Alexandria University,

Faculty of engineering,

Data structures.

Assignment 1.

Name: Mohamed Mohamed Abdlhakem. (43)

Problem statement:

Heap implementation:

It is required to implement the following some basic procedures:

• The MAX-HEAPIFY procedure, which runs in O (log n) time, is the key to maintaining the max-heap property. Its input is a root node. When it is called, it assumes that the binary trees rooted to the left and right of the given node are max-heaps, but that the element at the root node might be smaller than its children, thus violating the max-heap property.

• The BUILD-MAX-HEAP procedure, which runs in linear time, produces a max-heap from an unordered input array.

• The HEAPSORT procedure, which runs in O (n log n) time, sorts an array in place.

• The MAX-HEAP-INSERT, and HEAP-REMOVE-MAX procedures, which run in O (log n) time, allow the heap data structure to implement a priority queue.

Sorting techniques:

• It is required to implement the heap sort algorithm as an application for binary heaps. It is advised to compare the running time of your implementation against:

– An O (n^2) sorting algorithm such as Selection Sort, Bubble Sort, or Insertion sort.

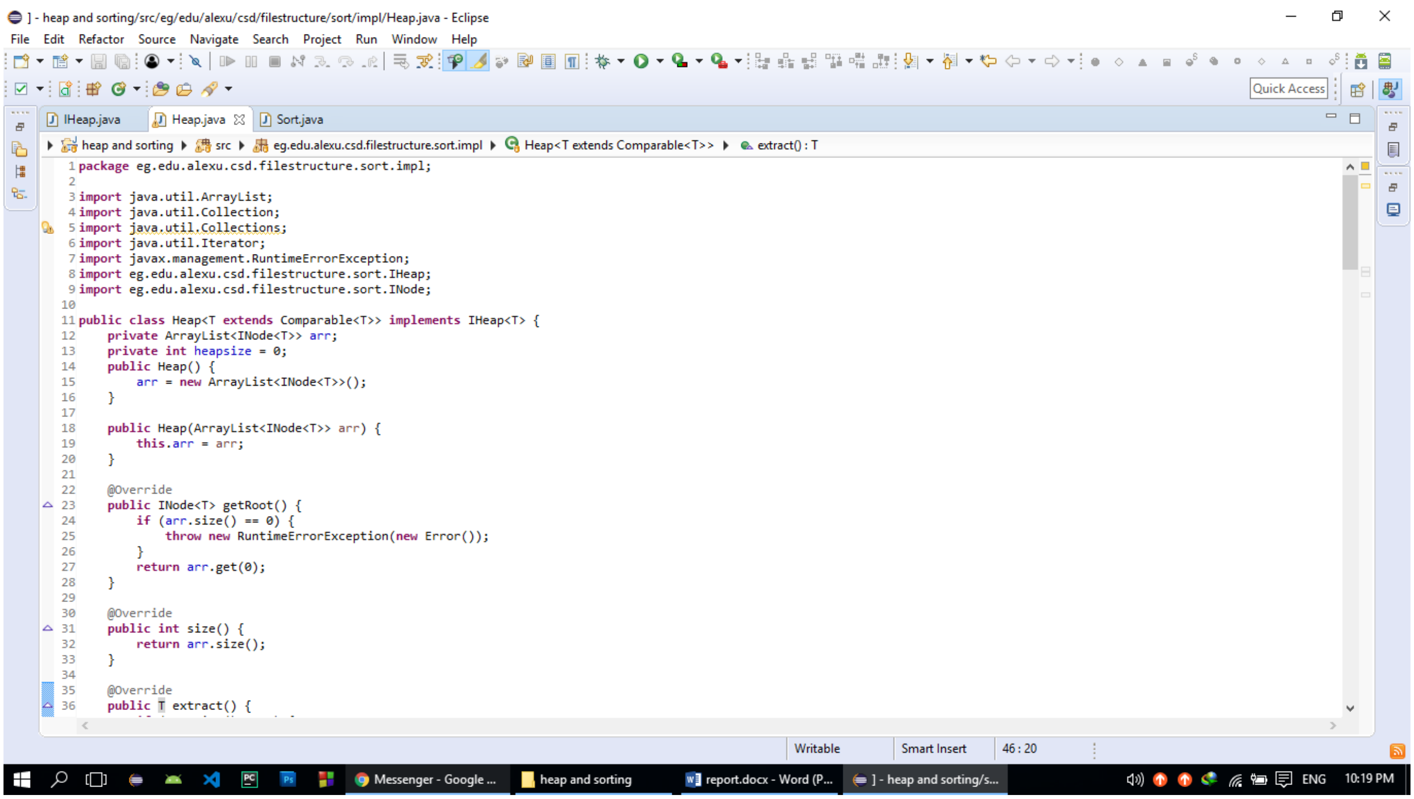
– An O (n log n) sorting algorithm such as Merge Sort or Quick sort algorithm in the average case.

