Vamsi

Change 1:

Within the MapGenerator class, text needs to be displayed when various game mechanics occur, for example, LeverUnlocked keeps track of the lever. When unlocked, it shows the message “lever unlocked”. This process happened through calling a function in a different class (GamePanel) called showMessage, this led to high coupling, to fix this: we added a new function to the MapGenerator class called showMessage, which in turn calls the function it shares a name within the GamePanel class. This reduced feature envy for the MapGenerator class.

Change 2:

loadImage() function was added to Entity and MainObject classes. This method was introduced to tackle the high code repetition in methods like getGoblinImage() and getPlayerImage(). Every image-loading method repeated the same ImageIO.read(...) method and a try-catch method for loading images; by implementing loadImage(String path), we were able to move the ImageIO.read methods and the try-catch method into a reusable loadImage method. The image-loading method was also too long because it handled both loading and assigning the images, adding the getImage() method takes the responsibility of loading the image away from the image-loading methods and makes them solely responsible for assigning the images making it shorter, and reduces the high coupling previously present. This solution improves cohesion, reduces coupling, and enhances code structure.

Change 3:

In the collision checker, we had repeated lines that were present in various methods across the class. The purpose of these lines are 1: for entities, constantly calculate and update the collision area in the map, 2: for objects, calculate the collision area on the map once and store these. We added the updateCollision method, which took the calculation and assignment that happened in the method itself and instead, updateCollision will calculate and assign the appropriate collision areas.

Change 4:

In map class, primitive attributes were used as a measure of time, specifically, 60 and 120 were used as guides, instead, these are replaced by final int oneMin and twoMin. These changes eliminate magic numbers with named constants.

Maxime

Change 5:

updateAnimation() and getSpriteForDirection() were added to the Entity class. This method was introduced to handle **duplicated code** in the Goblin and Player classes, as the implementation was the same. The only difference was that the Player class checked for collision and playerInput before updating the animation, while the Goblin only checked for collision. To overcome this, we used the **extract method** and added updateAnimation(). Then the method canUpdateSprite() (boolean) was implemented in the Entity class, returning if collision was on. We override this method in Player to check both collision and player input. Then we checked canUpdateSprite() in updateAnimation() before updating the animation. This change improves maintainability by centralizing animation logic and allows for easier changes to animation behavior across all entities.

Change 6:

The draw() method was added to the Entity class in order to reduce **duplicated code** in the Player and Goblin classes. Since the camera was following the player, we had to check to see if the Goblin was inside the camera view before drawing it. Besides that, the remaining implementation was the same. We realized that the player would always be in the camera view, so this check wouldn't affect the player being drawn. Hence we used the **extract method** and added the draw() method to Entity. Next, added the method isVisibleOnScreen() (boolean) to check whether the entity was on the `screen before drawing it in the draw() method. This change reduces the overall codebase size and makes future drawing-related changes simpler by having a single implementation point.

Change 7:

The checkEnemyCollision method had the properties of a **long method**. Using the **extract method**, we introduced the method changeDirection(). This method helps choose the direction the Goblin takes when colliding with another Goblin to prevent them from overlapping. By clearly expressing the purpose of the method in the name, we improved code readability. This change makes the collision handling logic easier to understand and maintain, facilitating future code reviews.

Change 8:

In the subclasses of Entity, GamePanel was an attribute for Goblin and Player classes, but not an attribute of Entity. This represents both **feature envy** and an encapsulation error. To fix this, we used the **move field** solution and removed this attribute from Goblin and Player. We added it as a protected attribute to Entity so the player and goblin would inherit it. Then we passed GamePanel to the constructor of Entity and initialized it there. After this, we called the Entity constructor in the Goblin and Player constructors, allowing them to have the attribute initialized when their constructors were called. We then changed the constructor calls for Player and Goblin throughout the codebase. This change creates a more logical class hierarchy, simplifies the subclasses, and follows the principle of putting related data and behavior together(OOP Design).