Section-End Project - Lesson 5

Data Structures - Sorting and Searching

Q1. How is Heap Sort implemented?

Create the solution in the Java Project called “Collections” used in Lesson 4.

### ****Heap Sort****

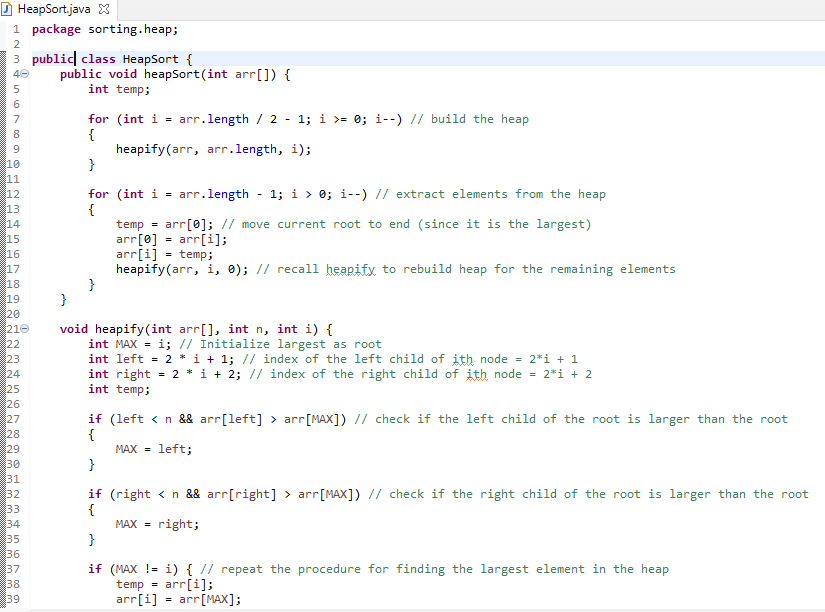
Heap sort is one of the most important sorting methods in java that one needs to learn to get into sorting. It combines the concepts of a tree as well as sorting, properly reinforcing the use of concepts from both. A heap is a complete binary search tree where items are stored in a special order depending on the requirement. A min-heap contains the minimum element at the root, and every child of the root must be greater than the root itself. The children at the level after that must be greater than these children, and so on. Similarly, a max-heap contains the maximum element at the root. For the sorting process, the heap is stored as an array where for every parent node at the index i, the left child is at index 2 \* i + 1, and the right child is at index 2 \* i + 2.

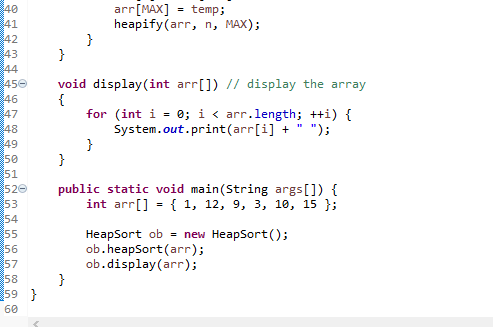
A [max heap](https://favtutor.com/blogs/heap-in-python" \o "Heap in Python | Min Heap and Max Heap Implementation" \t "https://favtutor.com/blogs/_blank) is built with the elements of the unsorted array, and then the maximum element is extracted from the root of the array and then exchanged with the last element of the array. Once done, the max heap is rebuilt for getting the next maximum element. This process continues till there is only one node present in the heap.

