Benoit Anctil-Robitaille, Benedek Horváth, Hunton Bowland

Cloud Databases Praktikum:

Milestone 2 test report

In order to test the functionality of our project at unit level we have compiled a library of tests including both our own and those prescribed by the course material. This report is formatted according to the individual test classes extending the TestCase class. The target, justification, and method of testing in each case will be given high level elaboration in their respective paragraphs.

Prescribed Unit tests (testing.ConnectionTest, testing.InteractionTest)
These tests have been slightly modified from their original form in the provided stub code to accommodate the naming convention of the project. Note that the assertions of the tests in these classes remain nearly the same as that of the originals.

ArgumentsValidatorTest
This class contains tests which examine the functionality of the ArgumentsValidator class and the exceptions it can throw. The ArgumentsValidator class is responsible for checking the validity of actual user input against expected user input. It's important to test the functionality of a class such as this to ensure that user input will be properly verified by the program during deployment.
The ArgumentsValidatorTest class contains tests which stress each function of the ArgumentsValidator class by testing validation of correct input and invalidation of incorrect input for all types of input.
Dummy input is provided by passing constant strings into constructors of input classes and passing that in turn to an ArgumentsValidator instance. *@Test* annotations are used to denote each test.
When testing invalid input a try-catch-finally block is used in conjunction with *assertTrue()* and in two cases *assertThat()* to verify that an exception has been thrown and that it is the expected type of exception.
When testing valid input the validation function of the validator is called with dummy input and allowed to fall through quietly.

UserInputParserTest
This class is responsible for testing the functionality of the UserInputParser class. This class' task is parsing user input from the input stream in order to establish which command the user wants to execute and to identify the arguments provided (if any). Testing this class is important in order to make certain that user input will be properly parsed by the client while it is running.
The test class UserInputParserTest implements its tests using three private methods that verify a successful parse using calls to the *assertThat()* function. These three methods are called inside three public *@Test* annotated methods which provide a series of different input types to the parser based on constants contained inside the class.

CommunicationServiceTest

A note should be made of this test case as it is still present in the project. These tests were not used to test this release and do not run properly. They have been kept for possible future use and reference.