CPTS 422 Software Testing  
Final Project Report

By:

Nathanael Ostheller

Darin Hardie

William Walker

# Table Of Contents

|  |  |
| --- | --- |
| Requirements Specification | 3 |
| Use Case Modeling | 5 |
| Design Specification | 8 |
| Discovered Bugs | 8 |
| Testing Plans | 9 |
| Call Graph | 9 |
| Integration Test Justifications | 9 |
| Profiling Results |  |
| Individual Contributions |  |
| Appendix 1: Test Case Markdowns |  |

# Requirements Specification

## Project Description

This project is a standalone desktop application developed using JavaFX and Java to create an application for tracking required information for tabletop game combat in an automated and organized manner.

## Requirements

Graphic JavaFX Interface

|  |  |
| --- | --- |
| Description | The application must have a graphics interface created using JavaFX to display information to the user. |
| Source | Requirements Provided by Team |
| Priority | Priority Level 0: Essential Feature |

Creature Addition and Removal

|  |  |
| --- | --- |
| Description | The application needs to be able to add creatures to the creature field and remove creatures from the creature field. |
| Source | Requirements Provided by Team |
| Priority | Priority Level 0: Essential Feature |

Creature Identification

|  |  |
| --- | --- |
| Description | The application needs to be able assign names and images to creatures for the identification of them on the creature field. Creatures should also be outlined with a portrait depicting them as allied, enemy, or neutral. |
| Source | Requirements Provided by Team |
| Priority | Priority Level 0: Essential Feature |

Health Tracking

|  |  |
| --- | --- |
| Description | The application needs to be able to assign a hit point value to each creature and maintain this hit point calculation through damage or healing done to the creature. |
| Source | Requirements Provided by Team |
| Priority | Priority Level 0: Essential Feature |

Initiative Tracking

|  |  |
| --- | --- |
| Description | The application must assign turn order initiative values to each creature and sort them accordingly on the creature field. |
| Source | Requirements Provided by Team |
| Priority | Priority Level 0: Essential Feature |

Bonus Health System

|  |  |
| --- | --- |
| Description | The application contains a system for assigning temporary HP according to the standards used in 5th Edition Dungeons and Dragons. Temporary HP may only come from one source, adding new temporary HP removes existing temporary HP. Damage taken is taken from temporary HP before normal HP. |
| Source | Requirements Provided by Team , D&D 5e |
| Priority | Priority Level 1: Desired Functionality |

Condition Tracking

|  |  |
| --- | --- |
| Description | The application can assign conditions to creatures from a list of pre-existing conditions. These conditions are tracked in a list in the creature information. |
| Source | Requirements Provided by Team |
| Priority | Priority Level 1: Desired Functionality |

Save and Load Functionality

|  |  |
| --- | --- |
| Description | The application can save existing combats of creatures and load combats of creatures from saved files. |
| Source | Requirements Provided by Team |
| Priority | Priority Level 2: Stretch Goals |

# Use Case Modeling

A diagram of a person

Description automatically generated

Add Creature

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Create a Creature Representation |
| Preconditions: | * None |
| Scenarios: | * Adding an allied creature * Adding a neutral creature * Adding an enemy creature |
| Exceptions: | * None |

Remove Creature

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Remove a Creature Representation |
| Preconditions: | * Creature Must Exist |
| Scenarios: | * Removing an allied creature * Removing a neutral creature * Removing an enemy creature |
| Exceptions: | * None |

Fill Out Creature Information

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Add Information to a Creature |
| Preconditions: | * Creature Must Not Exist, Creature Creation Dialog Must Be Open |
| Scenarios: | * Set Creature Health * Set Creature Initiative * Set Creature Name * Set Creature Image * Set Creature Alignment * Set Creature Temporary HP |
| Exceptions: | * Temporary HP can be changed on created creatures |

Damage Creature

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Reduce Creature’s Health |
| Preconditions: | * Creature Exists |
| Scenarios: | * Damage Creature without Temporary HP * Damage Creature with Temporary HP * Damage Creature with 0 HP |
| Exceptions: | * None |

Heal Creature

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Increase Creature Health |
| Preconditions: | * Creature Exists |
| Scenarios: | * Heal Creature with Full Health * Heal Creature Missing Less Health Than Healed * Heal Creature Missing More Health Than Healed |
| Exceptions: | * None |

Give Creature Temporary HP

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Give Creature Temporary HP |
| Preconditions: | * Creature Exists |
| Scenarios: | * Add Temp HP to Creature with Temp HP * Add Temp HP to Creature without Temp HP |
| Exceptions: | * None |

Add Creature Condition

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Add Condition to Creature |
| Preconditions: | * Creature Exists |
| Scenarios: | * Adding first condition * Adding additional condition |
| Exceptions: | * None |

Remove Creature Condition

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Remove Condition From Creature |
| Preconditions: | * Creature Exists, Creature Has Condition |
| Scenarios: | * Removing a Condition |
| Exceptions: | * None |

Save Combat

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Save Open Data |
| Preconditions: | * None |
| Scenarios: | * Saving Single Creature * Saving Multiple Creatures * Saving Empty Field |
| Exceptions: | * None |

Load Combat

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Load Previously Saved Data |
| Preconditions: | * Field is Empty |
| Scenarios: | * Loading Single Creature * Loading Multiple Creatures * Load Nothing |
| Exceptions: | * None |

