Programming 2

Testing a simple array-based Priority Queue

Student Number: w4s22

Student Name: Alex Farrell

**Part 1 – Testing a Priority Queue**

(Please Note: See ‘Part 2 – Testing an *array-based* Priority Queue’ for ‘Actual Outcome’)

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Number** | **Test Description** | **Predicted Outcome** | **Actual Outcome** |
| 1.1 | queueSize():  Testing whether the queuesize variable has been passed into the ArrayPriorityQueue method correctly. | The correct queue size of ‘3’ should be returned. |  |
| 1.2 | isEmpty():  Testing whether when the queue has not been enQueued any elements, the queue is empty. | The isEmpty() method should return ‘true’. |  |
| 1.3 | enQueue():  Testing whether an attempt at trying to add an element of another primitive data type, throws an exception. | enQueue(2.5) should cause a runtime exception explaining that there is source code that will not compile. |  |
| 1.4 | enQueue():  Testing that an attempt at trying to add an element of data type ‘int’ doesn’t throw an exception. | enQueue(5) should add the element ‘5’ to the queue. |  |
| 1.5 | isEmpty():  Testing that now the queue should contain elements, the queue is not empty. | The isEmpty() method should return ‘false’. |  |
| 1.6 | queueLength():  Testing that now the queue should contain elements, the queue length is not ‘0’. | The correct queue length of ‘1’ should be returned. |  |
| 1.7 | isFull():  Testing that because the queue length is smaller than the queue size, the queue is not full. | The isFull() method should return ‘false’. |  |
| 1.8 | getHead():  Testing that when the head of a queue which contains elements is requested, the correct head value is returned. | A head value of ‘5’ should be returned. |  |
| 1.9 | enQueue() & getHead():  Testing that when an element is added to the queue, the priority aspect of the queue, puts the elements in the correct places. | When an element of ‘3’ is added, the head of the queue should still be ‘5’. |  |
| When an element of ‘8’ is added, the head of the queue should now be ‘8’ |  |
| 1.10 | getHead():  Testing that when the getHead() method is called, the queue remains unchanged. | When the getHead() method is called again, the value of ‘8’ should once again be returned. |  |
| 1.11 | isFull():  Testing that now the queue length should be the same as the queue size, the queue is full. | The isFull() method should return ‘true’. |  |
| 1.12 | enQueue():  Testing that now the queue should be full, when an attempt is made to add an element, an exception is thrown. | A runtime exception should be thrown explaining that the queue is full and that no more elements can be added. |  |
| 1.13 | Serve():  Testing that when an attempt is made to serve an element from a queue containing data, an exception is not thrown. | The element at the head of the queue ‘8’ should be removed from the queue. |  |
| 1.14 | queueLength():  Testing that now an element has been removed from the queue, the queue length has decreased. | The correct queue length of ‘2’ should be returned. |  |
| 1.15 | Serve():  Testing that when an attempt is made to serve an empty queue, an exception is thrown. | A runtime exception should be thrown explaining that the queue is empty and that there are no elements to serve. |  |
| 1.16 | getHead():  Testing that when the head of a queue which is empty is requested, the value returned is null. | A head value of ‘0’ should be returned. |  |

**Part 2 – Testing an *array-based* Priority Queue**

(Please Note: Evidence in separate table below.)

