UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Testing Document

Project name: Arcane Arcade

Client: Tony vd Linden

Team name: Terabites

Team members

NG Maluleke	D Mulugisi	C Nel	LE Tom
13229908	13071603	14029368	13325095

July 29, 2016

Contents

1 Introduction

The project is called *Arcane Arcade*, which references the esoteric language users will have to use, as well as the gamification approach to try and make it as fun as possible.

2 Background

The client needs a way to test potential new employees without running the risk of tests being leaked and without having to manually create complicated in-house tests. Through this system the client can test potential new employees via a custom programming language, an esoteric language, while keeping the tests interesting through the use of gamification principles. Testees' performance and use of help can also help determine the ideal employment position.

3 Testing

3.1 Introduction

Unit testing is a crucial part of a continual software development approach and is necessary to ensure that all components properly work before integrating them into the system.

3.2 Approach

Most of the unit testing is to test the compiler component of the system. The compiler consists of a lexer, parser, and visitor. The lexer and parser are automatically generated by ANTLR4 using a specified grammar file which specifies the lexer and syntax rules for the language. The visitor is manually implemented and is in charge of walking the parse tree in the correct order and executing the appropriate commands on each node.

Testing is done through JUnit test files and Maven automatically running the tests when deploying the project or running the *test* command through Maven.

3.3 Tests conducted

The lexer and parser are tested together by feeding a list with a syntatically correct and wrong statements in the implemented language. The lexer then tokenizes the input whereafter the parser parses the token stream and builds a parse tree. It is then tested that the parser reports the correct symantic errors.

The visitor is tested by having it run through a list of statements. The visitor is implemented in such a way that it holds a list of values that need to be printed. The test then simply looks at this list in the visitor object and compares it with a list of expected results.