1. Structural Design

|  |  |
| --- | --- |
| **Data** | ***Interface => class*** |
| List of Scores to be displayed | List => ArrayList<Long> |
| Level Map | Tile[][] |
| List of Entities in the Room | List => ArrayList<Entity> |

We have decided to use an ArrayList to hold the scores obtained by the user. When the “Game Over” screen is shown, the score that was just obtained from the last game will be displayed along with some of the highest previous scores. Since the capacity of the ArrayList is mutable, we can continuously add scores to the list as necessary. The ArrayList will be sorted in descending order and several of the first integers of the ArrayList will be displayed in the scoreboard as we iterate through a finite range of the list.

The map for each level will be formatted as a grid that allows the user and the enemies to move tile by tile or cell by cell. In order to implement this, we will first use a 2D array of Tile objects where a boolean value will indicate whether or not a tile is passable. Additionally, each tile object will be associated with an image and label, to which an image shall be placed inside each cell of the 2D array for the display. Different tiles with different images will be placed in each cell. Objects such as torches will need to use the 2D array in order to combine with the Tile at their designated positions. The Room class will also be using the Tile map in order to properly place and display the enemies and the user and allow access to their locations and positions.

The ArrayList of Entities in the room of the level is kept in order to maintain a list of references to the Entities in each room. These references will be kept in order to allow both the User and the Enemy to communicate with each other. The Room, with the ArrayList of Entities, will be the means for the User to get the reference to the Enemy (and vice versa) in order to attack entities, kill entities, move entities, and remove entities including itself.

1. Class Diagram:

