Visual Paradigm Professional(Commercial License)

dataStructures

**GenericRedBlackTreeTest**
- testTree : GenericRedBlackTree<String, Integer>
+setupScenary1() : void
+setupScenary2() : void
+setupScenary3() : void
+setupScenary4() : void
+setupScenary5() : void
+testTest1() : void
+insertTest2() : void
+insertTest3() : void
+insertTest4() : void
+getGreaterThanTest() : void
+getLowerThanTest() : void

testTree (1) → **GenericRedBlackTree** <K : Comparable<K>, V>
- root : Node<K, V> = null
+GenericRedBlackTree()
-leftRotate(x : Node<K, V>) : void
-leftRotateFixup(x : Node<K, V>) : void
-rightRotate(y : Node<K, V>) : void
-rightRotateFixup(y : Node<K, V>) : void
+insert(key : K, value : V) : void
-insert(z : Node<K, V>, value : V) : void
-insertFixup(z : Node<K, V>) : void
+search(key : K) : List<V>
-contains(key : K) : Node<K, V>
-getGreaterThan(key : K, maxReturned : Integer) : List<V>
-getGreaterThan(current : Node<K, V>, key : K, list : List<V>) : void
+getLowerThan(key : K, maxReturned : Integer) : List<V>
-getLowerThan(current : Node<K, V>, key : K, list : List<V>) : void
-isnull(node : Node<K, V>) : boolean
+reset() : void
+getRoot() : Node<K, V>

**Node** <K : Comparable<K>, V>
-BLACK : int = 0
-RED : int = 1
- key : K
- numLeft : int
- numRight : int
- color : int
- heigth : int
- balance# : int
- values : List<V>
- father : Node<K, V>
- left : Node<K, V>
- right : Node<K, V>
+Node(key : K, value : V)
+addElement(newValue : V) : void
+getHeight() : int
+setHeight(h : int) : void
+getKey() : K
+setKey(key : K) : void
+getNumLeft() : int
+setNumLeft(numLeft : int) : void
+getNumRight() : int
+setNumRight(numRight : int) : void
+getColor() : int
+setColor(color : int) : void
+getBalance() : int
+setBalance(balance# : int) : void
+getValues() : List<V>
+setValues(values : List<V>) : void
+getFather() : Node<K, V>
+setFather(father : Node<K, V>) : void
+getLeft() : Node<K, V>
+setLeft(left : Node<K, V>) : void
+getRight() : Node<K, V>
+setRight(right : Node<K, V>) : void

-root (0..1) → Node; -father (0..1); -left (0..1); -right (0..1)

**<<Interface>> GenericRedBlackTreeInterface** <K, V>
+insert(key : K, value : V) : void
+search(key : K) : List<V>
+getGreaterThan(key : K, maxReturned : Integer) : List<V>
+getLowerThan(key : K, maxReturned : Integer) : List<V>

-RedBlackGeneralEvaluation (0..1) → GenericRedBlackTree

**GenericBinarySearchTreeTest**
- testTree : GenericBinarySearchTree<String, Integer>
+setupScenary1() : void
+setupScenary2() : void
+setupScenary3() : void
+setupScenary4() : void
+putTest1() : void
+putTest2() : void
+getGreaterThanTest() : void
+getLowerThanTest() : void

testTree (1) → **GenericBinarySearchTree** <K : Comparable<K>, V>
- root : Node<K, V>
+GenericBinarySearchTree()
+GenericBinarySearchTree(k : K, v : V)
+put(key : K, val : V) : void
-put(x : Node<K, V>, key : K, val : V) : Node<K, V>
+search(key : K) : List<V>
-searchRecursive(key : K, currentNode : Node<K, V>) : Node<K, V>
+getGreaterThan(key : K, maxReturned : Integer) : List<V>
-getGreaterThan(current : Node<K, V>, key : K, list : List<V>) : void
+getLowerThan(key : K, maxReturned : Integer) : List<V>
-getLowerThan(current : Node<K, V>, key : K, list : List<V>) : void
-isnull(node : Node<K, V>) : boolean
+reset() : void
+getRoot() : Node<K, V>

-BSTPointsPerGame (0..1) → GenericBinarySearchTree

**<<Interface>> GenericBinarySearchTreeInterface** <K, V>
+put(k : K, v : V) : void
+search(k : K) : List<V>
+getGreaterThan(key : K, maxReturned : Integer) : List<V>
+getLowerThan(key : K, maxReturned : Integer) : List<V>

**GenericAVLTreeTest**
- testTree : GenericAVLTree<String, Integer>
+setupScenary1() : void
+setupScenary2() : void
+setupScenary3() : void
+setupScenary4() : void
+setupScenary5() : void
+setupScenary6() : void
+insertTest1() : void
+insertTest2() : void
+insertTest3() : void
+insertTest4() : void
+SearchTest1() : void
+getGreaterThanTest() : void
+getLowerThanTest() : void

testTree (1) → **GenericAVLTree** <K : Comparable<K>, V>
- root : Node<K, V>
+GenericAVLTree()
+GenericAVLTree(k : K, v : V)
+search(key : K) : List<V>
-searchRecursive(key : K, currentNode : Node<K, V>) : Node<K, V>
+insert(key : K, value : V) : void
-insert(node : Node<K, V>, key : K, value : V) : Node<K, V>
-update(node : Node<K, V>) : void
-balance(node : Node<K, V>) : Node<K, V>
-leftLeftCase(node : Node<K, V>) : Node<K, V>
-leftRightCase(node : Node<K, V>) : Node<K, V>
-rightRightCase(node : Node<K, V>) : Node<K, V>
-rightLeftCase(node : Node<K, V>) : Node<K, V>
-leftRotation(node : Node<K, V>) : Node<K, V>
-rightRotation(node : Node<K, V>) : Node<K, V>
+getGreaterThan(key : K, maxReturned : Integer) : List<V>
-getGreaterThan(current : Node<K, V>, key : K, list : List<V>) : void
+getLowerThan(key : K, maxReturned : Integer) : List<V>
-getLowerThan(current : Node<K, V>, key : K, list : List<V>) : void
-isnull(node : Node<K, V>) : boolean
+reset() : void
+getRoot() : Node<K, V>

-AVLReboundsPerGame (0..1), -AVLAssistsPerGame (0..1), -AVLBlocksPerGame (0..1), -AVLRobberiesPerGame (0..1) → GenericAVLTree

**<<Interface>> GenericAVLTreeInterface** <K, V>
+insert(k : K, v : V) : void
+search(k : K) : List<V>
+getGreaterThan(key : K, maxReturned : Integer) : List<V>
+getLowerThan(key : K, maxReturned : Integer) : List<V>

model

**QuickBasketManager**
-SEPARATOR : String = ";"
-SAVE_PATH_FILE : String = "dataFIBAdatas2.csv"
-BSTPointsPerGame : GenericBinarySearchTree<Double, Player>
-AVLReboundsPerGame : GenericAVLTree<Double, Player>
-AVLAssistsPerGame : GenericAVLTree<Double, Player>
-AVLRobberiesPerGame : GenericAVLTree<Double, Player>
-AVLBlocksPerGame : GenericAVLTree<Double, Player>
-RedBlackGeneralEvaluation : GenericRedBlackTree<Double, Player>
-playersList : Player
+QuickBasketManager()
+searchUniqueParameter(option : int, key : double, searchGreaterThan : int) : List<Player>
+importData() : void
+exportData() : void
+reset() : void
+addNewPlayer(id : int, fullName : String, age : int, team : String, pointsPerGame : double, reboundsPerGame : double, assistsPerGame : double, robberiesPerGame : double, blocksPerGame : double, generalEvaluation : double) : String
+deletePlayer(id : int) : String
+binarySearchPlayer(id : int) : boolean
+binarySearchPlayer2(id : int) : int
+getPlayersListJunitTest() : List<Player>

1 -qbm → QuickBasketManager; 1 -qbl → QuickBasketManager

**QuickBasketManagerTest**
-qbm : QuickBasketManager
+setupScenary1() : void
+addAndSearchAndDeleteTest1() : void
+resetTest1() : void

