Dungeon Crafter - Unit Testing and Inspection Summary

Group 1 - Michal Bochnak, Sean Martinelli, Alex Viznystya, Artur Wojcik

Project Description

Dungeon Crafter is an online role playing mobile game for the Android operating system. The object of the game is complete adventures to level up your character. To begin an adventure, players enter the find group lobby in order to pair-up with three additional players to form a team. Each adventure consists of one hundred levels to be completed by a team. Each level contains multiple enemy characters that the team must defeat in order to move on. Players each take a turns moving their character and attacking an enemy if in range. The level of difficulty of each adventure increases as your character gains experiences and progresses.

Document Description

This document outlines the results of the inspection and unit testing performed on this application. The inspection of the methods in section II of this document was performed by three of the four team members that did not write that particular piece of code. The team members agreed upon a check list to confirm the validity of the code. Each method that was tested in the unit testing section was tested by a team member that did not write that particular piece of code. The team members identified the equivalence classes for the input of each method, and developed tests for each of the identified equivalence classes.

Test Results

public int extractRace(JSONObject obj)

This method extracts the characters race from the JSONObject sent from the server. The race is returned as an integer representation of the race. Inspection of this method revealed the need for comments to improve code readability. Unit testing did not reveal any bugs. All equivalence classes were handled correctly by the method.

private void processResponse(JSONObject response)

This method extracts all of the necessary information send in the JSONObject from the server. Inspection of this method revealed that some variables should have more meaningful names in order to improve code readability. Unit testing did not reveal any bugs. All equivalence classes were handled correctly by the method.

public class CharacterFactory

This class creates and returns Character objects that are used during gameplay. The Character objects returned can be either player characters or enemy characters. Inspection of this class revealed the need for comments to improve readability. It also revealed the need for a default

case in the switch statement. Unit testing did not reveal any bugs. All equivalence classes were handled correctly by the method.

synchronized private void updateGrid()

This method updates the view of the game grid for the user to show where characters and enemies are located. Inspection of this class revealed the need to place the most common case tested in the if-else chain first to allow for better code efficiency. Unit testing was not performed on this piece of code.

private boolean validateNewPosition(Position newPosition)

This method checks the specified Position to see if it is valid. It is valid if it is within the bounds of the grid and no other player is already occupying the position. If the position is valid, true is returned. If it is not, false is returned. Inspection did not reveal any changes that needed to be made to this piece of code. Unit testing revealed a bug that resulted from not addressing null being passed to the method. The code was corrected and was tested again. The second test illustrated that the problem was solved.

gamePlay.php

This server side script is responsible for processing the JSONObjects sent by the clients during gameplay. Upon the processing of an object, the database is updated to reflect the new state of the game. The new game state is then pushed to the other clients participating in the gameplay. Inspection revealed the need for comments and to remove unused variables. Unit testing did not reveal any bugs. All equivalence classes were handled correctly by the script.

private int rollDice()

This method rolls the dice by generating a random number between 1 and 6. The view corresponding to the number of moves remaining is updated to show the generated number and player is marked as having rolled the dice. Inspection revealed the need for comments to improve code readability. Unit testing did not reveal any bugs. All equivalence classes were handled correctly by the method.

private int determineDamageToDeal(Character character)

This method calculates and returns the amount of damage that is dealt by a character when attacking. This number is calculated based on the strength of the character passed to the method. Inspection was not performed on this piece of code. Unit testing revealed a bug that resulted from not addressing null being passed to the method. The code was corrected and was tested again. The second test illustrated that the problem was solved.

Summary

In summary, the results of the inspection revealed a number of areas with potential to increase efficiency. In addition, a number of styling inconsistencies throughout the program were apparent. The correction of the inconsistencies allowed for greater readability of the code. With regards to the unit testing, we were successful in uncovering two bugs. These bugs resulted from not addressing null being passed to the methods.