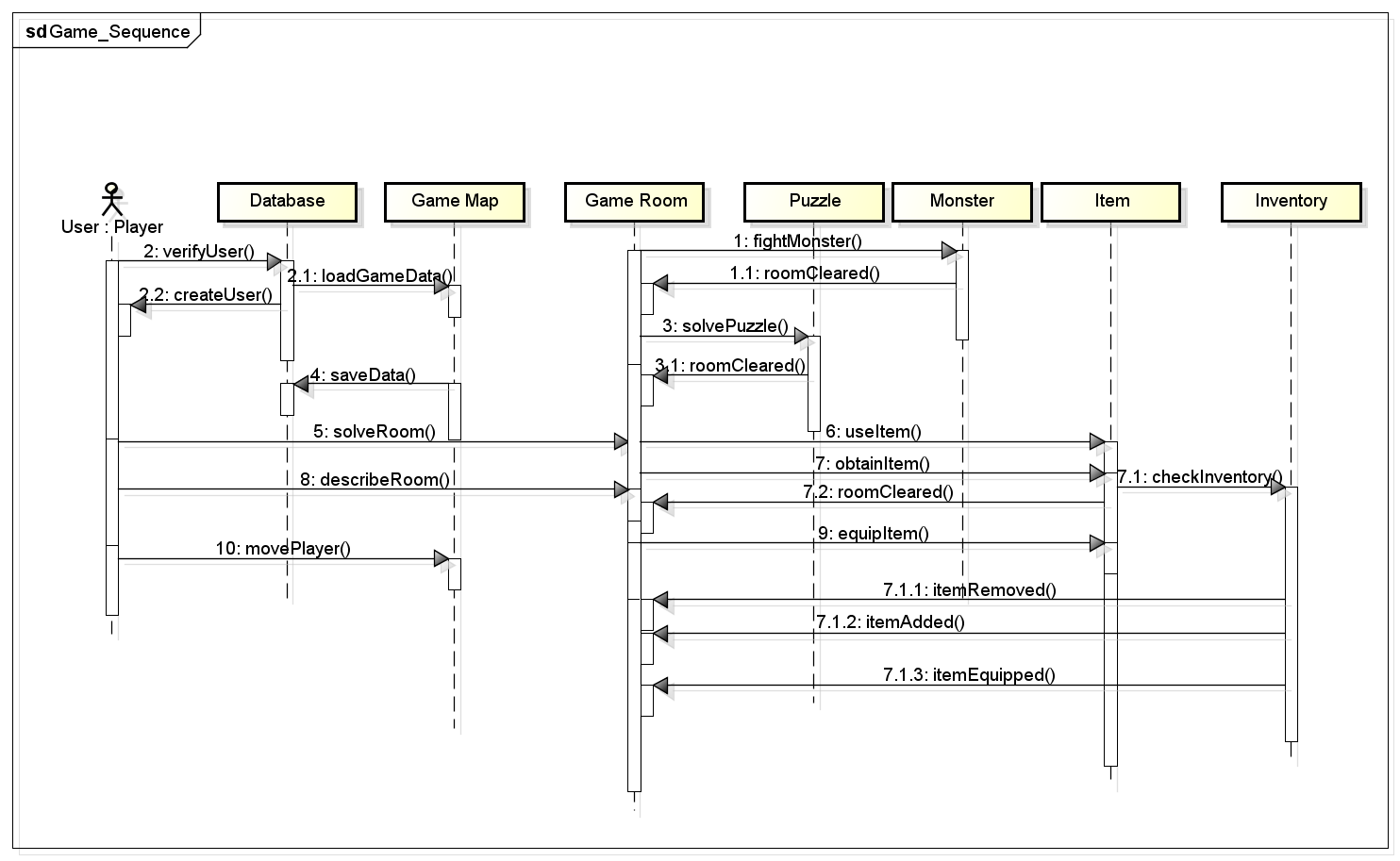




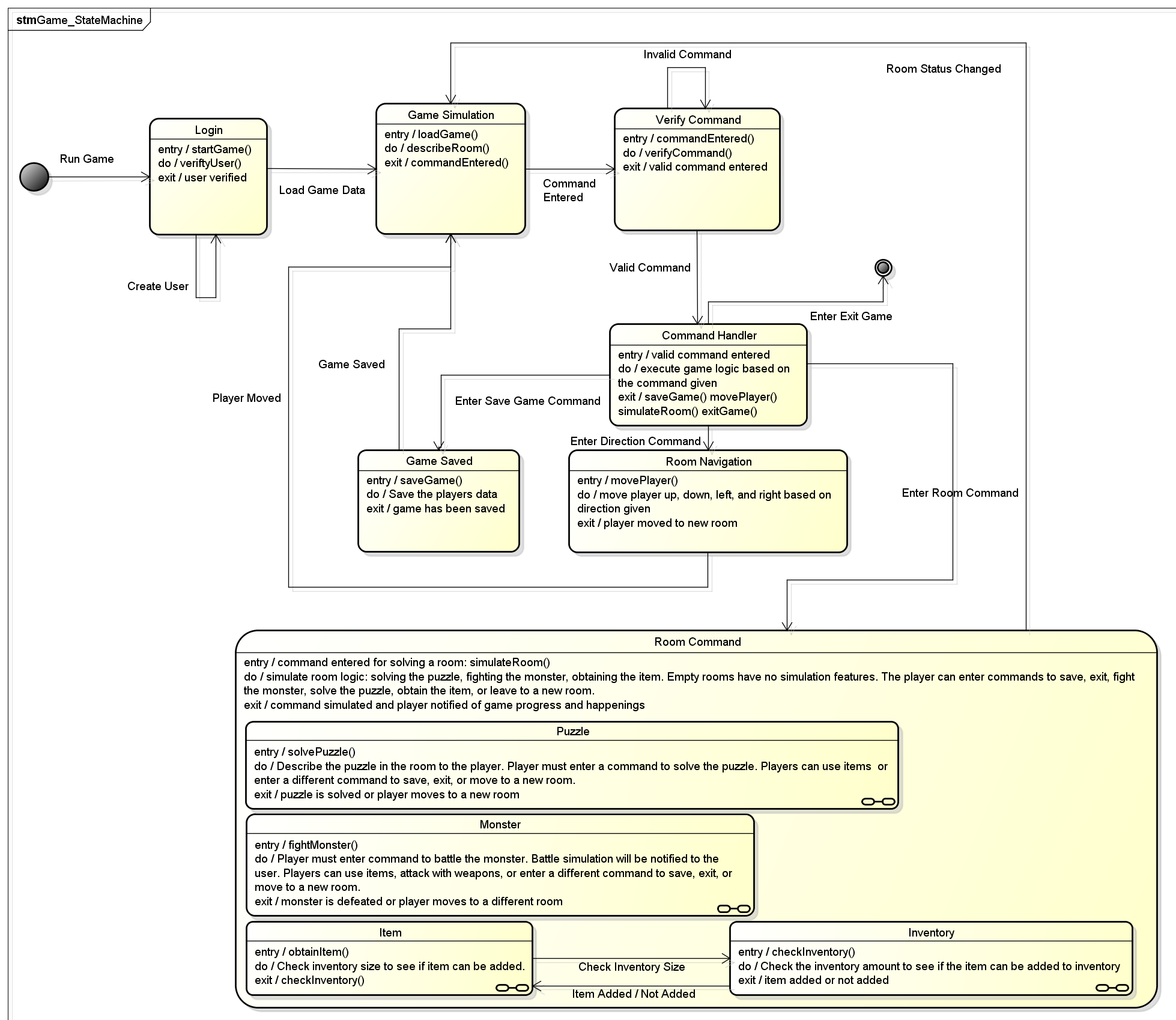
### Dynamic Model

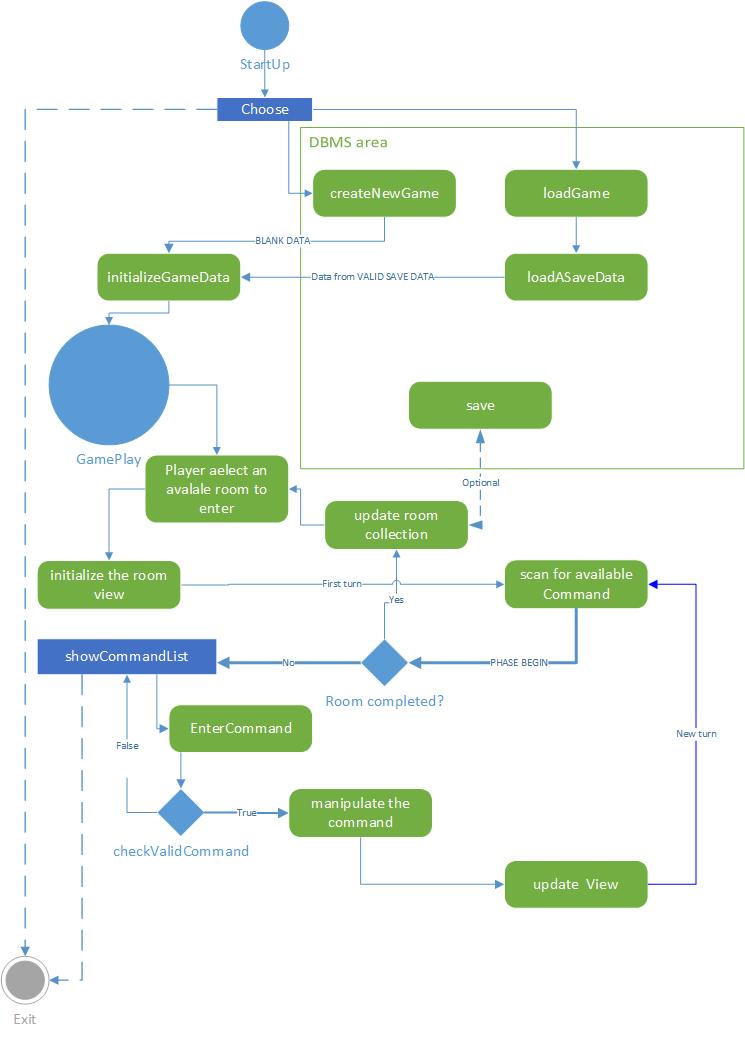
In our dynamic model the user interacts with the game in a variety of ways. They interact with the database, the game map, the game room, the puzzles/monster/items, and the inventory. All of these can be interacted with based on user inputs and methods which are shown in the diagram below.

The Dynamic model is represented by a sequence and two state machine diagrams detailing the flow of operations that happen when the player plays the game. This is shown to represent all operations in the system and how they are connected. This defines all entry and exit points of the system as it is interacted with the player. This includes external factors like the player and the database that provide input and output for the game. For example, when the player logs in to the game for the first time and creates a new user, that request is then sent to the Database for the user to be created. Once the user is verified, game data is read in from the database into the game. Now