# Introduction

## Background

This document is a test specification for MemCache project based on the Memcache software requirement specification [1].

## Purpose

The purpose of this document is to describe test strategy applied to the project and to list the test-cases.

# Testing strategy

All tests used for the project shall be created by means of Eiffel Studio embedded tool with respect to the requirements specification. Tests shall be stored in separated cluster in order to prevent their delivery with the application code.

# Test cases

* 1. Test cases summary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TC #** | **TC title** | **TC description** | **FR under test** | **FR title** |
| TC1 | StoreData | Storing <key,value> tuple | FR1 | Storing data in the cache |
| TC2 | StoreDuplicatedKey | Storing with duplicated key |
| TC3 | StoreEmptyValue | Storing with empty key or value |
| TC4 | GetData | Get data with the correct key | FR2 | Getting data from the cache |
| TC5 | GetByFakeKey | Get data with the key that doesn’t exist |
| TC6 | DeleteData | Delete data with the correct key | FR3 | Deleting elements from the cache by user request |
| TC7 | DeleteByFakeKey | Delete data with the key that doesn’t exist |
| TC8 | DeleteAllData | Delete all the data |
| TC9 | CheckAllDeleted | Check if data was cleaned after program termination | FR5 | Deleting elements from the cache automatically |
| TC10 | CheckLifo | Check deleting by lifo strategy |
| TC11 | CheckFifo | Check deleting by fifo strategy |
| TC12 | CheckLessUsed | Check deleting by less-used strategy |
| TC13 | SearchData | Search by stored value | FR6 | Searching for stored data by the value |
| TC14 | CheckByTime | Check automation deleting after storing time elapsing | FR7 | Parametrizing time of data storing |
| TC15 | CheckBySize | Check automation deleting after cache size overloading |
| TC16 | CheckKey | Check if the key exist | FR8 | Key existence checking |
| TC17 | InitializeDefault | Check default values initialization | FR9 | Cache size and preemption strategy initialization |

* 1. Test cases detailed description

|  |  |
| --- | --- |
| **TC id** | TC1 |
| **TC title** | StoreData |
| **TC description** | Storing <key,value> tuple |
| **Precondition** | Memcache doesn’t contain stored data |
| **Initial values** | key: “key”, value: “value” |
| **Steps** | 1. Memcache.store(key,value) |
| **Expected result** | Data with predefined key and value is stored in the library |

|  |  |
| --- | --- |
| **TC id** | TC2 |
| **TC title** | StoreDuplicatedKey |
| **TC description** | Storing with duplicated key |
| **Precondition** | TC1 |
| **Initial values** | key: “key”, value: “value” |
| **Steps** | 1. Memcache.store(key,value) |
| **Expected result** | Data is not duplicated.  Message “data with the same key exists” appears. (optional) |

|  |  |
| --- | --- |
| **TC id** | TC3 |
| **TC title** | StoreEmptyValue |
| **TC description** | Storing with empty key or value |
| **Precondition** | - |
| **Initial values** | key1: “”, value1: “value”  key2: “key”, value2: “” |
| **Steps** | 1. Memcache.store(key1,value1) 2. Memcache.store(key2,value2) |
| **Expected result** | No new data is stored.  Message “Can’t store data with empty key/value” appears. (optional) |

|  |  |
| --- | --- |
| **TC id** | TC4 |
| **TC title** | GetData |
| **TC description** | Get data with the correct key |
| **Precondition** | TC1 |
| **Initial values** | key: “key” |
| **Steps** | 1. Memcache.retrieve(key) |
| **Expected result** | Corresponding value (“value”) is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC5 |
| **TC title** | GetByFakeKey |
| **TC description** | Get data with the key that doesn’t exist |
| **Precondition** | No key (“key”) is stored |
| **Initial values** | key: “key” |
| **Steps** | 1. Memcache.retrieve(key) |
| **Expected result** | No value is obtained.  Message “Nothing was found” appears. (optional) |

|  |  |
| --- | --- |
| **TC id** | TC6 |
| **TC title** | DeleteData |
| **TC description** | Delete data with the correct key |
| **Precondition** | TC1 |
| **Initial values** | key: “key”, value: “value” |
| **Steps** | 1. Memcache.delete(key) 2. Memcache.retrieve(key) |
| **Expected result** | No value is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC7 |
| **TC title** | DeleteByFakeKey |
| **TC description** | Delete data with the key that doesn’t exist |
| **Precondition** | No key (“key”) is stored |
| **Initial values** | key: “key” |
| **Steps** | 1. Memcache.delete(key) |
| **Expected result** | No data is deleted. |