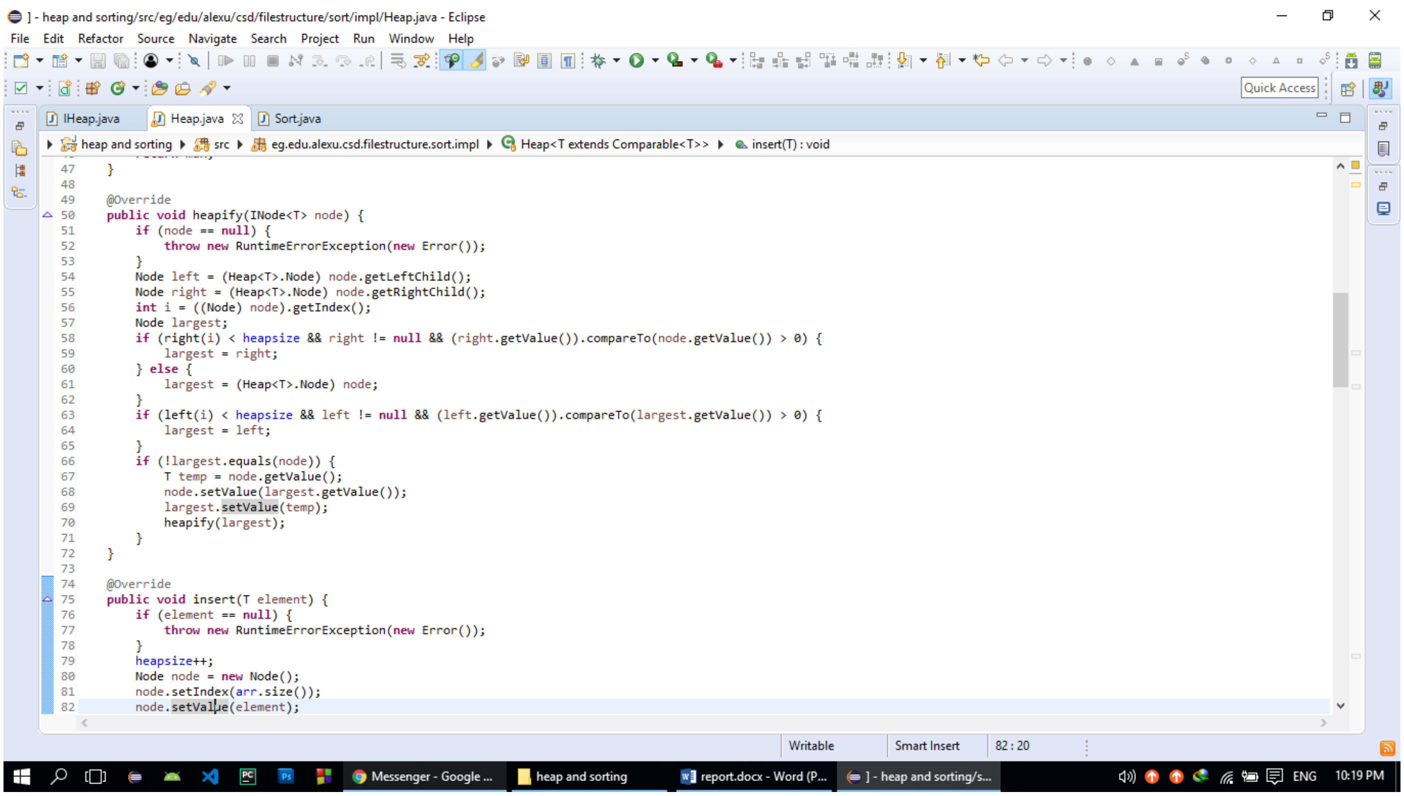
• In addition to heap sort, it is required to implement any of the sorting algorithms from each class mentioned above, O (n log n) and O (n^2). For example, I choose to implement Merge Sort and Bubble Sort.

