TEST REPORT FOR BASIC STORAGE SERVER

PURPOSE OF DOCUMENT:

The purpose of this document is to identify the different types of tests (JUnit) performed, specify their intended behavior and document their expected results.

CACHE TESTS (SELF CREATED TESTS):

Cache tests were created to test the three cache eviction strategies, LRU, LFU and FIFO. For each test, the cache object is created, which is then filled with key value pairs and then randomly accessed. Then another item is added and one already added entry is evicted on the basis of the strategy. The tests check whether the resulting key value pairs after eviction are equal to the expected key value pairs after eviction.

ID	01
Title	testFIFOCacheRemoval
Summary	Tests FIFO eviction strategy after adding and accessing elements to cache.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Size of cache should be specified.
Steps	• FIFOCache object is created. • Data is randomly added to the cache to fill it up. • Data is randomly accessed from cache. • One more data entry is added to cache. • An expected set of keys after eviction is created depending on FIFO order. • Expected set of keys and obtained set of keys are compared.
Expected Results	Expected set of keys and obtained set of keys is the same.

ID	02
Title	testLRUCacheRemoval
Summary	Tests LRU eviction strategy after adding and accessing elements to cache.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Size of cache should be specified.
Steps	• LRUCache object is created. • Data is randomly added to the cache to fill it up. • Data is randomly accessed from cache. • One more data entry is added to cache. • An expected set of keys after eviction is created depending on LRU order.

	• Expected set of keys and obtained set of keys are compared.
Expected Results	Expected set of keys and obtained set of keys is the same.

ID	03
Title	testLFUCacheRemoval
Summary	Tests LFU eviction strategy after adding and accessing elements to cache.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Size of cache should be specified.
Steps	• LFUCache object is created. • Data is randomly added to the cache to fill it up. • Data is randomly accessed from cache. • One more data entry is added to cache. • An expected set of keys after eviction is created depending on LFU order. • Expected set of keys and obtained set of keys are compared.
Expected Results	Expected set of keys and obtained set of keys is the same.

CONNECTION TESTS:

Connection tests are used to check the connection to the server by passing correct and incorrect arguments for both address and port and seeing whether the exception value is the right one for each case or not.

ID	04
Title	testConnectionSuccess
Summary	Tests the success of connection to server by passing correct values for host and port.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Server should be running.
Steps	• Exception object is created. • Client object is created with correct host and port. • Attempt to connect client is made with exception caught if any.
Expected Results	Exception's value remains null as at the start. I.e. successful connection.

ID	05
Title	testUnknownHost
Summary	Tests the failure of connection by passing unknown host as an argument.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Server should be running.
Steps	• Exception object is created. • Client object is created with unknown host and correct port. • Attempt to connect client is made with exception caught if any.
Expected Results	Unknown host exception is caught and asserted as true.

ID	06
Title	testIllegalPort
Summary	Tests the failure of connection by passing an illegal port which cannot be taken as a correct argument.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Server should be running.
Steps	• Exception object is created. • Client object is created with correct host but illegal (out of range) port. • Attempt to connect client is made with exception caught if any.
Expected Results	Illegal argument exception is caught and asserted as true.

INTERACTION TESTS:

Interaction tests are used for testing the communication between client and server where different key value pairs are added and accessed under various conditions to check the response of the server.

ID	07
Title	testPut
Summary	Tests the interaction between client server by putting a key value pair.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present.

	• Server should be running. • Client object should be created.
Steps	• Strings for key and value are created. • Exception and response objects are created. • Put command with key value pair is sent and response is saved. • Exception is caught if any.
Expected Results	Exception is null and response status is equal to PUT_SUCCESS.

ID	08
Title	testPutDisconnected
Summary	Tests the interaction by putting a key value pair after disconnectiong from the server.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Server should be running. • Client object should be created.
Steps	• Client is disconnected from the server. • Strings for key and value are created. • Exception object is created. • Put command with key value pair is sent. • Exception is caught if any.
Expected Results	Exception is not null.

ID	09
Title	testUpdate
Summary	Tests the update of value for a particular key after already putting a value for the same key.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Server should be running. • Client object should be created.
Steps	• Strings for key and value and updated value are created. • Exception and response objects are created. • Put command with key value pair is sent. • Put command with updated value is sent and response is saved. • Exception is caught if any.

Expected Results	Exception is null, the response status is PUT_UPDATE and response value is equal to updated value.

ID	10
Title	testDelete
Summary	Tests the deletion of an entry if for an already present key, null value is added.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Server should be running. • Client object should be created.
Steps	• Strings for key and value are created. • Exception and response objects are created. • Put command with key value pair is sent. • Put command with null value is sent and response is saved. • Exception is caught if any.
Expected Results	Exception is null and response status is equal to DELETE_SUCCESS.

ID	11
Title	testGet
Summary	Tests that after an entry is added, on accessing the same entry, the correct value is provided.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Server should be running. • Client object should be created.
Steps	• Strings for key and value are created. • Exception and response objects are created. • Put command with key value pair is sent. • Get command for added key is sent and response is saved. • Exception is caught if any.
Expected Results	Exception is null and response value is equal to the value added before.

ID	12

Title	testGetUnsetValue
Summary	Tests the failure of getting the value of an unadded key to the storage server.
Pre-Conditions	• Jdk1.8 should be present. • JUnit lib should be present. • Server should be running. • Client object should be created.
Steps	• String for an unadded key is created. • Exception and response objects are created. • Get command for unadded key is sent and response is saved. • Exception is caught if any.
Expected Results	Exception is null and response status is GET_ERROR.