REPORT

Data clumps: We noticed data clumps in the Game class, notably, the tileSize attribute was often called in a static way. We resolved this problem by creating a new class, GameSettings that stores the values, changed the access modifiers from public to private, and used getters to access values (commit SHA: 08f4a91). By extracting the values into a new class, we no longer have to access the tileSize attribute in a static way, which improves the encapsulation and readability of the code. The use of getters allows us to access these values in a more controlled manner, ensuring that they are not modified unintentionally. This refactoring also makes it easier to change these values in the future, as they are now centralized in the GameSettings class.

Unnecessary variables and confusing variable names: The Playing class had unnecessary variables, namely the enemyX and enemyY variables. We moved the initializations into the initClasses where the Position in the enemy was instantiated (commit SHA: 2c39212). Furthermore, we renamed the variables in the TileManager class in the draw method to prevent any confusion around naming.

Confusing class hierarchy: We also noticed that we had the Tile and TileManager class stored in the StaticEntity package which we found created a confusing class hierarchy because they had no relation to the StaticEntity package. To resolve this, we moved the two classes into the Helpers package which better reflects their purpose and relationship with other classes. This restructuring of the package hierarchy made our code more organized and intuitive to navigate, as classes are now grouped based on their functionality rather than their original package name (commit SHA: 5ef2e6a).

Dead code: Additionally, our project had dead code in that we were never using the BoardData class. We had initially planned from our UML diagram to use the BoardData class to store the map, but we found that the class would make our code unnecessarily complex and as a result, we removed the class from our workspace (commit SHA: 0f7bc36).

Badly structured project: In Phase 2, we found that our project was poorly structured (commit SHA: 13d05a8), in the StaticEntity package to load our map tiles we were storing the code and the tile sprites both in the same place which made the StaticEntity folder extremely crowded. We resolved this by separating src/main into src/main/java and src/main/resources which significantly improved our file management (commit SHA: b4ddb15). Also, we created two new packages, Animation and PathFinding. The Animation package contains the classes Animation and ImageUtils to handle all entity animations/images. The PathFinding package contains the classes TileNode and PathFinding to handle the A* algorithm used for the enemy (commit SHA: 51ebc77).

Unnecessary switch/case statements: The AnimationConstants class held unnecessary switch cases which returned an int value for the length of the animation array based on the movement a MoveableEntity is doing. We realized that it was returning the same value for every single movement, so we created a new attribute for the MoveableEntity class that holds the int

value and we removed the switch cases from AnimationConstants (commit SHA: d3e072a, 925e7d7, 488379c). In the Game class, we removed unnecessary switch statements in the update and render method because they were unused and as a result, this change also removed the corresponding enum constants in the Gamestate class (commit SHA: 319cc99),reducing complexity and making the code easier to maintain.

High coupling, unused and confusing variables: The Player class had an unused attribute, scoreObject, which was removed from the class and its constructor. This also made for looser coupling, as the Player depended on having a scoreObject to create a new Player instance. There were also confusing variable names such as animations for the sprite image array and animation for the actual Animation object, we adjusted these names to be more understandable. (commit SHA: 241c059)

Code duplication: For all classes that used image animations (Player, Enemy, Trap), we replaced duplicated code such as loadAnimation, updateAnimationTick, and loadImage with a new class Animations (commit SHA: 311f30d, bc6c41b, 01d4cfa). Also, the Menu class had repetitive code in the draw method for displaying text on the game menu which we found to be poorly structured. To resolve this, we created two new methods: drawShadowedString and drawOption that simplified our code (commit SHA: 2fa6d1e).

Lack of documentation: We also added more Javadocs to methods in various classes that we realized we had forgotten during the implementation phase. The addition of documentation enhances the readability and maintainability of our code, making it easier to add features and debug. (commit SHA: 0c1c357, b8b3b93, 9144ad3)