Assignment 3 - Group 10 - Ewan Brinkman and Hoi Chun Hogan Mok

Code Smell 1 - Code duplication - Duplicate code to create entities in levels
Commits:

- 658d421cbb08145cbe95df48114a10d642536ec5 - refactor: remove duplicated entity creation code in the Level class and create an abstract Entity factory

Explanation and code changed: our code had a different method to create each entity type, such as enemies, doors, bonus rewards, and so on. However, all functions took in the same parameter of a position. The only difference was each method used the corresponding factory for creating the entity. To resolve having duplicated codes in Level.java, we created an abstract entity factory (EntityFactory.java in the factory package) with a create method, and made all other factories in the factory package extend it. Then, we replaced all the methods in Level.java to create entities with a single method (createEntity). This new method to create entities takes a factory as an argument, and creates an entity using the given factory.

Code Smell 2 - Code duplication - Duplicate code to get level data
Commits:

- c66db2ffb53f2931b826fc923793d844b990cd87 - refactor: remove duplicated for loop in Level class

Explanation and code changed: the duplicated code was in the loadCurrentLevel method in Level.java (in Level.java in the level package). Especially during iterating different entities' json array in level.json, there is a nested for loop to iterate through the nested array, and also, getting the spawning location of the entities inside the inner array. To refactor the duplicated code, we have created the loadEntityArray method (in Level.java in the level package), and takes as arguments the key name, the json object, and the entities factory as parameters to iterate through the arrays and create entities properly.

Code Smell 3 - Long list of method parameters - Parameter list to create sprites is too long
Commits and code changed:

- 7ea3a5223f6f5c2a75622902a3abdb32a7bb6cf1 - refactor: add a class to store sprite data for creating sprites

Explanation and code changed: a sprite needs a lot of data to create itself. It needs information about its enclosing rectangle, relative hitboxes, offsets for drawing, animation images, animation image ordering, and the initial animation. This made the parameter list of the constructor for the Sprite.java class (in the sprite package) very long. Many classes extend the sprite class, and so had long constructor parameter lists as well. To fix this, we did the "Introduce Parameter Object" solution from slide 32 of lecture 17 of the course about refactoring. The new SpriteData.java class (now in the data package which is in the sprite package), stores six data elements mentioned earlier. Then, everything using the sprite was changed to reflect the six arguments being replaced by one class. Factories in the factory package in the entity package were changed to create a SpriteData class, and the rest of the classes in the entity package had their constructor parameter lists changed accordingly. Plus, tiles in the tile package which is in the level package had their constructor parameter lists updated accordingly as well. Also, the createTile method in Tileset.java in the tileset package created a SpriteData class for tiles.

Code Smell 4 - Dead code - Unused methods
Commits and code changed:

- *8d7a6b619d6576b8a036d53d38bc233ba9d9a6e0 - refactor: remove unused functions*

Explanation and code changed: removed getDifficulty (in Level.java in the level package), getBossAcceleration, getBossFriction, getBossMaxVelocity, getBossMinVelocity, getBossTrackingDistance (in ConfigReader.java in the resources package), getTileNumber (in Tile.java in the tile package), setPosition (in Sprite.java in the sprite package), resumeAnimation (in Drawable.java in the draw package), unsetTarget (in CameraManager.java in the draw package), getTileIndex (in TileLayer.java in level package), and setVelocity (in MoveableEntity in the moveable package) that we have not used in our game.

Code Smell 5 - Badly structured project (i.e., file setup) - Bad structure to setup files
Commits and code changed:

- *9e0cc434e579ec9291c6a14e4e82141e50b61aa3 - refactor: move classes into new folders*
- *3d06ced345e98306269b5a084528eac67e99f0e2 - refactor: further reorganize the project structure*

Explanation and code changed: classes and their package names didn't match that well. Changes made include moving the Sprite.java class into its own sprite package in the source code root directory. The Sprite.java class used to be in the sprite package which was in the entity package. However, this was bad since a sprite is not even an entity. Next, Hitbox.java and SpriteData.java were put in a data package in the sprite package, and CameraManager.java, Drawable.java, and DrawOffset.java were put in a draw package (since they are all about drawing) in the sprite package. Furthermore, the util package was reorganized. The Vector.java and CollisionChecker.java classes were moved into their own physics package in the util package, since they both help with game physics. Animation.java and TextGenerator.java were put in a media package in the util package, since they deal with various media (images and text). Next, the tileset package no longer has a package for DungeonTileset.java, since it wasn't needed for anything. Tiles were also moved into a tile package in the level package, since they were used by the TileLayer.java class. Moreover, a movable package was made for movable entities: MovableEntity.java, Player.java, Enemy.java, and the item classes in the item package. Finally, the manager package had various classes to help manage the game, but was very focused. So, every manager class was moved into different packages. AudioManager.java went into an audio package with Audio.java from the util package InputManager.java was moved into a game package in the app package with Game.java. MapManager.java was moved into the level package, and finally, CameraManager.java was put in the draw package in the sprite package, as mentioned before.

Code Smell 6 - Lack of documentation - Incomplete descriptions
Commits and code changed:

- fe77a1789d59f39ab6b207ef50ee95bd70f69881 - refactor: adding details to the descriptions in Level.java, TextGenerator.java, and Door.java
- 2f93bba675676d8b212577e7288b8fa45e1d4307 - docs: update the Javadoc version of the TextGenerator class

Explanation and code changed: added details to the descriptions for open(in Door.java in the moveable package), TextGenerator(in TextGenerator.java in the media package), and resetMapIndex(in Level.java in the level package)

Code Smell 7 - High coupling - Update in Door depends on Game and Player
Commits and code changed:

- e211fae47140b77d8696830abafb78b15c6b575d - refactor: remove high coupling in door and game
- a46e7f9f1da56a9b838e40b84488d1e3df8e330f - refactor: remove an unused import statement in the door class

Explanation and code changed: the update in Door (in Door.java in the moveable package) is highly dependent on Game and Player. We have moved the operations from the update in Door to the update in Game (in Game.java in the game package) to reduce the dependency between Door and Game.

Code Smell 8: - Methods that are too long and that could benefit from being refactored - Game update method is too long
Commits and code changed:

- ec8917cbda927238fe9622a966c8a41d872566a8 - *refactor: make the game update method smaller by splitting it into components that go together*

Explanation and code changed: the Game.java update method was getting a bit big, and could be split into more methods in Game.java. Created methods are: deleteEntities for deleting entities, checkCollisions for checking for collisions not checked by entities themselves, and applyInput for getting the results of player keyboard input not handled elsewhere.