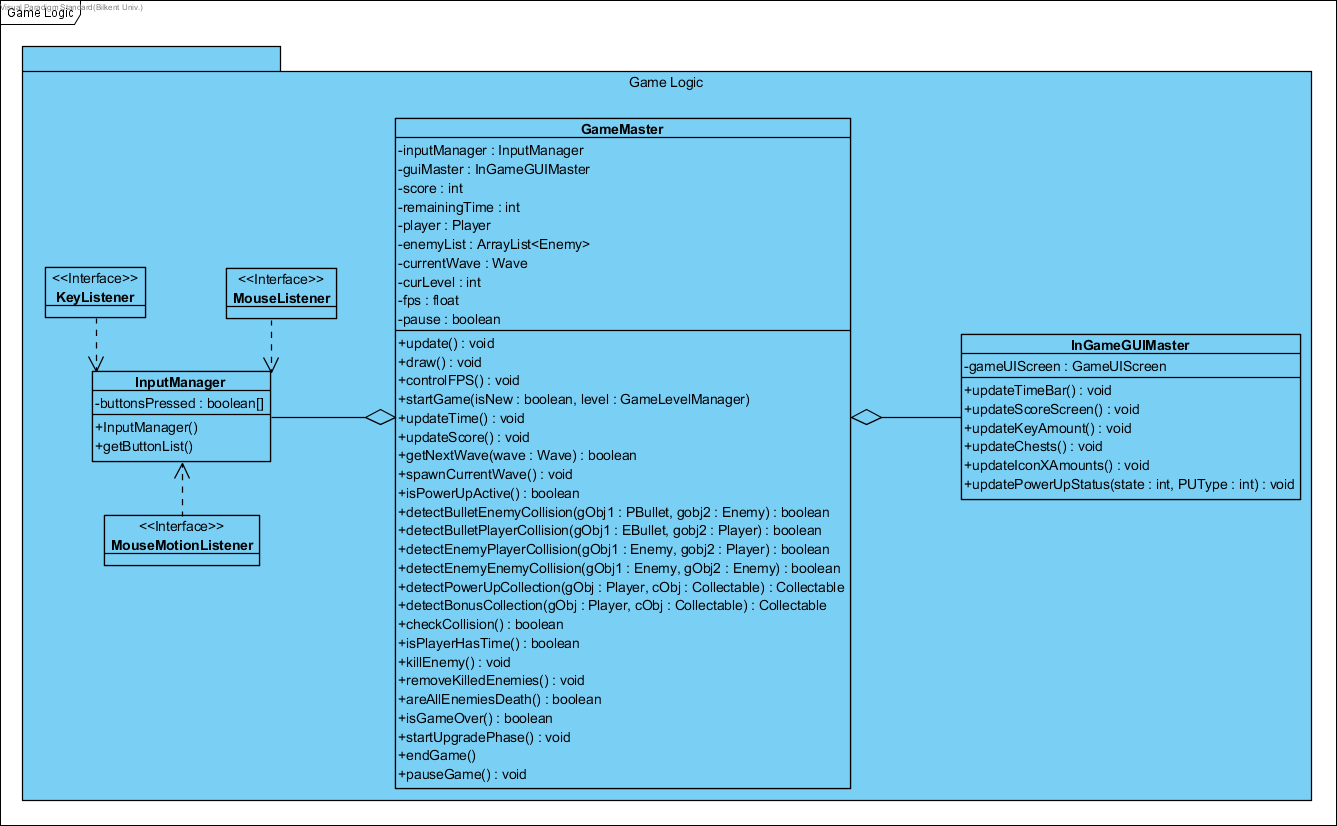
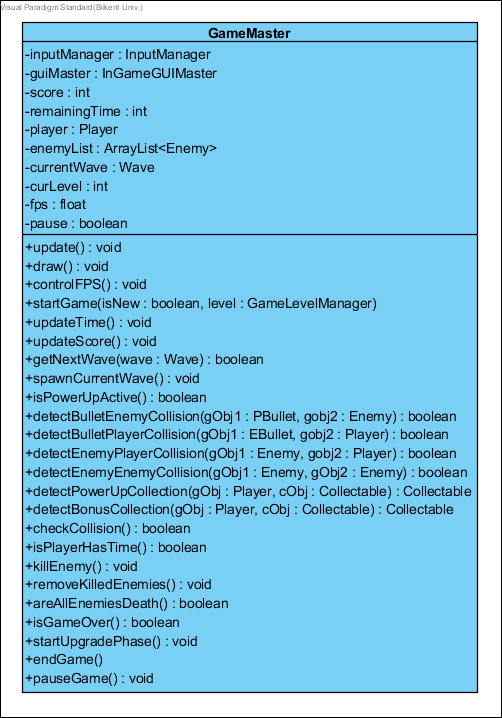
### Game Logic Subsystem