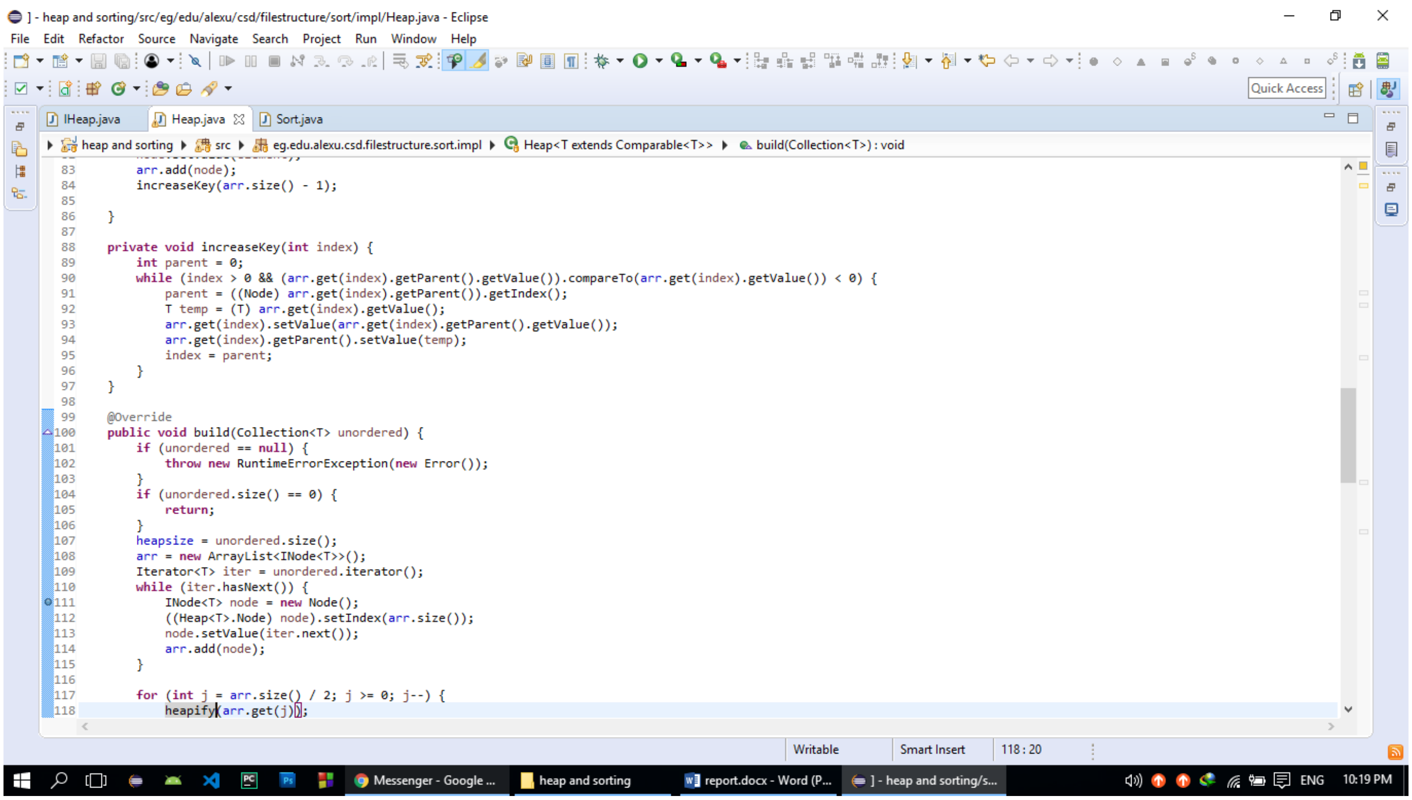
Data structures used:

- Java Array list: to store the nodes of the binary heap. The order of access its element is O (1) because of its random access property.

Code snippets:




] - heap and sorting/src/eg/edu/alexu/csd/filestructure/sort/impl/Heap.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
IHeap.java
Heap.java
Sort.java
heap and sorting
src
eg.edu.alexu.csd.filestructure.sort.impl
Heap<T extends Comparable<T>>
insert(T) : void
}
@Override
public void heapify(INode<T> node) {
if (node == null) {
throw new RuntimeErrorException(new Error());
}
Node left = (Heap<T>.Node) node.getLeftChild();
Node right = (Heap<T>.Node) node.getRightChild();
int i = ((Node) node).getIndex();
Node largest;
if (right(i) < heapsize && right != null && (right.getValue()).compareTo(node.getValue()) > 0) {
largest = right;
} else {
largest = (Heap<T>.Node) node;
}
if (left(i) < heapsize && left != null && (left.getValue()).compareTo(largest.getValue()) > 0) {
largest = left;
}
if (!largest.equals(node)) {
T temp = node.getValue();
node.setValue(largest.getValue());
largest.setValue(temp);
heapify(largest);
}
}
@Override
public void insert(T element) {
if (element == null) {
throw new RuntimeErrorException(new Error());
}
heapsize++;
Node node = new Node();
node.setIndex(arr.size());
node.setValue(element);
Writable
Smart Insert
82 : 20
Messenger - Google ...
heap and sorting
report.docx - Word (P...
] - heap and sorting/s...
ENG
10:19 PM