This algorithm has two main parts:-

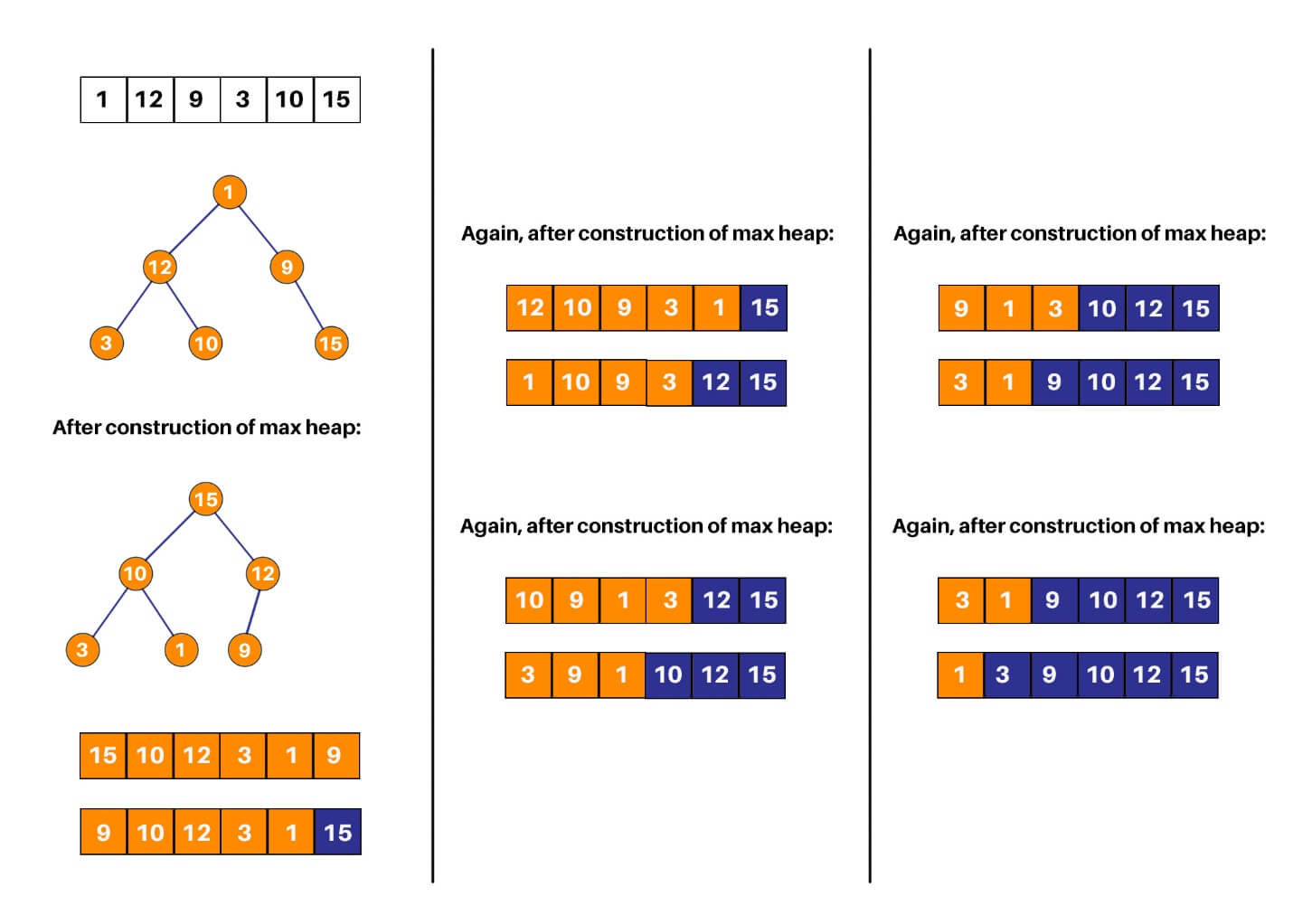
* **heapSort() –** This function helps construct the max heap initially for use. Once done, every root element is extracted and sent to the end of the array. Once done, the max heap is reconstructed from the root. The root is again extracted and sent to the end of the array, and hence the process continues.
* **heapify() –** This function is the building block of the heap sort algorithm. This function determines the maximum from the element being examined as the root and its two children. If the maximum is among the children of the root, the root and its child are swapped. This process is then repeated for the new root. When the maximum element in the array is found (such that its children are smaller than it) the function stops. For the node at index i, the left child is at index 2 \* i + 1, and the right child is at index 2 \* i + 1. (indexing in an array starts from 0, so the root is at 0).

