Class Diagram1 (JULIAN FERNANDO(Universidad Icesi))

ui

Main
-gui : GUIController
+start(primaryStage : Stage) : void
+main(args : String[]) : void

-gui

GUIController
-mainPanel : BorderPane
-addFileButton : Button
-sortingComboBox : ComboBox<String>
-progressBar : ProgressBar
-labelURL : TextField
-searchTextField : TextField
-dataTableView : TableView<Record>
-nameColumn : TableColumn<Record, String>
-ageColumn : TableColumn<Record, String>
-teamColumn : TableColumn<Record, String>
-totalReboundsColumn : TableColumn<Record, String>
-offensiveReboundsColumn : TableColumn<Record, String>
-blocksColumn : TableColumn<Record, String>
-trueShootingColumn : TableColumn<Record, String>
-freeThrowColumn : TableColumn<Record, String>
-dataList : ObservableList<Record> = FXCollections.observableArrayList()
-fileChooser : FileChooser
-file : File
-masterClass : MasterClass
+GUIController()
+initialize() : void
+openAddFileCSV(event : ActionEvent) : void
+openAddAnElement(event : ActionEvent) : void
+openSearchElement(event : ActionEvent) : void
+addFile(event : ActionEvent) : void
+addElement(event : ActionEvent) : void
+search(event : ActionEvent) : void
+endProgram(event : ActionEvent) : void

Record
-name : SimpleStringProperty
-age : SimpleStringProperty
-team : SimpleStringProperty
-totalRebounds : SimpleStringProperty
-offensiveRebounds : SimpleStringProperty
-blocks : SimpleStringProperty
-trueShooting : SimpleStringProperty
-freeThrow : SimpleStringProperty
+getName() : String
+getAge() : String
+getTeam() : String
+getTotalRebounds() : String
+getOffensiveRebounds() : String
+getBlocks() : String
+getTrueShooting() : String
+getFreeThrow() : String
+Record(name : String, age : String, team : String, totalRebounds : String, offensiveRebounds : String, blocks : String, trueShooting : String, freeThrow : String)

model

-masterClass

MasterClass
-file : File
-cell : String[]
-ts : RedBlackTree<Double, ArrayList<Long>>
-fbr : AVL<Double, ArrayList<Long>>
-trb : AVL<Double, ArrayList<Long>>
-orb : AVL<Double, ArrayList<Long>>
-blk : BST<Double, ArrayList<Long>>
+MasterClass()
+readFiles(file : File) : void
+searchBybts(p : double) : ArrayList<String>
+searchBybrtb(p : double) : ArrayList<String>
+searchBybort(p : double) : ArrayList<String>
+searchBybtrb(p : double) : ArrayList<String>
+searchBybrt(p : double) : ArrayList<String>
+exportData(fileName : String) : void
+searchByfrt(p : double) : ArrayList<String>

structs

K : Comparable<K>
V

RedBlackNode
-BLACK : int = 0
-RED : int = 1
<<Property>> -key : K
<<Property>> -value : V
<<Property>> -color : int
<<Property>> -numLeft : int
<<Property>> -numRight : int
<<Property>> -parent : RedBlackNode<K, V>
<<Property>> -left : RedBlackNode<K, V>
<<Property>> -right : RedBlackNode<K, V>
+RedBlackNode()
+RedBlackNode(key : K, value : V)

-left
-parent
-right

-root

K : Comparable<K>
V

RedBlackTree
-nil : RedBlackNode<K, V> = new RedBlackNode<>()
<<Property>> -root : RedBlackNode<K, V> = structs.RedBlackTree.nil
+RedBlackTree()
+leftRotate(a : RedBlackNode<K, V>) : void
+leftRotateFixup(a : RedBlackNode<K, V>) : void
+rightRotate(a : RedBlackNode<K, V>) : void
+rightRotateFixup(a : RedBlackNode<K, V>) : void
+insert(k : K, v : V) : void
-insert(z : RedBlackNode<K, V>) : void
+insertFixup(a : RedBlackNode<K, V>) : void
+remove(k : K) : boolean
+remove(v : RedBlackNode<K, V>) : void
+removeFixup(a : RedBlackNode<K, V>) : void
+fixNodeData(a : RedBlackNode<K, V>, b : RedBlackNode<K, V>) : void
+search(key : K) : RedBlackNode<K, V>
+searchValue(key : K) : V
+keyExists(k : K) : boolean
-isNil(node : RedBlackNode<K, V>) : boolean
+treeMinimum(node : RedBlackNode<K, V>) : RedBlackNode<K, V>
+numGreater(key : K) : int
+numSmaller(key : K) : int
+findNumGreater(node : RedBlackNode<K, V>, key : K) : int
+findNumSmaller(node : RedBlackNode<K, V>, key : K) : int
+size() : int
+treeSuccessor(x : RedBlackNode<K, V>) : RedBlackNode<K, V>

