**Test cases description**

**KVMarshallerTest**

|  |  |
| --- | --- |
| KVMarshallerTest.testMarshall | * + testing the marshall logic for general values |
| KVMarshallerTest.testMarshallSpecialCharacters | * + testing marshall logic when the marshall values has special characters that are part of the message protocol |

**KVUnmarshallerTest**

|  |  |
| --- | --- |
| KVUnmarshallerTest.testUnmarshall | * + testing the unmarshall logic for general values |
| KVUnmarshallerTest.testUnmarshallScpecialCharacters1 | * + testing unmarshall logic when the unmarshall values has special characters that are part of the message protocol |
| KVUnmarshallerTest.testUnmarshallScpecialCharacters2 | * + testing unmarshall logic when the unmarshall values has special characters that are part of the message protocol |
| KVUnmarshallerTest.testUnmarshallScpecialCharacters3 | * + testing unmarshall logic when the unmarshall values has special characters that are part of the message protocol |
| KVUnmarshallerTest.testUnmarshallScpecialCharacters4 | * + testing unmarshall logic when the unmarshall values has special characters that are part of the message protocol |
| KVUnmarshallerTest.testUnmarshallScpecialCharacters5 | * + testing unmarshall logic when the unmarshall values has special characters that are part of the message protocol |
| KVUnmarshallerTest.testUnmarshallScpecialCharacters6 | * + testing unmarshall logic when the unmarshall values has special characters that are part of the message protocol |
| KVUnmarshallerTest.testInvalidMessage | * + testing unmarshall logic when message type is wrong |
| KVUnmarshallerTest.testEmptyMessage | * + testing unmarshall logic when the key value is empty |
| KVUnmarshallerTest.testMessageWithSpaces | * + testing unmarshall logic when the unmarshall values has spaces |

**FifoCacheTest**

|  |  |
| --- | --- |
| FifoCacheTest.addToCacheTest | * + test the logic of adding a key value pair to the cache |
| FifoCacheTest.getFromCacheForExistingKeyTest | * + test the logic when getting the value for an existing key |
| FifoCacheTest.getFromCacheForNonExistingKeyTest | * + test the logic when trying to get the value for non existing key |
| FifoCacheTest.addToCacheReplaceTest | * + test the cache replace logic when the cache is full |

**LFUCacheTest**

|  |  |
| --- | --- |
| LFUCacheTest.addToCacheTest | * + test the logic of adding a key value pair to the cache |
| LFUCacheTest.getFromCacheForExistingKeyTest | * + test the logic when getting the value for an existing key |
| LFUCacheTest.getFromCacheForNonExistingKeyTest | * + test the logic when trying to get the value for non existing key |
| LFUCacheTest.addToCacheReplaceTest | * + test the cache replace logic when the cache is full |

**LRUCacheTest**

|  |  |
| --- | --- |
| LRUCacheTest.addToCacheTest | * + test the logic of adding a key value pair to the cache |
| LRUCacheTest.getFromCacheForExistingKeyTest | * + test the logic when getting the value for an existing key |
| LRUCacheTest.getFromCacheForNonExistingKeyTest | * + test the logic when trying to get the value for non existing key |
| LRUCacheTest.addToCacheReplaceLogicTest | * + test the cache replace logic when the cache is full |

**SimpleKeyValueStoreTest**

|  |  |
| --- | --- |
| SimpleKeyValueStoreTest.shouldGetValue | * + test get value logic from the database for a given key |
| SimpleKeyValueStoreTest.shouldGetMultipleValues | * + test multiple consecutive retrievals for keys works correctly |
| SimpleKeyValueStoreTest.shouldReturnCorrectValuesForHasKey | * + test whether the hasKey() method works properly |
| SimpleKeyValueStoreTest.shouldThrowKeyNotFound | * + test the proper exception is thrown when trying to retrieve non existent key |
| SimpleKeyValueStoreTest.shouldWriteValues | * + test the logic of writing to the database works properly for a given key value pair |
| SimpleKeyValueStoreTest.readWrites | * + test that the written key value pairs can be read after |
| SimpleKeyValueStoreTest.readMultipleWrites | * + test that the written multiple key value pairs can be read after |

**RandomAccessKeyValueStoreTest**

|  |  |
| --- | --- |
| RandomAccessKeyValueStoreTest.shouldGetValue | * + test get value logic from the database for a given key |
| RandomAccessKeyValueStoreTest.shouldGetMultipleValues | * + test multiple consecutive retrievals for keys works correctly |
| RandomAccessKeyValueStoreTest.shouldReturnCorrectValuesForHasKey | * + test whether the hasKey() method works properly |
| RandomAccessKeyValueStoreTest.shouldThrowKeyNotFound | * + test the proper exception is thrown when trying to retrieve non existent key |
| RandomAccessKeyValueStoreTest.shouldWriteFile | * + test the logic of writing to the database works properly for a given key value pair |
| RandomAccessKeyValueStoreTest.shouldDeleteValue | * + test the logic of deleting a key value pair works properly |
| RandomAccessKeyValueStoreTest.shouldDeleteValueAndReadOthers | * + test that when there are multiple key value pairs in the database and when delete one, others can be read without error |
| RandomAccessKeyValueStoreTest.shouldUpdateValue | * + test that the value is updated properly for an existing key |