In the Game Logic subsystem, our controller objects are grouped together to manage the actual game dynamics and game logic. We have GameMaster, InGameGUIMaster and InputManager. These classes will be explained in detail, in this section.

## 

### GameMaster Class



GameMaster class is the Façade class of the Game Logic subsystem, it performs the proper operations according to the requests that came from User Interface subsystem, and also this class runs the game in a loop.

#### Attributes:

**private InputManager inputManager:** this attribute is used for detecting user actions in the game.

**private InGameGUIMaster guiMaster:** this attribute is used for updating graphical user interface on the game screen.

**private int score:** it is used for player’s score to represent the success in the game.

**private int remainingTime:** it is used for determining the game time representing both real-time and player’s health.

**private Player player:** this attribute initializes a player object to use in the game.

**private ArrayList<Enemy> enemyLists:** this attribute initialize enemy list taking from the level subsystem.

**private Wave currentWave:** this attribute initialize a wave of the enemy by taking the enemy list when the player killed them

**private int curLevel:** this attribute holds the which level is called.

**private float fps:** this attribute takes the frame per second to determine the movements of game objects.

**private boolean pause**: this attribute is used for whether the game is paused or not.

#### Methods:

**public void update():** runs a loop in which the system is updated continuously until the breakpoints(such as pause, game over, or finish game).

**public void draw():** draws the all game objects and UI on the game screen according to the level manager.

**public void controlFPS():** holds the average frame per second until the 30.

**public void startGame(isNew: boolean, level: GameLevelManager):** starts a new game by taking level information (to call level’s enemy list).

**public void updateTime():** updates the time by decreasing while the game processes and when the player takes damages.

**public void updateScore():** updates the score when the player success to kill enemies.

**public boolean getNextWave(wave: Wave):** calls the new wave of enemies when all enemies in the previous wave are killed by the player.

**public void spawnCurrentWave() :** if getNextWave is true, this method creates current wave enemies.

**public boolean isPowerUpActive():** checks whether the player has activated any power-ups or not.

**public boolean detectBulletEnemyCollision(gObj1: PBullet, gobj2: Enemy):** detects the Player Bullet and Enemy collision, in this type collision enemy takes damage and its health decreases. After the collision, bullet disappears.

**public boolean detectBulletPlayerCollision(gObj1: EBullet, gobj2: Player):** detects the Enemy Bullet and Player collision, in this type collision player takes damage and its health decreases. After the collision, bullet disappears.

**public boolean detectEnemyPlayerCollision(gObj1: Enemy, gobj2: Player):** detects the Enemy and Player collision, in this type collision both enemy and player takes damage and their health decreases.

**public boolean detectEnemyEnemyCollision(gObj1: Enemy, gObj2: Enemy):** detects the Enemy and Enemy collision, in this type collision both enemies hits and removes each other without any damages.