View Existing Creatures

|  |  |
| --- | --- |
| Actors: | * General User |
| Goal: | * Have Creatures Displayed |
| Preconditions: | * A Creature Exists |
| Scenarios: | * Looking At Creature Field |
| Exceptions: | * None |

# Design Specification

A diagram of a data flow

Description automatically generated

# Discovered Bugs

1. Creature.removeHealth bypassed the Creature.getCurrentConditions function and used the one in the ConditionDao instead
   1. removeHealth\_with\_auto\_crits
   2. Fixed
2. CreatureDaoImpl had a section of untouched code in the advanceTurn function, added that code in for function optimization
   1. CreatureDao.advanceTurn
   2. Fixed
3. CreatureDaoImpl.groupCreatures would skip some creatures when grouping
   1. Pairwise Testing
   2. Fixed

# Testing Plans

## System Testing

For system testing we hope to focus on black box testing, and test things from the GUI and a user perspective, as that is something that we were unable to really do during our unit testing in this section.

## Integration Testing

For integration testing we intend to do neighborhood integration testing and integrate items with the classes directly surrounding it. This will allow us to test the systems that will actually be interacting in groups without needing to limit to pairs or having large amounts added at the same time.

## Performance Testing

As stated in class, due to us not having a web application, and instead developing a GUI application we do not have a clear way to performance test. We do intend to run tests regarding filling the application with large number of creatures to test performance.

# Call Graph

I recommend at least 300% zoom for viewing this graph. It should be high quality enough to read when zoomed in, if not, the image itself is in the documents folder of our GitHub and a markdown version can be found in the tests file of the source code.

A diagram of a flowchart

Description automatically generated

# Integration Test Justifications

Advancing to the next round is very important for our application as it makes users able to track whose turn it is as well as how long their conditions will last for. The advance turn button has different functionality depending on if the creatures’ group is the first/middle, or last. We wanted to make sure that conditions, a member of the creature class, decremented correctly when advancing a turn.

We chose to pair sortByInitiative and groupCreatures in CreatureDaoImpl as group creatures makes up about half the functionality of sortByInitiative but the two aren’t used together in any unit testing. This ensures that sorting actually groups creatures as it is supposed to as groupCreatures is not called alone, only when the inventory is sorted or a creature is deleted.

# Profiling Results

# Milestone 1 Contributions

## Nathanael Ostheller

* Created Condition Enumerated Class
* Created Image Selection Code for GUI
  + Integrated Image Support into Creature Class
* Added Scroll Bar to Creature Manager
* Created Requirements Specification
* Created Component Diagram
* Code Review and Exporting

## Darin Hardie

* Setup GUI System
* GUI Integrations
* Creature Manager Application Class

## Will Walker

* Basic GUI Wireframes
* Creature Class Implementation
* Creature Class Functions
* Creature Dao and Implementation
* Save and Load Functionality

# Milestone 2 Contributions

## Nathanael Ostheller

* Added critical hit functionality in backend
* Updated GUI for allowing critical hits
* Updated damage functions to account for crits and autocrits
* Creature.addHealth Tests
* Creature.removeHealth Tests
* Creature.addBonusHealth Tests
* Creature.addCondition Test
* CreatureDao.advanceTurn Test
* CreatureDao.SaveCreatures Test
* CreatureDao.LoadCreatures Test

## Darin Hardie

* ConditionDaoImplTest
* Creature.removeCondition Tests
* Creature.decrementCondition Tests
* Creature.getName Tests
* Creature.getMaxHealth Tests
* Creature.getCurrentHealth Tests
* Creature.getInitative Tests
* Creature.getCurrentConditions Tests
* Creature.getAvailableConditions Tests

## Will Walker

* Updated condition objects
* Created condition durations
* Integrated new conditions into GUI
* Turn tracking implementation
* Creature grouping for turn tracking
* Advance Turn Button in GUI
* GUI rounds label
* Active turn borders
* Creature.getAvailableConditions Tests
* Fixed Creature.removeHealth Tests
* CreatureDao.createCreature Tests
* CreatureDao.deleteCreature Tests

# Milestone 3 Contributions

## Nathanael Ostheller

* GUI Code Restructuring Creating Following Functions from Existing Code
  + selectImageAction()
  + selectSaveFileAction()
  + selectLoadFileAction()
  + dialogSetResultLoad()
  + dialogSetResultSave()
  + addButtonAction()
  + bonusHealthButtonEvent()
  + conditionButtonEvent()
  + deleteConditionButtonEvent()
  + deleteButtonEvent()
  + damageButtonEvent()
  + healButtonEvent()
  + doDamage()
  + doHealing()
  + convertResultToPair()
  + ifBonusHealthPresent()
  + ifConditionPresent()
  + pairPresent()
  + nextTurnPressed()
* Call Graph
* Unit Test TMS Documentation
* SortByInitiative -> groupCreatures Pairwise Tests
* Final Report Structuring

## Darin Hardie

## William Walker

* AdvanceTurn - Empty groupedByTurnCreatures Test
* AdvanceTurn - Single Group with Condition Test
* AdvancedTurn - Multiple Groups, Middle Group Without Condition Test
* AdvancedTurn - Multiple Groups, Last Group With Condition Test