**Design Document**

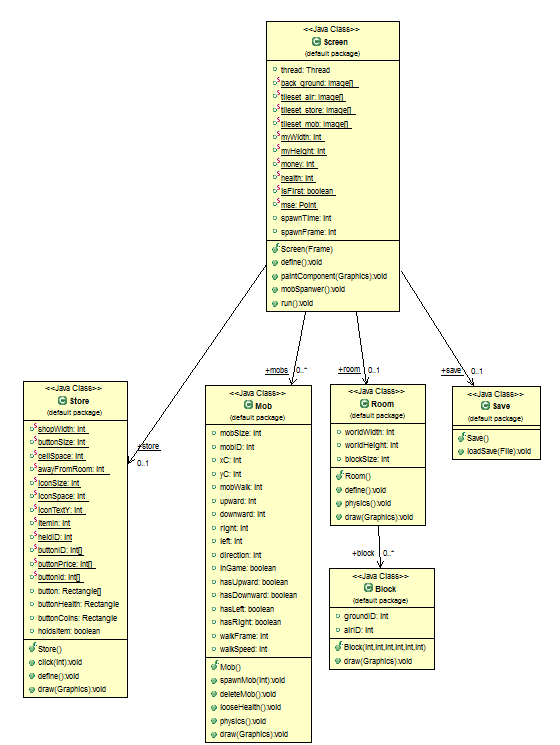
**CSCE-361**

**Team 11**

**Bryce Andresen, Ronnie Gallagher, Dat Nguyen, Hoa Nguyen**

**Tower Defense Game**

1. Introduction
   1. The purpose of this design document is to display the entity relations and high-level architecture for the Tower Defense Game. This document contains a UML diagram outlining the relations between the different classes within the main program. It also contains a schema diagram outlining the structure of the utilized database.
2. Architecture
   1. Introduction
      1. Untitled Diagram(2).pngThe high-level architecture of the system is based on a view-model-controller system structure. The system allows user interaction via a Java GUI layout embedded into an HTML webpage. The user can interact with the GUI, which will update and run the view component of the architecture, which will return the modified GUI components. The user can also use the controller system to send database queries to the MySQL server, as well as access game data via the Java GUI libraries. The model component can return requested data
   2. Modules
      1. Controller component
         1. The controller consists of the Java code used to define the system, including the different classes and subroutines that carry out the functions of the game. The classes perform these subroutines to interact with the view and model components of the system.
      2. View component
         1. The view component consists of the GUI components that are generated by the Java code, consisting of graphics included in the Java libraries as well as custom graphic assets. When commands are given through the controller, the necessary GUI changes are modified made visible to the controller.
      3. Model component
         1. The model component consists of the MySQL database and Java libraries that hold all of the data necessary for the system to run. The controller can send data queries to the MySQL server and the server will return the data to the controller, either directly via data modifications in the main system or passed to the view component to be formatted and presented to the user in the controller system as viewable data. The controller system also draws from the model system via Java libraries to create GUI components.
3. Class Diagrams
   1. The UML class diagram shows the basic relations of the Tower Defense Game. The specifics of each relation and component are listed in section 5.



1. Database Schema

Untitled Diagram(3).png

* 1. Database Schema Information
     1. The database consists of two tables, one for players and the other for scores. They will be related by a one-to-many relationship as one player can have multiple scores. The data in this database will be used to generate a leaderboard consisting of the highest scores achieved by game players.
        1. The player table is composed of a username and player ID. The username is generated by the player upon completion of the game. The user ID is the primary key used to link the player and score tables.
        2. The scores table contains a player ID, score ID, score value, date of game completion, and game difficulty. The primary key is the score ID. The foreign key is the user ID which will link the score data to the user.

1. Class Information
   1. Screen
      1. The screen class will generate the necessary GUI that will allow the user to interact with the other classes and methods. All changes to the GUI are affected by the other classes.
   2. Store
      1. The store class will generate a GUI that will overlay the screen and allow the user to buy towers and other game objects, which will then load the necessary assets to the screen.
   3. Mob
      1. The mob class is responsible for spawning and controlling the enemies produced for each mob. An enemy in the mob can be assigned to spawn and begin its path through the map with the spawnMob(int) function. Every time an enemy is hit by a tower, the loseHealth() function is called to decrease its health until it reaches zero and then triggers the deleteMob() function. Enemies are also assigned a graphic with the draw(Graphics) function.
   4. Room
      1. The room class handles the map length and width, and draws the layout (as depicted in Project Requirements section 4.1) to the screen.
   5. Block
      1. The block class handles the areas on the map where players can place towers. Each block is assigned to target either air or ground enemies, as well as a size, location, and graphic for the GUI.
   6. Save
      1. The save class is responsible for saving and loading game status. Save is handled by the function Save(), and load is handled by the function loadSave(file). These will allow for users to save and load the games they play.