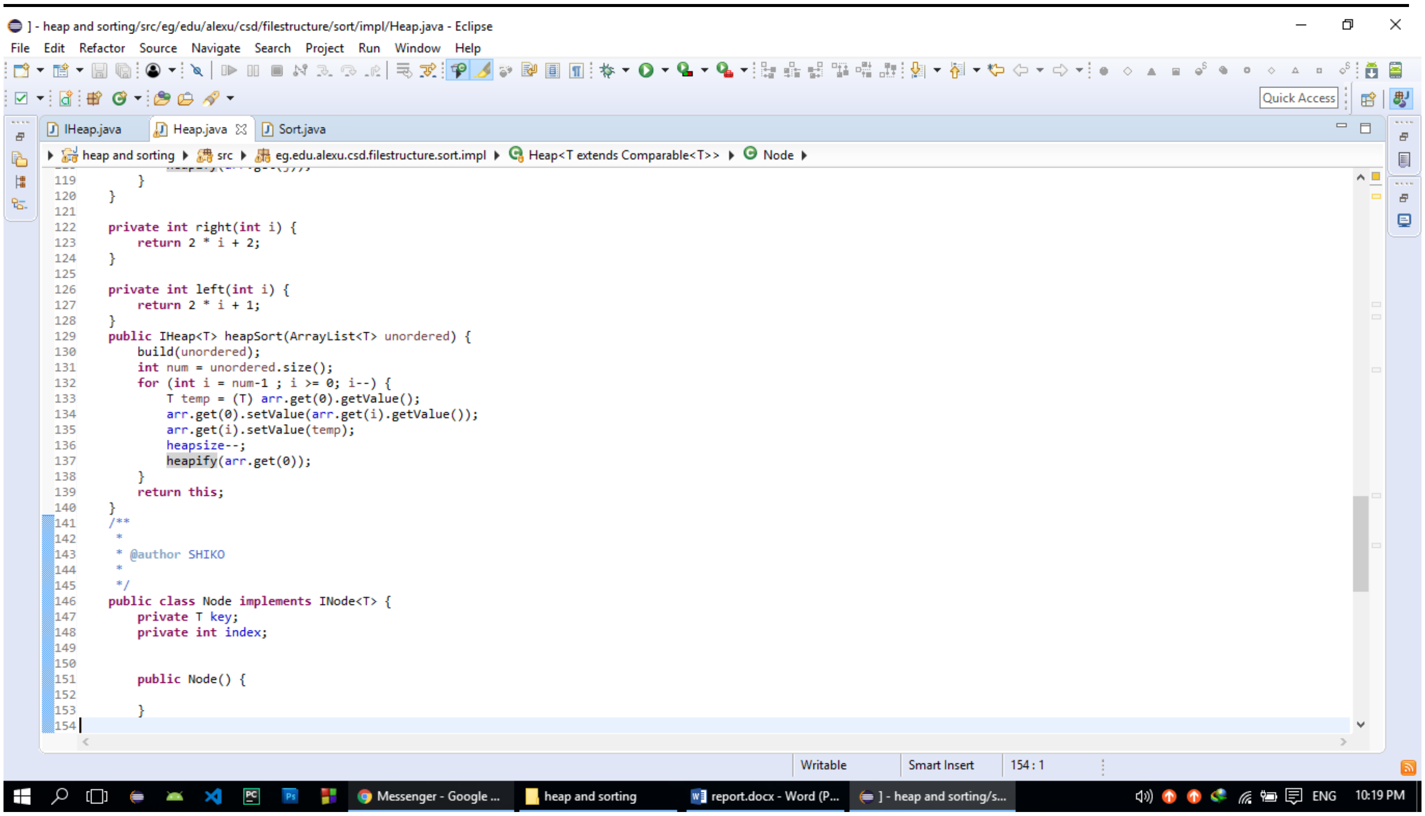

] - heap and sorting/src/eg/edu/alexu/csd/filestructure/sort/impl/Heap.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
IHeap.java
Heap.java
Sort.java
heap and sorting
src
eg.edu.alexu.csd.filestructure.sort.impl
Heap<T extends Comparable<T>>
build(Collection<T>) : void
        arr.add(node);
        increaseKey(arr.size() - 1);

    }

    private void increaseKey(int index) {
        int parent = 0;
        while (index > 0 && (arr.get(index).getParent().getValue()).compareTo(arr.get(index).getValue()) < 0) {
            parent = ((Node) arr.get(index).getParent()).getIndex();
            T temp = (T) arr.get(index).getValue();
            arr.get(index).setValue(arr.get(index).getParent().getValue());
            arr.get(index).getParent().setValue(temp);
            index = parent;
        }
    }

    @Override
    public void build(Collection<T> unordered) {
        if (unordered == null) {
            throw new RuntimeErrorException(new Error());
        }
        if (unordered.size() == 0) {
            return;
        }
        heapsize = unordered.size();
        arr = new ArrayList<INode<T>>();
        Iterator<T> iter = unordered.iterator();
        while (iter.hasNext()) {
            INode<T> node = new Node();
            ((Heap<T>.Node) node).setIndex(arr.size());
            node.setValue(iter.next());
            arr.add(node);
        }

        for (int j = arr.size() / 2; j >= 0; j--) {
            heapify(arr.get(j));
Writable
Smart Insert
118 : 20
Messenger - Google ...
heap and sorting
report.docx - Word (P...
] - heap and sorting/s...
ENG
10:19 PM

