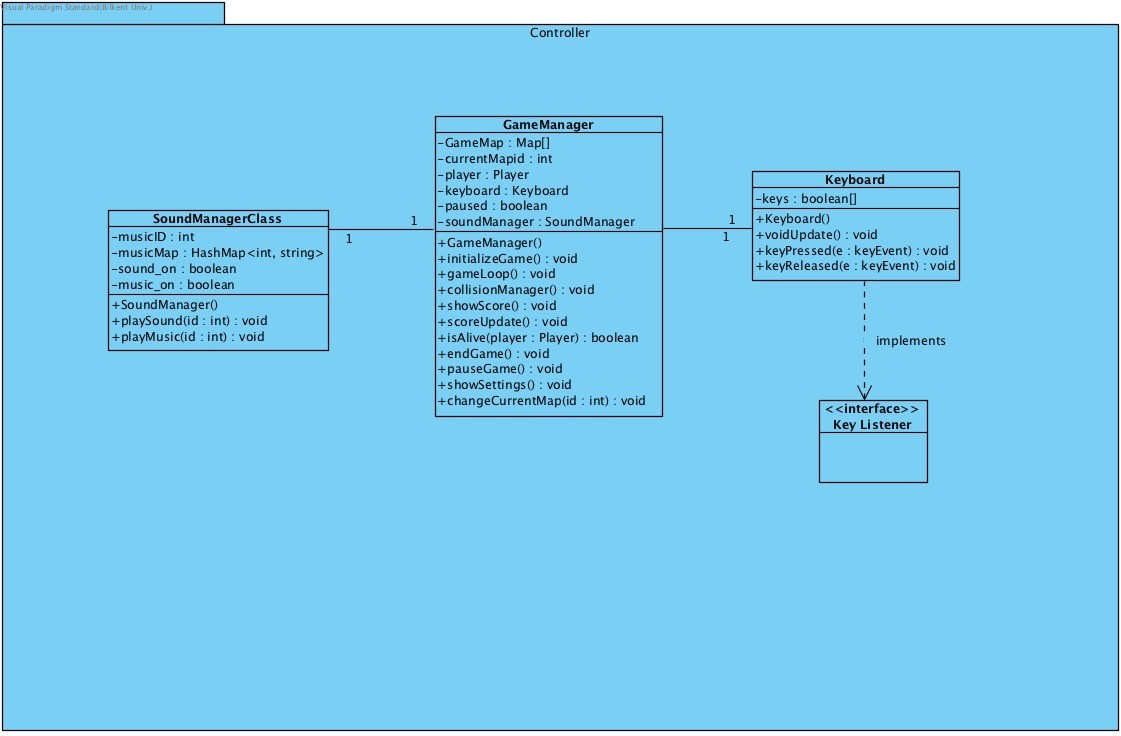
### Game Controller Subsystem

Game Controller subsystem will be the inter-layer in three-tier system which will contain necessary classes to manage user input, game sounds and detect any change in game state. The most important class in this subsystem is Game Manager class has access to all the classes in the subsystem. Game Manager class has the gameLoop() method which runs the game until there is an interruption or the player dies. Keyboard class uses KeyEvent interface and detects any user input and GameManager uses this data from the Keyboard class. SoundManager class plays or disables any sound and music in the game.



It is also controlled by GameManager class which manipulates it with respect to the user input. GameManger has access to the model and view subsystems of the game as well.

Basically, it is the heart of our game.

* + 1. **Game Manager Class**

Game Manager class is the main class which runs the game. Game loop runs in this class. It performs the actions requested from users. The state of game is determined by this class, for example it determines the game if it is in pause state or run state.

###### Atttibutes:

private GameMenu gameMenu: This is initialized when game runs. It offers players the menu view of the game which they can start the game or change the settings, or look at high scores, help, credits etc.

private boolean paused: This determines the state of the game. It can be paused or running.

private GameMap gameMap[ ]: This is an array of Map objects. It contains the maps which are the layers in the game. Every map contains the rooms where player walks. Shortly it is the game plan which enables the game manager to manage game mechanics in the model subsystem.

private Player player: This is player object. It is initialized when game initializes. Game Manager class uses player object to manage its actions. Game manager is like a bridge between the room and the player. Everytime player changes a state game manager updates the rooms conditions and etc.

private int currentMapID: Game manager uses this integer value to determine the current map that the player is inside of.

private SoundManager soundManager: Sound Manager object enables Game Manager to control sounds and music in game. Everytime a settings is selected Game Manager passes this information to Sound Manager class to change the sound.

private Keyboard key; Keyboard object enables Game Manager to control key actions and game mechanics in game.

###### Constructor:

public GameManager(): It initializes the attributes and some other necessary functions. It initializes Game Manager class.

###### Methods:

public void gameLoop(): This method runs as long as player plays the game and updates continuously. It checks all the changes of the game.

public void initilizeGame(): If user presses the play button, this method initializes the game sets de.

public void manageCollision(): This method invoked by the keyboard press of the player. It calls the getCurrentRoom() method of Map class. Then, it uses the checkCollision method of the Room. Room returns an object to Game manager class. Game Manager checks if this object is null or not. If the object is null, player can move and update its position. Otherwise, if the object is an Item, GameManager class calls addItem() method of the Player and player gets upgrade. If the object is an Obstacle, then player can not move. Basically, this method manages the movement of the player.

public void isAlive(): Checks if player has health.

public void endGame():This method uses isAlive() method to check if the player can still play the game. If the player has no health left, then the Game Manager finishes the game.

public void pauseGame(): This method is invoked by the user keyboard input if the pause key is pressed. Game Manager uses this method to stop the game and using the Game Menu object it makes the menu appear on the screen.

public void showSettings(): This method is invoked by the player input and it displays the sound settings on the screen by using Game Menu object.

public void changeCurrentMap(): This method is used when GameManager detects wether the player passed to another map(layer) and updates the view and the generates the new map.

## Sound Manager Class



This class is for controlling sound effects and music in game. Game Manager class uses this class as an attribute and together with keyboard class, Game Manager changes the sound states according to user input.

###### Atttibutes:

private boolean sound\_on: This attribute is for enabling sound effects in game or disabling it. private boolean music\_on: This attribute is for enabling music in game or disabling it. private int soundMap<int,string>: This map hashes id’s of sounds with respect to their name.

private int musicMap<int,string>: This map hashes id’s of map’s current id with respect to their names.

###### Constructor:

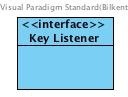
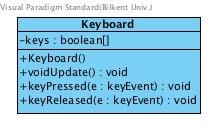
public SoundManager(); initiliazes object, sound\_on and music\_on boolean is true by default.

###### Methods:

public void playSound(int soundID); Takes sound id for example collision sound, projectile sound etc and plays it by using the musicMap which holds names of sound.

public void playMusic(int musicID); Takes current map’s musicID and plays it.

**3.5.3. Keyboard Class**



This class enables Game Manager to get keyboard inputs from player. This class implements KeyListener interface. Moreover, Game Manager uses this class to manage view and model subsystems in order to change sound settings, show menu, move player, attack and etc.

###### Attributes:

Private boolean keys;

###### Constructor:

Public Keyboard(); Initializes the keyboard object for game. The keys are false by default.

###### Methods:

Public void keyPressed(KeyEvent e); Determines pressed key. Public void keyReleased(KeyEvent e); Determines released key.