|  |  |
| --- | --- |
| **TC id** | TC8 |
| **TC title** | DeleteAllData |
| **TC description** | Delete all the data |
| **Precondition** | No keys (“key1”, “key2”) is stored |
| **Initial values** | key1: “key1”, value1: “value1”, key2: “key2”, value2: “value2” |
| **Steps** | 1. Memcache.store(key1, value1) 2. Memcache.store(key2, value2) 3. Memcache.delete() 4. Memcache.retrieve(key1) 5. Memcache.retrieve(key2) |
| **Expected result** | No value is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC9 |
| **TC title** | CheckAllDeleted |
| **TC description** | Check if data was cleaned after program termination |
| **Precondition** | No key (“key1”) is stored |
| **Initial values** | key: “key”, value: “value” |
| **Steps** | 1. Memcache.store(key, value) 2. Exit program 3. Memcache.retrieve(key) |
| **Expected result** | No value is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC10 |
| **TC title** | CheckLifo |
| **TC description** | Check deleting by lifo strategy |
| **Precondition** | No keys (“key1”, “key2”) is stored |
| **Initial values** | key1: “key1”, value1: “value1”, key2: “key2”, value2: “value2”,  strategy: “lifo”, time: 1, capacity: 1 |
| **Steps** | 1. Memcache.make(strategy, capacity, time) 2. Memcache.store(key1, value1) 3. sleep (1 minute) 4. Memcache.store(key2, value2) 5. sleep (1 minute) 6. Memcache.retrieve(key2) |
| **Expected result** | No value is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC11 |
| **TC title** | CheckFifo |
| **TC description** | Check deleting by fifo strategy |
| **Precondition** | No keys (“key1”, “key2”) is stored |
| **Initial values** | key1: “key1”, value1: “value1”, key2: “key2”, value2: “value2”,  strategy: “fifo”, time: 1, capacity: 1 |
| **Steps** | 1. Memcache.make(strategy, capacity, time) 2. Memcache.store(key1, value1) 3. sleep (1 minute) 4. Memcache.store(key2, value2) 5. sleep (1 minute) 6. Memcache.retrieve(key2) |
| **Expected result** | No value is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC12 |
| **TC title** | CheckLessUsed |
| **TC description** | Check deleting by less-used strategy |
| **Precondition** | No keys (“key1”, “key2”) is stored |
| **Initial values** | key1: “key1”, value1: “value1”, key2: “key2”, value2: “value2”,  strategy: “less\_used”, time: 1, capacity: 1 |
| **Steps** | 1. Memcache.make(strategy, capacity, time) 2. Memcache.store(key1, value1) 3. Memcache.store(key2, value2) 4. Memcache.retrieve(key1) 5. sleep (1 minute) 6. Memcache.retrieve(key2) |
| **Expected result** | No value is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC13 |
| **TC title** | SearchData |
| **TC description** | Search by stored value |
| **Precondition** | TC1 |
| **Initial values** | key: “key”, value: “value” |
| **Steps** | 1. Memcache.search(value) |
| **Expected result** | Data with corresponding key (“key”) is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC14 |
| **TC title** | CheckByTime |
| **TC description** | Check automation deleting after storing time elapsing |
| **Precondition** | No keys (“key1”, “key2”) is stored |
| **Initial values** | key1: “key1”, value1: “value1”, key2: “key2”, value2: “value2”,  strategy: “less\_used”, time: 1, capacity: 1 |
| **Steps** | 1. Memcache.make(strategy, capacity, time) 2. Memcache.store(key1, value1) 3. Memcache.store(key2, value2) 4. Memcache.retrieve(key1) 5. sleep (1 minute) 6. Memcache.retrieve(key2) |
| **Expected result** | No value is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC15 |
| **TC title** | CheckBySize |
| **TC description** | Check automation deleting after cache size overloading |
| **Precondition** | No key (“key”) is stored |
| **Initial values** | value: “value”, key: “key”,  strategy: “fifo”, time: 1, capacity: 1 |
| **Steps** | 1. Memcache.make(strategy, capacity, time) 2. for (i=1,1<1000)Memcache.store(“key”+i, “value”+i) 3. Memcache.retrieve(key1) |
| **Expected result** | No value is obtained. |

|  |  |
| --- | --- |
| **TC id** | TC16 |
| **TC title** | CheckKey |
| **TC description** | Check if the key exist |
| **Precondition** | TC1 |
| **Initial values** | key: “key”, value: “value” |
| **Steps** | 1. Memcache.if\_exist(key) |
| **Expected result** | Positive result is retrieved |

|  |  |
| --- | --- |
| **TC id** | TC17 |
| **TC title** | InitializeDefault |
| **TC description** | Check default values initialization |
| **Precondition** | - |
| **Initial values** | key: “key”, value: “value” |
| **Steps** | 1. Memcache.make 2. Memcache.strategy 3. Memcache.capacity 4. Memcache.time |
| **Expected result** | Default values for strategy (“fifo”), capacity(100), time (30) are retrieved. |

# References

1. Memcache SRS <https://docs.google.com/document/d/1a_ePCO7vkiSV4vSEQ1ZnmRo_7LWERt9tIV5zbjTmLns/edit#>