|  |
| --- |
| **TAD BBT** |
| BBT<T,S> = {T value = null, S key = null} |
| { inv : x is the root and y a node. If y belongs to the left subtree, key[y] ≤ key[x] and if y belongs to the rigth subtree key[y] ≥ key[x]} |
| Primitive operations   * BBT None  BBT * insert Object, Object  BBT * search Object  BBT * delete BBT  BBT * balanceFactor BBT  int * rightRotate BBT  void * isLeaf BBT 🡪 boolean * leftRotate BBT  void * rebalanced BBT 🡪 void * bstHeight BBT 🡪 int |

Constructor

|  |
| --- |
| BBT()  “initialize the constructor of the BBT class  {pre: none}  { post : BBT = { T value= null, S key = null } } |

Modifier

|  |
| --- |
| insert(T value, S key)  “This method adds a new node with a value and key”  { pre : constructor already initialized, value & key Will be different of a null value}  { post : a node with value and key was added to the BBT} |

|  |
| --- |
| rightRotate(node BBT)  “This method rotate the tree to the right to balance the tree ”  { pre : constructor already initialized, node different of a null value}  { post : The tree has swung to the right} |

|  |
| --- |
| leftRotate(node BBT)  ““This method rotate the tree to the left to balance the tree ”  { pre : constructor already initialized, node different of a null value}  { post : The tree has swung to the left} |

Analyzer

|  |
| --- |
| search(S key)  “This method search and return the node BBT if that belong to BBT”  { pre : constructor already initialized }  { post : a null value if that key does not belong to the BBT or an BBT node if belong } |

|  |
| --- |
| rebalanced(node BBT)  “This method analyzes the unbalanced sectors of the tree and depending on the unbalance it orders the action of balancing to the left or right”  { pre : constructor already initialized , node different from null}  { post : The tree executes the order to move to the right or to the left depending on its imbalance} |

|  |
| --- |
| balanceFactor(node BBT)  “This method obtains the height of each node and subtracts the right height with the left to obtain the balance factor of each node.”  { pre : constructor already initialized , node different from null}  { post : An integer containing the balance factor of each node} |

|  |
| --- |
| bstHeight(node BBT)  “This method returns the height of the tree”  { pre : constructor already initialized , node different from null}  { post: integer that contain the tree height} |

|  |
| --- |
| isLeaf(node BBT)  “This method search if a BBT node has or not children”  { pre : constructor already initialized }  { post: true if the node is a leaf or false if not} |

Destruction

|  |
| --- |
| delete(node BBT)  “This method delete the node that is containing in BBT”  { pre : constructor already initialized, node different from null }  { post: delete the BBT node if that node is contained in the tree} |