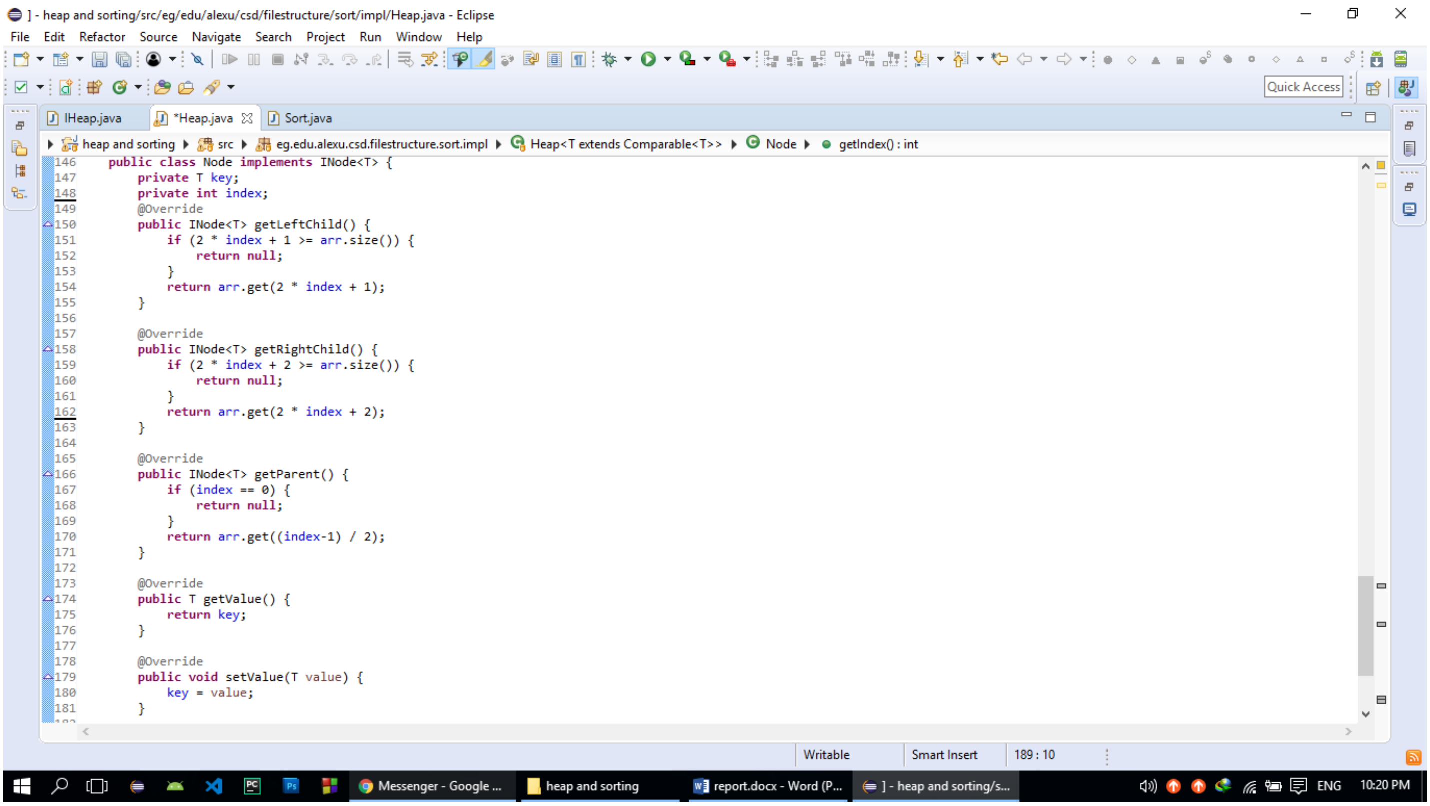

] - heap and sorting/src/eg/edu/alexu/csd/filestructure/sort/impl/Heap.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
IHeap.java
Heap.java
Sort.java
heap and sorting
src
eg.edu.alexu.csd.filestructure.sort.impl
Heap<T extends Comparable<T>>
Node
        }
    }

    private int right(int i) {
        return 2 * i + 2;
    }

    private int left(int i) {
        return 2 * i + 1;
    }
    public IHeap<T> heapSort(ArrayList<T> unordered) {
        build(unordered);
        int num = unordered.size();
        for (int i = num-1 ; i >= 0; i--) {
            T temp = (T) arr.get(0).getValue();
            arr.get(0).setValue(arr.get(i).getValue());
            arr.get(i).setValue(temp);
            heapsize--;
            heapify(arr.get(0));
        }
        return this;
    }
    /**
     *
     * @author SHIKO
     *
     */
    public class Node implements INode<T> {
        private T key;
        private int index;

        public Node() {

        }
Writable
Smart Insert
154 : 1
Messenger - Google ...
heap and sorting
report.docx - Word (P...
] - heap and sorting/s...
ENG
10:19 PM

