**Lab 8: black box testing, unit**

Goal of this lab is to practice black box testing of small software modules with state. For each of the following modules define test cases applying equivalence classes partitioning, and boundary conditions.

Use the following structure to document the test cases, defining clearly the criteria, the conditions on the criteria (partition), the test cases per each partition.

This lab should be done individually (not in teams).

**Documentation structure**

Criteria

|  |  |
| --- | --- |
| Criterion id | description |
| Criterion 1 | C1 |
| Criterion 2 | C2 |
| … |  |

Predicates

|  |  |
| --- | --- |
|  | Predicate |
| Criterion1 | C1 == true |
|  | C1 == false |
| Criterion2 | C2 < 0 |
|  | C2 > 0 |
| … |  |

Boundaries

|  |  |
| --- | --- |
| Criterion | Boundary |
| C2 | C2 == 0 |
|  |  |

Equivalence classes and tests

|  |  |  |  |
| --- | --- | --- | --- |
| C1 | C2 | Valid invalid | Test case |
| True | < 0 |  | T1 = |
|  | > 0 |  | T2 =  T3B =  (B indicates boundary test case) |
| False | < 0 |  |  |
|  | > 0 |  |  |

**Exercise 1**

A queue of events in a simulation system receives events. Each event has a time tag.

It is possible to extract events from the queue, the extraction must return the event with lower time tag.

The queue discards events with negative or null time tag.

The queue must accept at most 100.000 events.

Events with the same time tag must be merged (i.e. the second received is discarded)

public class EventsQueue{

public void reset();

// cancels all events

public void push(int timeTag) throws InvalidTag, QueueOverflow

public int pop() throws EmptyQueue

}

EventsQueue eq = new EventsQueue();

eq.reset();

eq.push(10); eq.push(20); eq.push(5); push(5); push(-1); push(0)

eq.pop(); // returns and cancels 5

eq.pop(); // returns and cancels 10

eq.pop(); // returns and cancels 20

Criteria

|  |  |
| --- | --- |
| Criterion id | description |
| Criterion 1 | C1 |
| Criterion 2 | C2 |
| … |  |

Focus: **push()** function

**Exercise 2**

A retail support system manages an inventory of items. Each item has a descriptor and the number of available items.

public class Item { // descriptor of items in inventory

private String itemCode // aka bar code, unique identifier of item

private int availability; // number of items available

private String description; // description of item

private String name; // name of item

public Item(String itemCode,int quantity, String description, String name); // creates a new item

}

public class Inventory{

void addItem(Item i) throws ItemAlreadyExists // adds new descriptor

Item searchItem (String itemCode) throws ItemNotExists; // returns item with given code

int availabilityItem (String itemCode) throws ItemNotExists; // returns availability of item

void subtractItem (String itemCode) throws ItemNotExists, ItemNotAvailable;

// subtracts 1 to availability

void subtractQtyToItem(String itemCode, int qty\_to\_sub) throws ItemNotExists, ItemNotAvailable;

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ItemCode  Present (valid or not) or missing | Quantity  negative or positive | Description  Present (valid or not) or missing | Name  Present (valid or not) or missing |  |
| Present valid | Negative | Present | Present |  |
|  |  |  | missing |  |
|  |  | Missing | present |  |
|  |  |  | Missing |  |
|  | Positive | Present | Present |  |
|  |  |  | Missing |  |
|  |  | Missing | Present |  |
|  |  |  | Missing |  |
| Present invalid | negative | Present | Present |  |
|  |  |  | Missing |  |
|  |  | Missing | Present |  |
|  |  |  | Missing |  |
|  | positive | present | Present |  |
|  |  |  | Missing |  |
|  |  | Missing | Present |  |
|  |  |  | Missing |  |
| Missing | Negative | Positive |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| ItemCode  Present (valid or not) or missing | qty\_to\_sub  Negative or positive (< Quantity) |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Criteria

|  |  |
| --- | --- |
| Criterion id | description |
| Criterion 1 | sign of quantity |
| Criterion 2 | sign of qty\_to\_sub |
| Criterion 3 | Quantity >= qty\_to\_sub |

Predicates

|  |  |
| --- | --- |
|  | Predicate |
| Criterion1 | C1 > 0 |
|  | C1 < 0 |
| Criterion2 | C2 > 0 |
|  | C2 < 0 |
| … |  |

Boundaries

|  |  |
| --- | --- |
| Criterion | Boundary |
| C1 | C1 == 0 |
| C1 | C1 == maxint |
| C2 | C2 == 0 |
| C2 | C2 == maxint |

Equivalence classes and tests

|  |  |  |  |
| --- | --- | --- | --- |
| C1 | C2 | Valid invalid | Test case |
| >0 | >0 |  | T1 = |
|  | > 0 |  | T2 =  T3B =  (B indicates boundary test case) |
| False | < 0 |  |  |
|  | > 0 |  |  |