Group 11 report

Since our group was a group of 3, Saba said it would be ok to do the assignment together, but we would have to find more than 8 code smells (Around 10).

Nicholas Lagasse was assigned smell 1-6, 12-13
Felmer Macadangdang was assigned smell 7-8
Ekansh Chawla was assigned smell 9-11

Identified Smells:

1. BaseThread.java - Refactor sleep code into a function sleep().

There were two sections of identical code for sleeping in the run() function in thread (line 49 and 61), we put them both into a sleep() function.

2. EnemayAI.java - Duplicate enemy movement code in enemy and enemyAI, remove file.

There was a redundant class , EnemyAI, which could be easily implemented in Enemy.java. We put the code into Enemy.java to increase encapsulation.

3. EnemyGenerator.java - remove file

We had another redundant class ‘EnemyGenerator.java’, as it was hard-coded to only generate samurai. We could instead do ‘new Samurai()’ or ‘new Hole()’ wherever needed. Having the enemy generator added unnecessary code on top of doing new Object().

4. Multiple hard coded values across all files

We created a master ‘Constants.java’ class to hold all hard coded data values such as time and damage with getters and setters to increase coupling.

5. GameObjectFactory.java - replace string switch statement with integer values

We rewrote the switch cases to use unique integer IDs instead of strings so that typos would not be an issue. It would also allow for cases 10, 11, 12, and so on for example to be easily added logically later on.

6. GameObjectFactory.java - don’t pass object typeId to object constructor.

In each switch statement for object creation, we passed an integer Id to each constructor which did not make sense because each constructor was already unique, and extended gameObject. Instead, we hard coded these values into each gameObject constructor which would pass these values to the base gameObject constructor for increased readability.

7. GameState.java - code duplication for initiliazeItemsAndEnemies() and spawnEnemies().

We removed initialize Items and Enemies as it was redundant and used a pre existing method in place, this removed the duplicate code and made our code more readable.

8. GameState.java - movePlayer method takes long 4 parameters (up, down, left, right)
Passed in a keyhandler object instead of 4 booleans. This made the code more readable.

9. GameState.java - Lack of exception handling eg. ArrayIndexOutOfBoundsException when accessing gameBoard. Refactor to include array bounds check and use exception handling when out of bounds.

10. Refactor IsButtonClicked methods into one method

    We centralized the button click detection logic into a single method.
    This makes the code more maintainable and reduces redundancy.
    Repeated Functionality in mouseClicked():

11. GamePanel.java - put repeat functionality in mouseClicked() into one function
    We extracted the repeated code into a helper method.
    This simplifies the event handler and makes the code cleaner.

12. GameState.java - game crashes when enemy and player move into same square
In the updateEnemies() function we were initially reading and writing to one enemy array of enemies to update them, which caused conflicts. In the function itself we made a temporary array of enemies to update and return so that there would be no conflicts with other parts of the code editing the original array at the same time.

13. Renderer.java - render() function calling too many variables in function
Create drawSprite() in the renderer to put common functionality in a single function which increases readability.