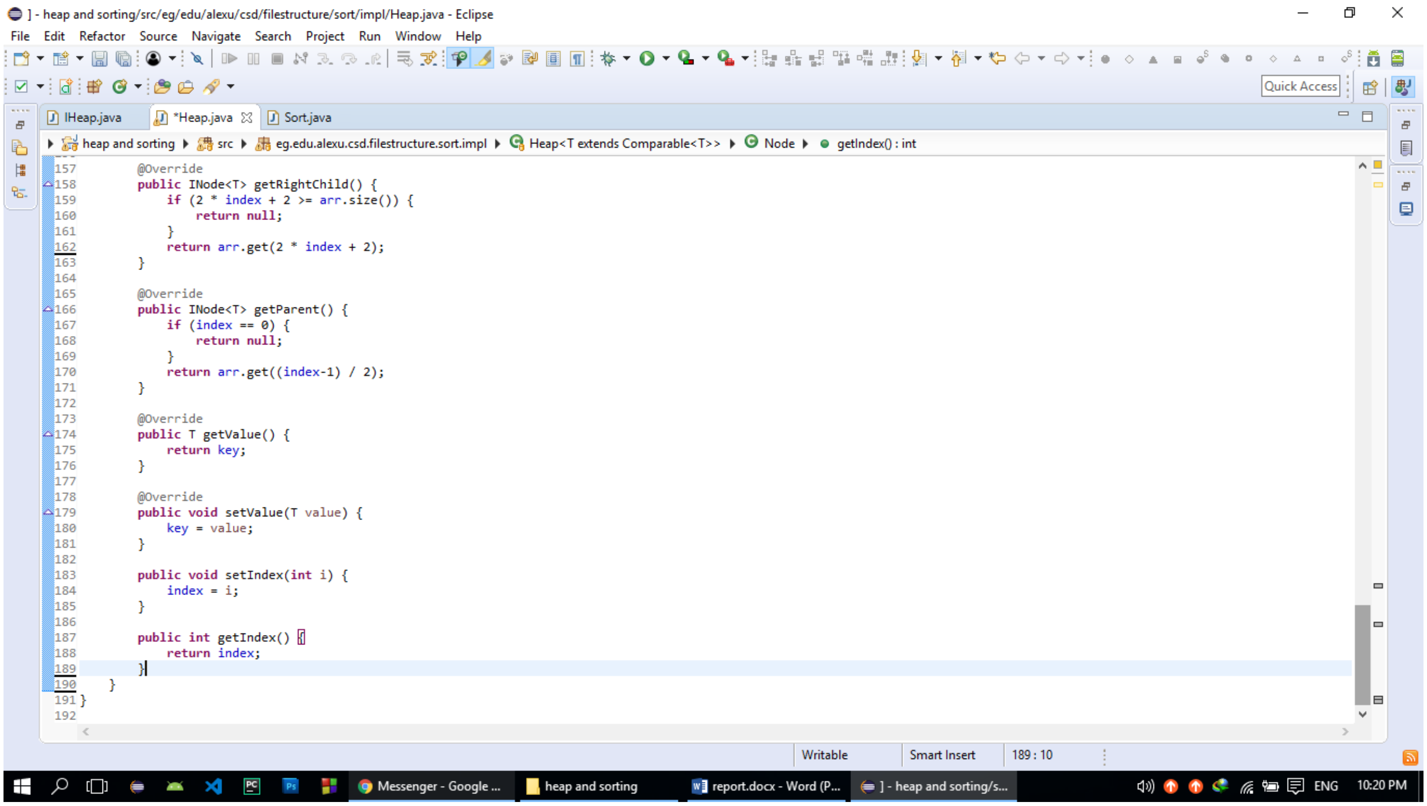

] - heap and sorting/src/eg/edu/alexu/csd/filestructure/sort/impl/Heap.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
IHeap.java
*Heap.java
Sort.java
heap and sorting > src > eg.edu.alexu.csd.filestructure.sort.impl > Heap<T extends Comparable<T>> > Node > getIndex() : int
    public class Node implements INode<T> {
        private T key;
        private int index;
        @Override
        public INode<T> getLeftChild() {
            if (2 * index + 1 >= arr.size()) {
                return null;
            }
            return arr.get(2 * index + 1);
        }

