```
    1: /**************************************************************************
***********/
    2: /* RingBuffer.cpp
          */
    3: /* Yoo Min Cha
   */
    4: /* RingBuffer
          */
    5: /* Professor Martin
   */
    6: /* 16 March 2014
   */
    7: /**************************************************************************
***********/
    8:
    9: #include <iostream>
   10: #include <exception>
   11: #include <stdexcept>
   12: #include "RingBuffer.hpp"
   13:
   14: using namespace std;
   15: using namespace sf;
   16:
   17: RingBuffer::RingBuffer(int capacity):
   18: ringBuff(capacity, 0), _first(0), _last(0), _capacity(capacity), _full(fa
lse)
   19: {
   20:   if(capacity < 1)
   21:     throw invalid_argument("Capacity must be larger than zero");
   22: }
   23: int RingBuffer::size()
   24: {
   25:   return _capacity;
   26: }
   27: void RingBuffer::empty()
   28: {
   29:   _first = 0;
   30:   _last = 0;
   31:   _full = false;
   32: }
   33: bool RingBuffer::isEmpty()
   34: {
   35:   return !_full && (_first == _last);
   36: }
   37: bool RingBuffer::isFull()
   38: {
   39:   return _full;
   40: }
   41: void RingBuffer::enqueue(Int16 x)
   42: {
   43:   if (this->isFull())
   44:     throw runtime_error("Ring Buffer is full!");
   45:   ringBuff[_last] = x;
   46:   ++_last;
   47:   if (_last == this->size()) _last = 0;
   48:   if (_last == _first) _full = true;
   49: }
   50: Int16 RingBuffer::dequeue()
   51: {
   52:   if (this->isEmpty())
   53:     throw runtime_error("Ring Buffer is empty!");
   54:   Int16 x = ringBuff[_first];
   55:   ++_first;
   56:   if (_first == this->size()) _first = 0;
   57:   if (_full) _full = false;
```

```
   58:   return x;
   59: }
   60: Int16 RingBuffer::peek()
   61: {
   62:   if(isEmpty())
   63:     throw runtime_error("Ring Buffer is empty!");
   64:   return ringBuff[_first];
   65: }
```