* -playersList → **Player**
-id : int
-fullname : String
-age : int
-team : String
-pointsPerGame : double
-reboundsPerGame : double
-assistsPerGame : double
-robberiesPerGame : double
-blocksPerGame : double
-generalEvaluation : double
+Player(id : int, fullName : String, age : int, team : String, pointsPerGame : double, reboundsPerGame : double, assistsPerGame : double, robberiesPerGame : double, blocksPerGame : double, generalEvaluation : double)
+toString() : String
+compareTo(p : Player) : int
+getId() : int
+setId(id : int) : void
+getFullname() : String
+setFullName(fullName : String) : void
+getAge() : int
+setAge(age : int) : void
+getTeam() : String
+setTeam(team : String) : void
+getPointsPerGame() : double
+setPointsPerGame(pointsPerGame : double) : void
+getReboundsPerGame() : double
+setReboundsPerGame(reboundsPerGame : double) : void
+getAssistsPerGame() : double
+setAssistsPerGame(assistsPerGame : double) : void
+getRobberiesPerGame() : double
+setRobberiesPerGame(robberiesPerGame : double) : void
+getBlocksPerGame() : double
+setBlocksPerGame(blocksPerGame : double) : void
+getGeneralEvaluation() : double
+setGeneralEvaluation(generalEvaluation : double) : void

ui

**QuickBasketGUI**
-basePane : BorderPane
-consultOptions : ComboBox<String>
-consultyeboi : TextField
-consultbyeelabel : Label
-referencenumbeitxt : TextField
-consultype : Combobox<String>
-showResultBtn : Button
-otherOperationsCp : ComboBox<String>
-labelseparation : Label
-idToAddTxt : TextField
-idloaddlabel : Label
-pointspergametxt : TextField
-reboundspergametxt : TextField
-assistspergametxt : TextField
-robberiespergametxt : TextField
-fullnametxt : TextField
-AgeTxt : TextField
-blockspergametxt : TextField
-teamtxt : TextField
-generalevaluationtxt : TextField
-fullnamelabel : Label
-agelabel : Label
-teamlabel : Label
-pointspergamelabel : Label
-assistspergamelabel : Label
-reboundspergamelabel : Label
-robberiesPergamelabel : Label
-blockspergamelabel : Label
-generalevaluationlabel : Label
-addBtn : Button
-deleteBtn : Button
-idToDeleteTxt : Label
-idToDelete : TextField
-playersTable : TableView<Player>
-idC : TableColumn<Player, Integer>
-fullnameC : TableColumn<Player, String>
-AgeC : TableColumn<Player, Integer>
-teamC : TableColumn<Player, String>
-pointspergameC : TableColumn<Player, Double>
-reboundsC : TableColumn<Player, Double>
-assistsC : TableColumn<Player, Double>
-robberiesC : TableColumn<Player, Double>
-blocksC : TableColumn<Player, Double>
-generalEC : TableColumn<Player, Double>
-qbm : QuickBasketManager
+QuickBasketGUI(q : QuickBasketManager)
+loadMainMenu() : void
+toChooseCounsul(event : ActionEvent) : void
+setNotVisible() : void
+consultOperation() : List<Player>
+initializechoiboxconsultoptions() : void
+initializechoiceboxconsultype() : void
+loadPlayerScreen() : void
-showResult1(event : ActionEvent) : void
+initializeplayersTable() : void
-toChooseOtherOperations(event : ActionEvent) : void
+initializechoiceboxotheroperationstype() : void
+checkOtherOp(event : ActionEvent) : void
+showOtherOperationsMenu() : void
-addPlayer(event : ActionEvent) : void
-returnPlayer(event : ActionEvent) : void
-toDeletePlayer(event : ActionEvent) : void
+showAlertWhenIdAlreadyExists() : void
+showAlertWhenPlayerDoesntExist() : void
+showAlertWhenPlayersAdded() : void
+showAlertWhenPlayersDeleted() : void
+clearTextAdd() : void
+clearTextDeleted() : void

-guiMain (1) → **Main**
-guiMain : QuickBasketGUI
-qb : QuickBasketManager
+Main()
-main(args : String[]) : void
+start(primaryStage : Stage) : void