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Number** | **Test Description** | **Predicted Outcome** | **Actual Outcome** |
| 1.1 | queueSize():  Testing whether the queuesize variable has been passed into the ArrayPriorityQueue method correctly. | The correct queue size of ‘3’ should be returned. | A queue size of ‘3’ was returned. |
| 1.2 | isEmpty():  Testing whether when the queue has not been enQueued any elements, the queue is empty. | The isEmpty() method should return ‘true’. | A boolean value of ‘true’ was returned indicating that the queue is empty. |
| 1.3 | enQueue():  Testing whether an attempt at trying to add an element of another primitive data type, throws an exception. | enQueue(2.5) should cause a runtime exception explaining that there is source code that will not compile. | There is a warning stating that the source code may not compile and runtime exception is thrown. |
| 1.4 | enQueue():  Testing that an attempt at trying to add an element of data type ‘int’ doesn’t throw an exception. | enQueue(5) should add the element ‘5’ to the queue. | The element ‘5’ was added to the queue. |
| 1.5 | isEmpty():  Testing that now the queue should contain elements, the queue is not empty. | The isEmpty() method should return ‘false’. | A boolean value of ‘false’ was returned indicating that the queue is not empty. |
| 1.6 | queueLength():  Testing that now the queue should contain elements, the queue length is not ‘0’. | The correct queue length of ‘1’ should be returned. | A queue length of ‘1’ was returned. |
| 1.7 | isFull():  Testing that because the queue length is smaller than the queue size, the queue is not full. | The isFull() method should return ‘false’. | A boolean value of ‘false’ was returned, indicating that the queue is not full. |
| 1.8 | getHead():  Testing that when the head of a queue which contains elements is requested, the correct head value is returned. | A head value of ‘5’ should be returned. | A head value of ‘5’ was returned. |
| 1.9 | enQueue() & getHead():  Testing that when an element is added to the queue, the priority aspect of the queue, puts the elements in the correct places. | When an element of ‘3’ is added, the head of the queue should still be ‘5’. | The element ‘3’ was added and the head of the queue was still ‘5’. |
| When an element of ‘8’ is added, the head of the queue should now be ‘8’ | The element ‘8’ was added and the head of the queue is now ‘8’. |
| 1.10 | getHead():  Testing that when the getHead() method is called, the queue remains unchanged. | When the getHead() method is called again, the value of ‘8’ should once again be returned. | The head value of ‘8’ was once again, returned. |
| 1.11 | isFull():  Testing that now the queue length should be the same as the queue size, the queue is full. | The isFull() method should return ‘true’. | The boolean value of ‘true’ was returned, indicating that the queue is full. |
| 1.12 | enQueue():  Testing that now the queue should be full, when an attempt is made to add an element, an exception is thrown. | A runtime exception should be thrown explaining that the queue is full and that no more elements can be added. | A runtime exception is thrown indicating that there is overflow and that the data was not enQueued. |
| 1.13 | Serve():  Testing that when an attempt is made to serve an element from a queue containing data, an exception is not thrown. | The element at the head of the queue ‘8’ should be removed from the queue. | The element ‘8’ was removed from the queue. |
| 1.14 | queueLength():  Testing that now an element has been removed from the queue, the queue length has decreased. | The correct queue length of ‘2’ should be returned. | A queue length of ‘2’ was returned. |
| 1.15 | Serve():  Testing that when an attempt is made to serve an empty queue, an exception is thrown. | A runtime exception should be thrown explaining that the queue is empty and that there are no elements to serve. | A runtime exception is thrown indicating that there is underflow and that no data can be served. |
| 1.16 | getHead():  Testing that when the head of a queue which is empty is requested, the value returned is null. | A head value of ‘0’ should be returned. | A head value of ‘3’ was returned.  This is due to cyclic buffering needing an element to point to. |
| 1.17 | (Added due to test 1.16)  When an element smaller than the apparent head is added, is this value overwritten? | A head value of ‘2’ should be returned. | A head value of ‘2’ was returned. |

**Test Evidence**

|  |  |
| --- | --- |
| **Test Number** | **Evidence (Code & Output)** |
| 1.1 |  |
|  |
| 1.2 |  |
|  |
| 1.3 |  |
|  |
| 1.4 |  |
|  |
| 1.5 |  |
|  |
| 1.6 |  |
|  |
| 1.7 |  |
|  |
| 1.8 |  |
|  |
| 1.9 |  |
|  |
| 1.10 |  |
|  |
| 1.11 |  |
|  |
| 1.12 |  |
|  |
| 1.13 |  |
|  |
| 1.14 |  |
|  |
| 1.15 |  |
|  |
| 1.16 |  |
|  |
| 1.17 |  |
|  |