Chapter 1

TEST PLAN (TP) TEMPLATE

Version 1.1, April 2014

TEST PLAN: SCENARIO CLASS

FOR

Swords & Sorcery

Version 1
6 May 2014

Prepared for:
Clinton Jeffery

Prepared by:
Tyler Jaszkowiak
University of Idaho
Moscow, ID 83844-1010
CS383 TPD

RECORD OF CHANGES (Change History)

Change Number	Date completed	Location of change (e.g., page or figure #)	A M D	Brief description of change	Approved by (initials)	Date approved
1	6 May 2014	Document	A	Creation of Test Plan	TJ	6 May 2014

A - ADDED M - MODIFIED D – DELETED

[put program /system name here]

TABLE OF CONTENTS

Section Page

1 TEST PLAN IDENTIFIER

This test plan covers the Scenario class in the Swords & Sorcery project for Dr. Jeffery's Spring 2014 CS383 class. The Scenario class is part of the Rules and Game Play team's work.

2 REFERENCES

The Scenario class itself can be found at https://github.com/cjeffery/sworsorc/blob/master/src/utilities/sscharts/Scenario.jav
The automated unit tests for the class reside in https://github.com/cjeffery/sworsorc/blob/master/src/test/sschartstests/Scer
The versions refered to in this document shall be that from commit b86744e0a0d31de79eba0f9663298e7ba1bc7196.

3 INTRODUCTION

This test plan is to provide as much coverage as possible to the methods and data of the Scenario class by verifying the results of execution against expectations through a series of automated unit tests and a manual GUI test.

4 TEST ITEMS

These are things you intend to test within the scope of this test plan. Essentially, something you will test, a list of what is to be tested. This can be developed from the software application inventories as well as other sources of documentation and information.

The data read by the Scenario test will be read from one of the simple scenario configuration files and then verified against expectations through the class's accessor methods. These data items include

- The scenario's name
- Number of players
- Game length
- The blue sun's initial position
- The names of armies in the scenario
- The controlling players of these armies
- The setup order of these armies
- The movement order of these armies
- Names of nations within these armies
- Names of the neutrals in the scenario
- The provinces controlled by a nation
- The characters within a nation
- The units within a player nation
- The units within a neutral nation
- The races of both neutrals and player nations
- The reinforcement and replacement description strings
- Data related to where a neutral is leaning toward
- Whether or not a neutral accepts human sacrifice

Unfortunately, the most complex functionality of the Scenario class cannot be tested by automated unit tests. The unit pool populator requires a manual check.

5 SOFTWARE RISK ISSUES

Identify what software is to be tested and what the critical areas are, such as:

- Depends on Java's JSON reader and the programmer's understanding of it
- Complex data structures such as a map of maps
- Complex loops to iterate over data structures
- Poor documentation surrounding some of these iterators

6 FEATURES TO BE TESTED

This is a listing of what is to be tested from the USERS viewpoint of what the system does. This is not a technical description of the software, but a USERS view of the functions.

From the user's perspective, much of this data reading occurs under the hood. In all cases except for populating the unit pool, the Scenario class does not manipulate any pieces of the rest of the project. Other components read the data from the Scenario class. Therefore, while the solar configuration relies on a properly-working Scenario class, this is not apparent to the user because solar configuration is also handled by SolarConfig and HUDInitializer classes.

Therefore, the only visible component being tested by this plan is the placement of units and characters into the correct provinces of the map.

7 FEATURES NOT TO BE TESTED

This is a listing of what is NOT to be tested from both the Users viewpoint of what the system does and a configuration management/version control view. This is not a technical description of the software, but a USERS view of the functions.

All data is verified, but testing the Scenario class alone cannot ensure that it reaches the HUD successfully. Integration tests are required outside of this plan for information such as solar configuration, move order, setup order, game length, and diplomacy.

8 APPROACH

The automated unit tests rely on the JUnit tool. Code coverate metrics can also be acquired regarding the automated tests by running the NetBeans Java Code Coverage plugin on the project. This will verify all data read in.

Because units cannot be placed into the unit pool if the map is not loaded, automated testing of unit placement is difficult. Therefore, the placement of units and characters in the correct provinces will have to be verified manually by counting units of each type in one of the scenario's nations when running the game.

9 ITEM PASS/FAIL CRITERIA

The automated tests must all pass and result in 100each of the class's accessor methods. One of the nations in the scenario must have the correct number and types of units loaded into the provinces it controls.

10 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

If any of the accessor method tests fails, the test must be suspended as this is likely due to a change in the class's interface or the scenario itself. Any such bug must be resolved before it makes sense to find others.

11 TEST DELIVERABLES

- Test plan document
- JUnit test cases
- A report of passes/failures from JUnit
- A coverage report from JaCoCo

12 REMAINING TEST TASKS

All other classes in the project require some level of testing as well. In particular, the HUD and SolarConfig related classes directly affect the performance of Scenario.

13 ENVIRONMENTAL NEEDS

No specific requirements exist.

14 STAFFING AND TRAINING NEEDS

Understading of NetBeans, Java, JUnit, and JaCoCo.

15 RESPONSIBILITIES

Anyone may perform these tests. However, as the author of this particular class, it would probably be most efficent to have Tyler Jaszkowiak run them.

16 SCHEDULE

Tests will be conducted once before the semester ends.

17 PLANNING RISKS AND CONTINGENCIES

[Insert text here.]

18 APPROVALS

Dr. Jeffery has the final word. However, I imagine that the team generating the test documentation will decide which tests can be approved and whether to continue.

19 GLOSSARY

JaCoCo: The Java Code Coverage plugin for NetBeans