**public Collectable detectPowerUpCollection(gObj : Player, cObj : Collectable) :** detects the Player and Power-ups collision. After the collision, the player takes the power up and isPowerUpActive is returned true. And also, the power up disappears on the game arena then it goes to the power-up box on the corner of the game screen to be used later in the game.

**public Collectable detectBonusCollection(gObj : Player, cObj : Collectable) :** detects the Player and collectable item(key, chest or coin) collisions. After the collision player takes the items and one of these InGameGUIMsster methods (updateKeyAmount, updateChests, updateIconXAmounts) is called and this item is added player’s inventory. And also, the item disappears on the game arena.

**public boolean checkCollision():** checks any collision, if there is any call the detectCollision methods and return true, else return false.

**public boolean isPlayerHasTime():** checks whether the player has the time or not. If the time is over then returns false, otherwise, returns true.

**public void killEnemy():** changes the death-flag to true if the enemy has no life.

**public void removeKilledEnemies():** checks death-flags of all enemies and removes them if the flags are true.

**public boolean areAllEnemiesDeath():** checks the death-flags of enemies in the list of the enemy, if all of them are true then this method is also return true, otherwise, return false.

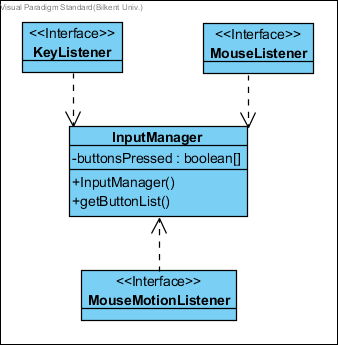
**public boolean isGameOver() :** checks the player has enough time (if isPlayerHasTime is true) or not. This method returns true if isPlayerHasTime is false, otherwise, returns false.

**public void startUpgradePhase():** before each level, this method is called and upgrade screen appears.

**public void endGame():** exits the game when the player presses the “exit the game” button.

**public void pauseGame():** pauses the game and call the pause screen when the player presses the pause button.

**InputManager Class**

****

InputManager class is design to detect and designate the user actions performed by mouse (such actions to turn the barrel to shoot enemies) and keyboard (such actions to move the player in the game arena and to pause the game). Therefore, InputManager class implements MouseListener, MouseMotionListener and KeyListener interfaces of Java.

#### Attributes:

**private boolean buttonsPressed:** this attribute checks whether the specified buttons are pressed or not.

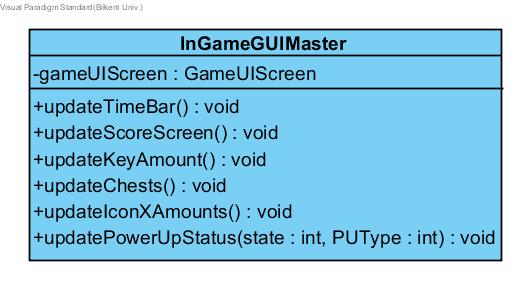
#### Constructors:

**public InputManager()**: it initializes the attributes of the GameManager for the first run of the system.

#### Methods:

**public getButtonList()**: this method specified the list of buttons in the game.

### InGameGUIMaster Class



#### Attributes:

**private GameUIScreen gameUIScreen:** User Interface screen for the in-game GUI. It allows the connection between the Game Screen Elements package’s GameUIScreen class and Game Logic package’s InGameGUIMaster.

#### Methods:

**public void updateTimeBar():** Updates the time bar of the player depending on the events occurring in the Game Master.

**public void updateScoreScreen():** Updates the score screen in the game depending on the player’s score collection taken from the Game Master.

**public void updateKeyAmount():** Updates a number of keys in the game depending on the Level Manager’s Collectables class.

**public void updateChests():** Updates the chests in the game depending on the Level Manager’s Collectables class.

**public void updateIconXAmounts():** Updates the number of Items (Number X “Icon of Object”) in the user’s inventory.

**public void updatePowerUpStatus(state: int, PUType: int):** Updates the current PowerUp icon on the screen by taking its states and types.