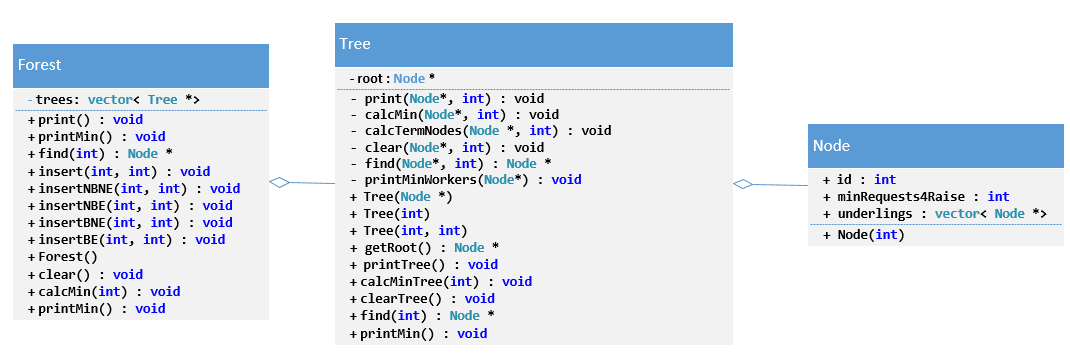
## UML Class Diagram



## Time/Space Analysis

### Forest

|  |  |  |
| --- | --- | --- |
| Function | Time | Space |
| void print() | //nodes \* trees  O(nt) | //nodes \* trees adding to the recursive stack  O(nt) |
| void printMin() | printMin() // 1  // 1 \* trees  O(t) | O(1) |
| Node \* find(int target) | //nodes \* trees  O(nt) | //nodes \* trees adding to the recursive stack  O(nt) |
| void insert(int boss, int underling) | N/A | N/A |
| void insertNBNE(int boss, int underling) | O(1) | O(1) //adding 1 item to the trees vector |
| void insertNBE(int boss, int underling) | O(1) | O(1) //adding 1 item to the trees vector |
| void insertBNE(int boss, int underling) | O(1) | O(1) //adding 1 item to the tree |
| void insertBE(int boss, int underling) | O(1) | O(1) //adding 1 item to the tree |
| Forest(); | O(1) | O(1) //adding 1 item to the trees vecor |
| void clear(); | //nodes \* trees  O(nt) | N/A |
| void calcMin(int percThreshold); | //nodes \* trees  O(nt) | //nodes \* trees adding to the recursive stack  O(nt) |

### Tree

|  |  |  |
| --- | --- | --- |
| Function | Time | Space |
| void print(Node\*ptr, int level) | O(n) //looping through each underling | O(n) //looping through recursion |
| void printMinWorkers(Node \* ptr) | O(1) | O(1) |
| void calcMin(Node\*ptr, int minRaisePerc) | calcTermNodes //n  n+n = O(n) | O(n) //loops through all nodes recursively adding to the stack |
| void calcTermNodes(Node \* ptr, int minRaisePerc) | O(n) //loops through all nodes | O(n) //loops through all nodes recursively adding to the stack |
| void clear(Node\*ptr, int level) | O(n) //loops through all nodes | N/A |
| Node \* find(Node\*root, int target) | O(n) //recursively loops through all nodes | O(n) //loops through all nodes recursively adding to the stack |
| Tree(Node \* rt = nullptr) | O(1) | O(1) |
| Tree(int id) | O(1) | O(1) |
| Tree(int boss, int emp) | O(1) | O(1) |
| Node \* getRoot() | O(1) | O(1) |
| void printTree() | Calls print(Node\*ptr, int level) // n  O(n) | Calls print(Node\*ptr, int level) // n  O(n) |
| void printMin() | Calls printMinWorkers(Node \* ptr) // 1  O(1) | Calls printMinWorkers(Node \* ptr) // 1  O(1) |
| void calcMinTree(int minRaisePerc) | Calls calcMin(Node\*ptr, int minRaisePerc) //n  O(n) | Calls calcMin(Node\*ptr, int minRaisePerc) //n  O(n) |
| void clearTree() | Calls clear(Node\*ptr, int level) // n  O(n) | N/A |
| Node \* find(int target) | Calls find(Node\*root, int target) // n  O(n) | Calls find(Node\*root, int target) // n  O(n) |

|  |  |  |
| --- | --- | --- |
| Function | Time | Space |
| readInput(Forest myFor, int employees, int threshold) | Tree::find() // nt  O(nt) | //nodes \* trees adding to the recursive stack  O(nt) |
| Main() | Nodes \* trees \* orgCharts  O(nto) | Nodes \* trees \* orgCharts recursively adding to stack  O(nto) |