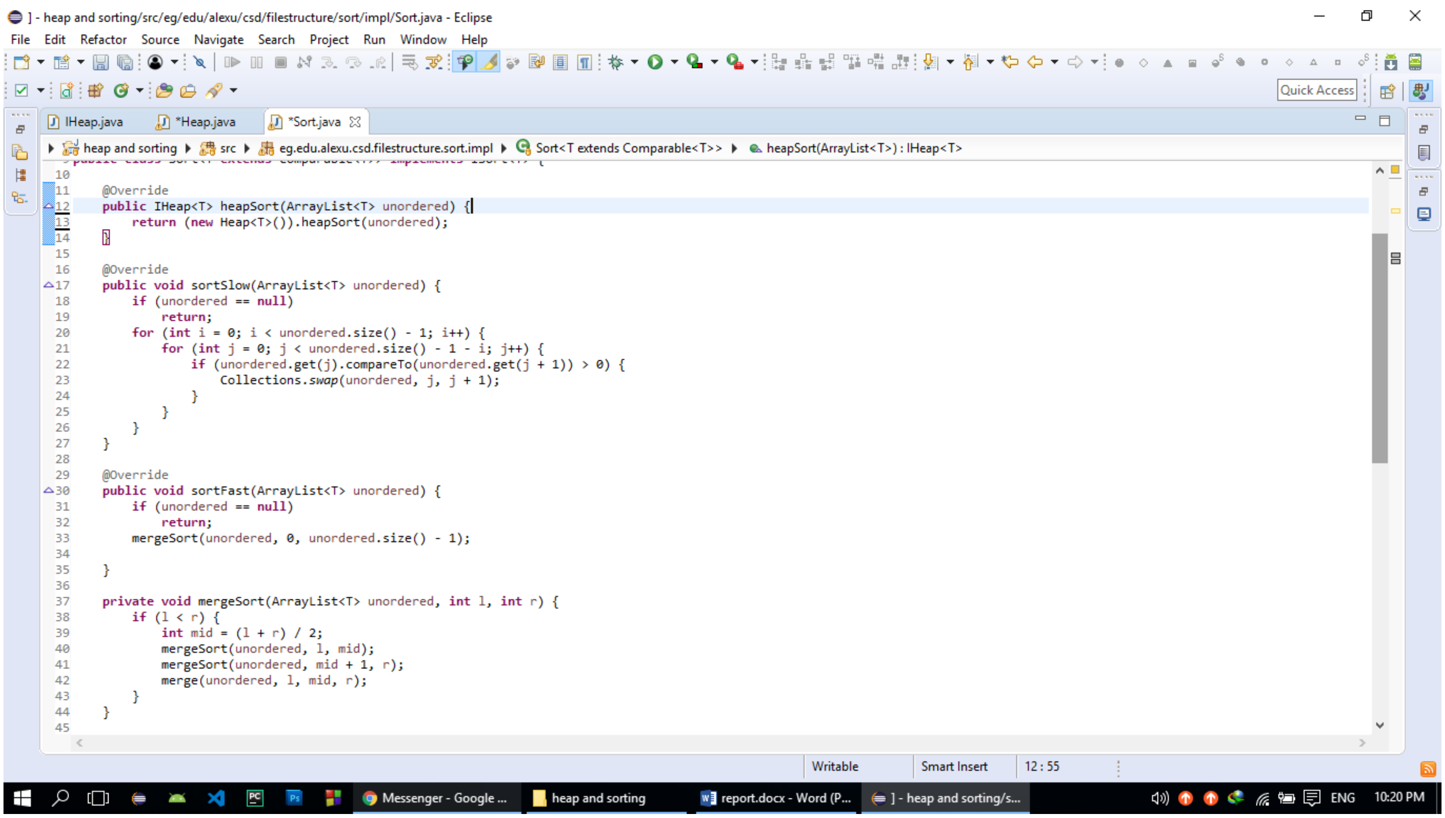
        @Override
        public INode<T> getRightChild() {
            if (2 * index + 2 >= arr.size()) {
                return null;
            }
            return arr.get(2 * index + 2);
        }

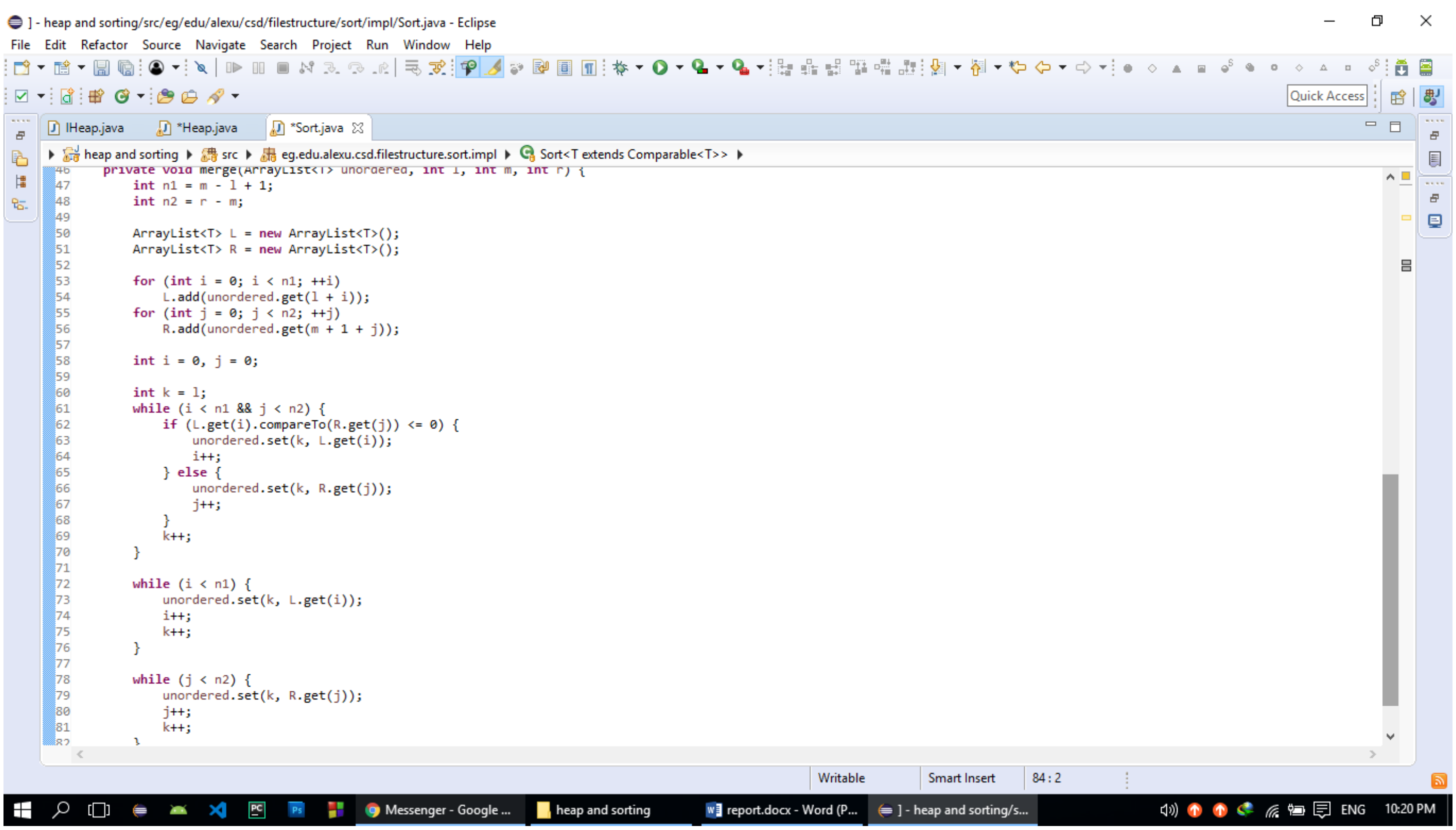
        @Override
        public INode<T> getParent() {
            if (index == 0) {
                return null;
            }
            return arr.get((index-1) / 2);
        }

