Code	temporal	espacial
public void sortByNombreAtoZ() {	0	0
refreshArrayList();	O(1)	0
Client clientTmp;	1	1
for(int i = 0; i < clientsToSort.size(); i++) {	n+1	1
for(int j = 1; j < (clientsToSort.size() - i); j++) {	n(n+1)/2	1
if(clientsToSort.get(j-1).getName().compareTo(clientsToSort.get(j).getName()) >	(n(n+1)/2)-n	0
clientTmp = clientsToSort.get(j-1);	(n(n+1)/2)-n	1
clientsToSort.set(j - 1, clientsToSort.get(j));	(n(n+1)/2)-n	1
clientsToSort.set(j, clientTmp);	(n(n+1)/2)-n	1
}		
}		
}		
}		
Total	O(n^2)=((5n^2)/2)-(n/2)+3	6=0(1)

Code	temporal	espacial
public void sortByStartDate() {	0	0
refreshArrayList();	O(1)	0
Client[] b = new Client[clientsToSort.size()];	1	n
for(int i=0;i<b.length;i++) {	n+1	1
b[i] = clientsToSort.get(i);	n	1
}	0	0
mergeSort(b);	O(nlogn)	O(n)
clientsToSort.clear();	O(1)	0
for(Client tmp:b) {	n+1	1
clientsToSort.add(tmp);	n	0
}		
Total	n log(n)+ 4n+5	O(n)+n+3= O(n)

Code	temporal	espacial
public void sortByValue() {		
refreshArrayList();	O(1)	0
Client[] b = new Client[clientsToSort.size()];	1	n
for(int i=0;i<b.length;i++) {	n+1	1
b[i] = clientsToSort.get(i);	n	0
}		
quickSort(b,0,b.length-1);	O(nlogn)	O(log n)
clientsToSort.clear();	1	0
for(Client tmp:b) {	n+1	1
clientsToSort.add(tmp);	n	0
}		
Total	n log(n)+ 4n+5	log n+ n +2

Code	temporal	espacial
public void sortByCC() {		
refreshArrayList();	O(1)	0
MinHeap<Client> a = new MinHeap<Client>(100);	1	O(n)
for(Client tmp:clientsToSort) {	n+1	1
a.insert(tmp);	n	1
}		0
a.minHeap();	O(n log n)	O(1)
clientsToSort.clear();	1	0
int size = a.size();	1	1
for(int i=0;i<size;i++) {	n+1	1
clientsToSort.add(a.extractMin());	n	0

}		
}		
Total	(n log n)+4n+6	n+5= O(n)
public void minHeap() {		
for (int pos = (size/2)-1; pos>=0; pos--) {	(n/2)+1	1
minHeapify(pos);	(n log n)/2	0
}		
}		
Total	((n log n +n)/2)+1)= O(n log	O(1)