```python
from exceptions import Empty

class LinkedQueue:
  """FIFO queue implementation using a singly linked list for storage."""

  #-------------------------- nested _Node class --------------------------
  class _Node:
    """Lightweight, nonpublic class for storing a singly linked node."""
    __slots__ = '_element', '_next'         # streamline memory usage

    def __init__(self, element, next):
      self._element = element
      self._next = next

  #------------------------------- queue methods -------------------------------
  def __init__(self):
    """Create an empty queue."""
    self._head = None
    self._tail = None
    self._size = 0                          # number of queue elements

  def __len__(self):
    """Return the number of elements in the queue."""
    return self._size

  def is_empty(self):
    """Return True if the queue is empty."""
    return self._size == 0

  def first(self):
    """Return (but do not remove) the element at the front of the queue.

    Raise Empty exception if the queue is empty.
    """
    if self.is_empty():
      raise Empty('Queue is empty')
    return self._head._element              # front aligned with head of list

  def dequeue(self):
    """Remove and return the first element of the queue (i.e., FIFO).

    Raise Empty exception if the queue is empty.
    """
    if self.is_empty():
      raise Empty('Queue is empty')
    answer = self._head._element
    self._head = self._head._next
    self._size -= 1
    if self.is_empty():                     # special case as queue is empty
      self._tail = None                     # removed head had been the tail
    return answer

  def enqueue(self, e):
    """Add an element to the back of queue."""
    newest = self._Node(e, None)            # node will be new tail node
    if self.is_empty():
      self._head = newest                    # special case: previously empty
    else:
      self._tail._next = newest
    self._tail = newest                      # update reference to tail node
    self._size += 1
```