        @Override
        public T getValue() {
            return key;
        }

        @Override
        public void setValue(T value) {
            key = value;
        }
Writable
Smart Insert
189 : 10
Messenger - Google ...
heap and sorting
report.docx - Word (P...
] - heap and sorting/s...
ENG
10:20 PM


] - heap and sorting/src/eg/edu/alexu/csd/filestructure/sort/impl/Heap.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
IHeap.java
*Heap.java
Sort.java
heap and sorting src eg.edu.alexu.csd.filestructure.sort.impl Heap<T extends Comparable<T>> Node getIndex() : int
@Override
public INode<T> getRightChild() {
if (2 * index + 2 >= arr.size()) {
return null;
}
return arr.get(2 * index + 2);
}
@Override
public INode<T> getParent() {
if (index == 0) {
return null;
}
return arr.get((index-1) / 2);
}
@Override
public T getValue() {
return key;
}
@Override
public void setValue(T value) {
key = value;
}
public void setIndex(int i) {
index = i;
}
public int getIndex() {
return index;
}
}
}
Writable
Smart Insert
189 : 10
Messenger - Google ...
heap and sorting
report.docx - Word (P...
] - heap and sorting/s...
ENG
10:20 PM


] - heap and sorting/src/eg/edu/alexu/csd/filestructure/sort/impl/Sort.java - Eclipse
IHeap.java
*Heap.java
*Sort.java
heap and sorting › src › eg.edu.alexu.csd.filestructure.sort.impl › Sort<T extends Comparable<T>> › heapSort(ArrayList<T>) : IHeap<T>
@Override
public IHeap<T> heapSort(ArrayList<T> unordered) {
return (new Heap<T>()).heapSort(unordered);
}
@Override
public void sortSlow(ArrayList<T> unordered) {
if (unordered == null)
return;
for (int i = 0; i < unordered.size() - 1; i++) {
for (int j = 0; j < unordered.size() - 1 - i; j++) {
if (unordered.get(j).compareTo(unordered.get(j + 1)) > 0) {
Collections.swap(unordered, j, j + 1);
}
}
}
}
@Override
public void sortFast(ArrayList<T> unordered) {
if (unordered == null)
return;
mergeSort(unordered, 0, unordered.size() - 1);
}
private void mergeSort(ArrayList<T> unordered, int l, int r) {
if (l < r) {
int mid = (l + r) / 2;
mergeSort(unordered, l, mid);
mergeSort(unordered, mid + 1, r);
merge(unordered, l, mid, r);
}
}
Writable
Smart Insert
12 : 55
Messenger - Google ...
heap and sorting
report.docx - Word (P...
] - heap and sorting/s...
ENG
10:20 PM


] - heap and sorting/src/eg/edu/alexu/csd/filestructure/sort/impl/Sort.java - Eclipse
IHeap.java
*Heap.java
*Sort.java
heap and sorting › src › eg.edu.alexu.csd.filestructure.sort.impl › Sort<T extends Comparable<T>>
private void merge(ArrayList<T> unordered, int l, int m, int r) {
int n1 = m - l + 1;
int n2 = r - m;
ArrayList<T> L = new ArrayList<T>();
ArrayList<T> R = new ArrayList<T>();
for (int i = 0; i < n1; ++i)
L.add(unordered.get(l + i));
for (int j = 0; j < n2; ++j)
R.add(unordered.get(m + 1 + j));
int i = 0, j = 0;
int k = l;
while (i < n1 && j < n2) {
if (L.get(i).compareTo(R.get(j)) <= 0) {
unordered.set(k, L.get(i));
i++;
} else {
unordered.set(k, R.get(j));
j++;
}
k++;
}
while (i < n1) {
unordered.set(k, L.get(i));
i++;
k++;
}
while (j < n2) {
unordered.set(k, R.get(j));
j++;
k++;
Writable
Smart Insert
84 : 2
Messenger - Google ...
heap and sorting
report.docx - Word (P...
] - heap and sorting/s...
ENG
10:20 PM