25.5 *Lab:* Heapsort

As we saw briefly in Chapter 19, a priority queue can be used for sorting. All you have to do is add all the items to a priority queue in any order, then remove them one by one. The items will be returned in ascending order. If the priority queue is implemented as a min-heap, this sorting algorithm will run in $O(n \log n)$ time.

It is possible to apply this algorithm to an array, without creating a separate priority queue. This efficient algorithm, proposed by J. Williams in 1964, is called *Heapsort*.

Suppose you have an array a and you need to sort its elements in ascending order. Heapsort is performed in two phases. In the first phase, the elements are rearranged to form a max-heap with the root (the largest element) in a[0]. This is accomplished by calling the reheapDown procedure for every node that has children, starting with the last one:

```
int n = a.length;
for (int i = n/2; i >= 1; i--)
  reheapDown(a, i, n);  // reheap down the subtree with the root
                        // in a[i-1]
```

In the second phase, the root element is swapped with the *n*-th element of the array, the size of the heap n is decremented by one, and the heap is repaired by applying the reheapDown procedure to the root:

```
while (n > 1)
{
  // swap a[0] with a[n-1]:
  double temp = a[0], a[0] = a[n-1], a[n-1] = temp;

  n--;
  reheapDown(a, 1, n);
}
```

As a lab exercise, write a class Heapsort with a method

```
public static void sort(double[] a)
```

and a private method reheapDown. Add your Heapsort to the *Benchmarks* program from Chapter 13 to compare it with the other sorting algorithms.