### ****Heap Sort Java Code:****

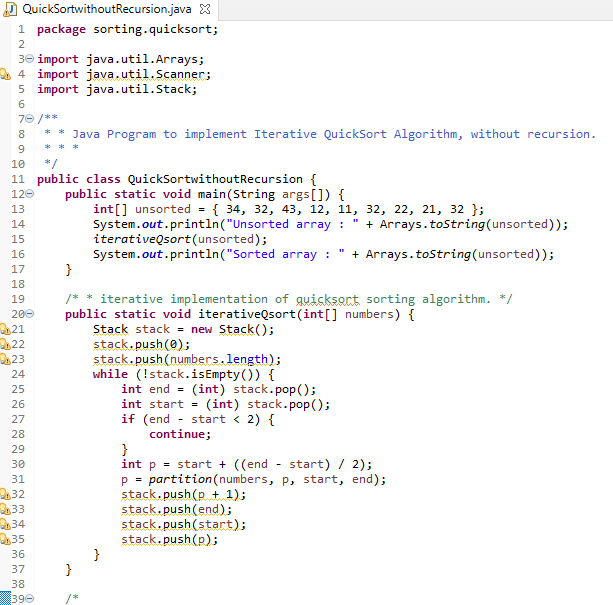


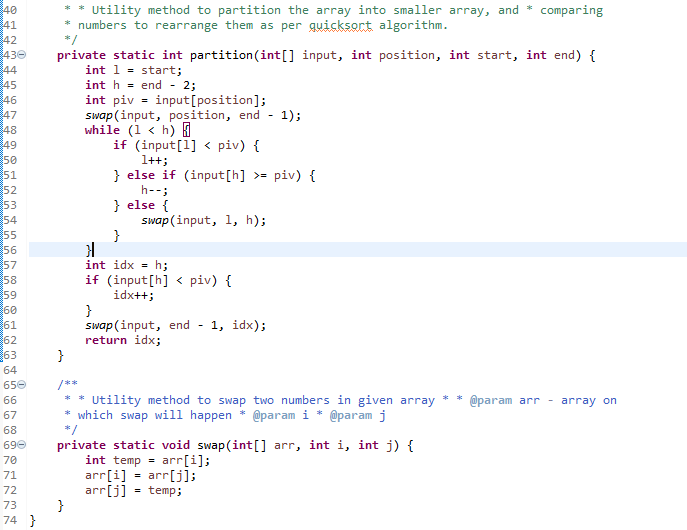


**Explanation of how it works:**



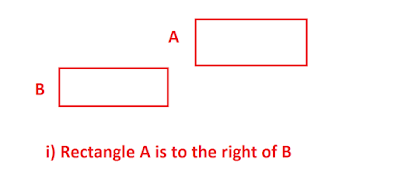
Q2. How is an iterative quicksort algorithm implemented?



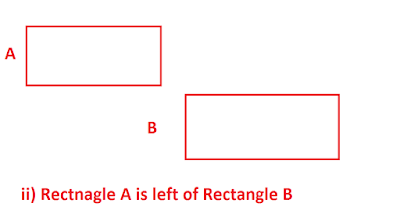


Q3. Write an algorithm to check if two rectangles overlap with each other?

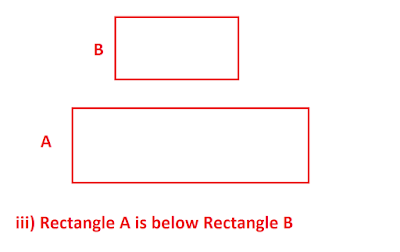
**Algorithm to check if rectangles are overlapping**  
Two rectangles A and B will not overlap or intersect with each other if one of the following four conditions is true.  
  