the game has data and interacts with the player. If the player decides to save their game with the save command at any point in the game, game data is then sent back to the database. These three systems of the user, database, and game and their operations between each other are modeled to accurately reflect the system.

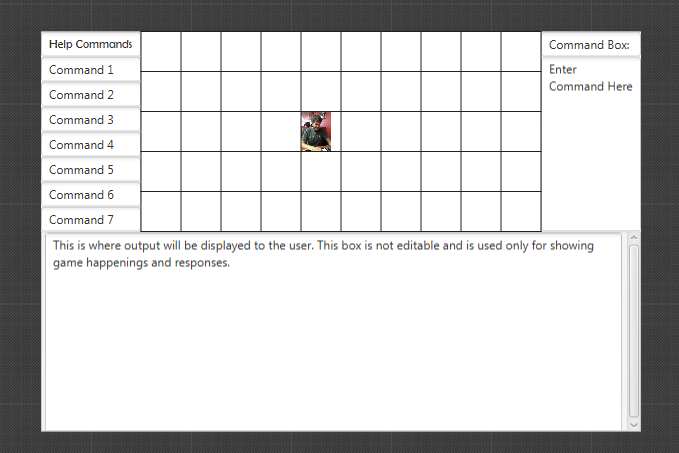


In our state machine diagram, it shows the users track through the various processes of our game. It goes from the initial running of the game all the way through the various interactions they have with the rooms and the various contents within them. This diagram walks through each of the interactions the user will have with the environment we have created.



User Interface

The User Interface for this text based game is optional. The game can be shown through a command line console terminal where all output is displayed to the user. However, a UI is allowed to be implemented and below is a design document detailing how the game will be modeled with a UI. Each section represents aspects of the game, including the game map, command list, command text box to enter commands, and a text display area to show game responses to the user.



# Glossary

Java API – Application Programming Interface that is the documentation of class libraries used to program the game.

JRE – Java Runtime Environment

This is required to run a java application on a device. Contains all the packaged libraries and necessary programs to build and run applications for users. Can be downloaded at oracles home website for free use to the user.

DBMS – Database Management System

This is the storage system used for monsters, puzzles, rooms (game map), and user saved data in the game.

RPG – Role Playing Game

Fantasy game that involves a hero trying to accomplish a task, objective, or quest. Typically they have monsters, magic, expansive world or dungeon, and mythical elements that define the culture of the game. Players usually are involved in a type of combat system that is real time or turn based.

TBG – Text Based Game

All functionality the player has in the game is issuing text based commands to the console. They are then notified on the outcomes of their commands.