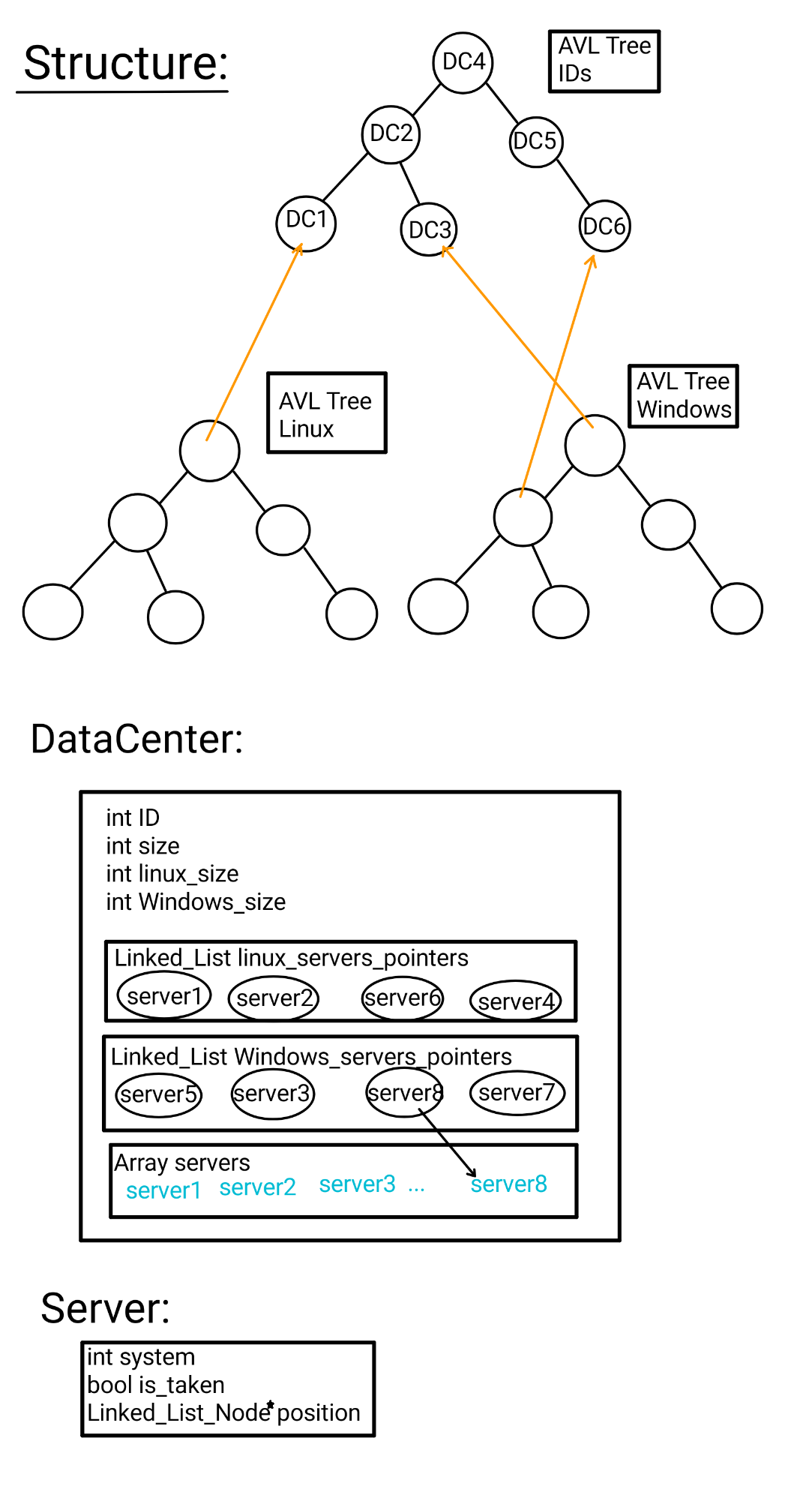
**Code Structure:**



Our code includes an interface class DataCenterManager, which works with the following underlying structure:

* An AVL Tree (marked “IDs” in sketch) that stores the DataCenters, ordered by ID. This tree is for retrieval of arbitrary DataCenters in O(logn) time complexity.
* An AVL Tree (marked “Linux” in sketch) that stores pointers (actually, wrapper classes for pointers, with minimal functionality) to DataCenters, ordered by number of Linux-running servers in a given DataCenter (and secondarily by ID). This tree is for maintaining a sorted ordering of the DataCenters in preparation for GetDataCenterByOS function call, and is revised every time a server changes operating system to maintain specified ordering.
* An AVL Tree (marked “Windows” in sketch) that stores pointers (actually, wrapper classes for pointers, with minimal functionality) to DataCenters, ordered by number of Windows-running servers in a given DataCenter (and secondarily by ID). This tree is for maintaining a sorted ordering of the DataCenters in preparation for GetDataCenterByOS function call, and is revised every time a server changes operating system to maintain specified ordering.

Structural Classes:

* DataCenter
  + Remembers its ID, size (total number of servers), linux\_size (number of Linux-based servers), and windows\_size (number of Windows-based servers)
  + Maintains an immutable array of its servers, sorted by server\_ID. This is to enable O(1) time complexity retrieval of servers.
  + 2 Linked lists containing pointers to servers (that actually reside in servers array), one for each operating system, sorted by order described in assignment. This is to allow O(1) retrieval of proper alternative server (as defined in assignment) in case requested server is already taken. Proper ordering is maintained by:
    - moving servers over to proper list when they change operating system,
    - always adding newly released serves to tail of linked list (queue-style) and always popping servers from the head, to maintain FIFO ordering (as described in assignment with “priorities”. Here priorities are defined as “distance from the tail of linked list”),
    - always removing successfully requisitioned servers from list, so that only available servers remain in list. Servers are returned to appropriate list when they are freed.
* Server
  + Remembers which operating system it is running, its availability status, and where it is located on the pair of linked lists its host DataCenter maintains (a pointer to the Node containing a pointer to said server). The location is to allow proper maintenance of the host DataCenter’s linked lists in O(1) time complexity.