-nil

K : Comparable<K>
V

<<Interface>>
RedBlackInterface
+leftRotate(a : RedBlackNode<K, V>) : void
+leftRotateFixup(a : RedBlackNode<K, V>) : void
+rightRotate(a : RedBlackNode<K, V>) : void
+rightRotateFixup(a : RedBlackNode<K, V>) : void
+insert(k : K, v : V) : void
+insertFixup(a : RedBlackNode<K, V>) : void
+remove(k : K) : boolean
+removeFixup(a : RedBlackNode<K, V>) : void
+fixNodeData(a : RedBlackNode<K, V>, b : RedBlackNode<K, V>) : void
+search(key : K) : RedBlackNode<K, V>

K : Comparable<K>
V

AVL
<<Property>> -size : int
<<Property>> -root : AVLNode<K, V>
+add(k : K, v : V) : boolean
+remove(k : K) : boolean
+searchNode(k : K) : AVLNode<K, V>
+search(k : K) : V
+isEmpty() : boolean
+size() : int
+keyExists(k : K) : boolean

-owner

-owner

-orb
-tr
-trb

K : Comparable<K>
V

<<Interface>>
AVLInterface
<<Property>> +empty : boolean
+add(k : K, v : V) : boolean
+remove(k : K) : boolean
+searchNode(k : K) : AVLNode<K, V>
+search(k : K) : V
+size() : int
+keyExists(k : K) : boolean

K : Comparable<K>
V

AVLNode
<<Property>> -key : K
<<Property>> -value : V
<<Property>> -weight : int
<<Property>> -lWeight : int
<<Property>> -rWeight : int
<<Property>> -balanceFactor : int
<<Property>> -owner : AVL<K, V>
<<Property>> -left : AVLNode<K, V>
<<Property>> -right : AVLNode<K, V>
<<Property>> -parent : AVLNode<K, V>
+AVLNode(k : K, v : V, owner : AVL<K, V>)
+add(k : K, v : V) : boolean
+remove(k : K) : boolean
+getSuccesor() : AVLNode<K, V>
-removeLeft() : boolean
+removeRight() : boolean
+getMinChild() : AVLNode<K, V>
-catchMinChild() : AVLNode<K, V>
+search(k : K) : AVLNode<K, V>
+leftRotate() : void
+rightRotate() : void
+calculateBalanceFactor() : void
+verifySpecialCase(n : int) : boolean
+balance() : boolean
+isBalanced() : boolean
+setValanceFactor(balanceFactor : int) : void

-root

-parent
-left

-right

K : Comparable<K>
V

<<Interface>>
BSTInterface
<<Property>> +empty : boolean
+add(k : K, v : V) : boolean
+remove(k : K) : boolean
+searchNode(k : K) : Node<K, V>
+search(k : K) : V
+size() : int
+keyExists(k : K) : boolean

K : Comparable<K>
V

Node
<<Property>> -key : K
<<Property>> -value : V
<<Property>> -BalanceFactor : int
<<Property>> -parent : Node<K, V>
<<Property>> -left : Node<K, V>
<<Property>> -right : Node<K, V>
<<Property>> -owner : AVL<K, V>
+Node(k : K, v : V)
+Node(k : K, v : V, a : boolean)
+add(k : K, v : V) : boolean
+remove(k : K) : boolean
-removeLeft() : boolean
+removeRight() : boolean
+getSuccesor() : Node<K, V>
+getMinChild() : Node<K, V>
+search(k : K) : Node<K, V>

-left
-parent
-right

-root

K : Comparable<K>
V

BST
<<Property>> -size : int
<<Property>> -root : Node<K, V>
+BST()
+add(k : K, v : V) : boolean
+remove(k : K) : boolean
+isEmpty() : boolean
+size() : int
+keyExists(k : K) : boolean
+searchNode(k : K) : Node<K, V>
+search(k : K) : V

-blk