1. The left edge of A is to the right of the right edge of B. In this case, the first rectangle A is completely on the right side of second rectangle B as shown in the following diagram

[](https://3.bp.blogspot.com/-f3J9dGYwXh0/V_MMoRzsYHI/AAAAAAAAHLM/ok0OLWjf1g8zxckLR2jCcF0-Pkq5c605QCEw/s1600/How+to+check+if+two+Rectangle+overlap+each+other.png)

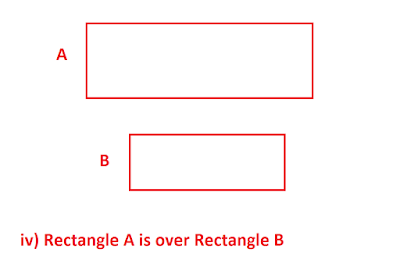
2. right edge of A is to the left of the left edge of B. In this case, the first rectangle A is completely on the left side of second rectangle B, as shown below

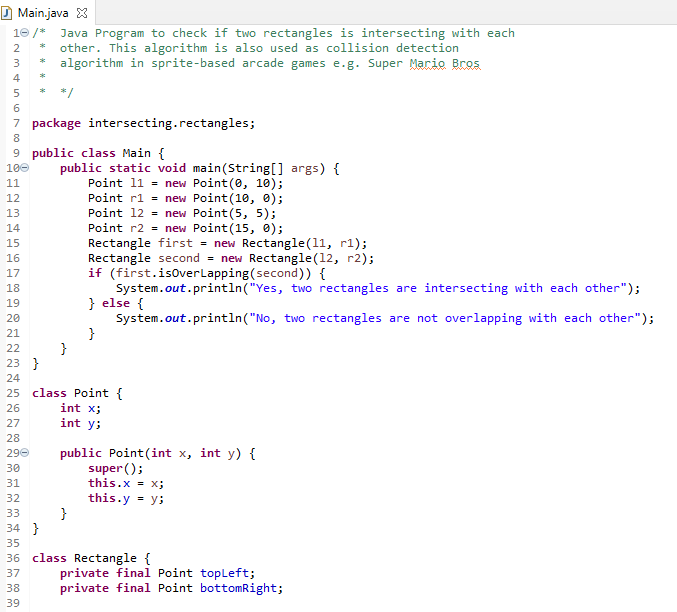
[](https://2.bp.blogspot.com/-f_xWiTxn8_Y/V_MMo3XjuLI/AAAAAAAAHLY/PjCIQHhA-pw8sfs8eFIYGabvX9QVJwLuwCEw/s1600/How+to+check+if+two+rectangle+collided+with+each+other.png)

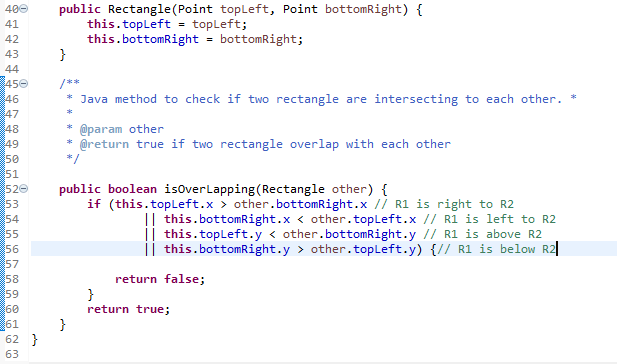
3. Top edge of A is below the bottom edge of B. In this case, the first rectangle A is completely under second rectangle B as shown in the following diagram

[](https://4.bp.blogspot.com/-kNfMOw-GHXI/V_MMoQufIgI/AAAAAAAAHLU/uBXEmioyCvopBeOsWMCrbT1SQ0WJ8D_MgCEw/s1600/Algorithm+of+collision+detection+of+two+rectangle+in+Java.png)

4. Bottom edge of A is above the top edge of B. In this case, the first rectangle A is completely above second rectangle B.

[](https://1.bp.blogspot.com/-vmwjYxekFmU/V_MMoSRME7I/AAAAAAAAHLQ/ZcYlC1NaAWkhsWxScUrvT0ACHb-jNet-QCEw/s1600/Algorithm+to+check+if+two+rectangle+overlap+with+each+other.png)

If any of the above four conditions are not true then two rectangles are overlapping with each other, the first condition is violated, hence rectangle A intersects rectangle B.  
  




\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*