DESING DOCUMENTATION

PARTICIPANTS

CAMILO CAMPAZ JIMÉNEZ

DANIEL ESTEBAN JARABA GAVIRIA

JOHAN STIVEN RICARDO SIBAJA

TEACHER

URAM ANIBAL SOSA

INTEGRATIVE TASK #2

ALGORITHMS AND DATA STRUCTURES

2021-02

Tabla de contenido

Functional Requirements

The program must be able to:

1. **Enter** data of the players, that is name, age, team and 5 statistics (e.g. points per game, rebounds per game, assists per game, steals per game, blocks per game), either in bulk (with .csv files for example) or through an interface.

2. **Delete** or modify data of any player selected by the user and save the changes, showing them in a table.

3. **Make** player queries using the statistical categories included as search criteria. This search criteria could be given as an interval, building the player queries with the data which belongs to the interval.

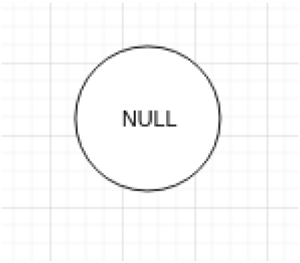
4. **Retrieve** players according to the selected search category and the value given for it.

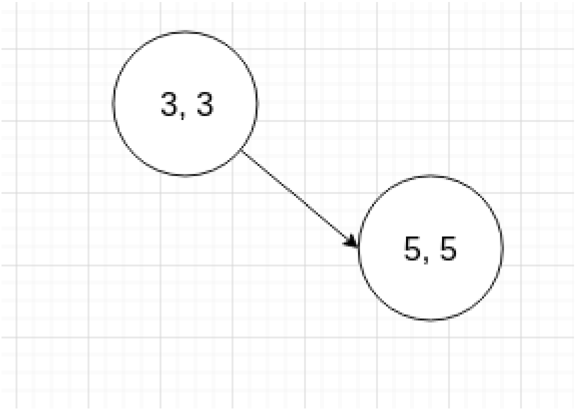
5. **Run** player queries according to all criteria, which correspond to the attributes of a player name, age, team or any of the 5 statistics.

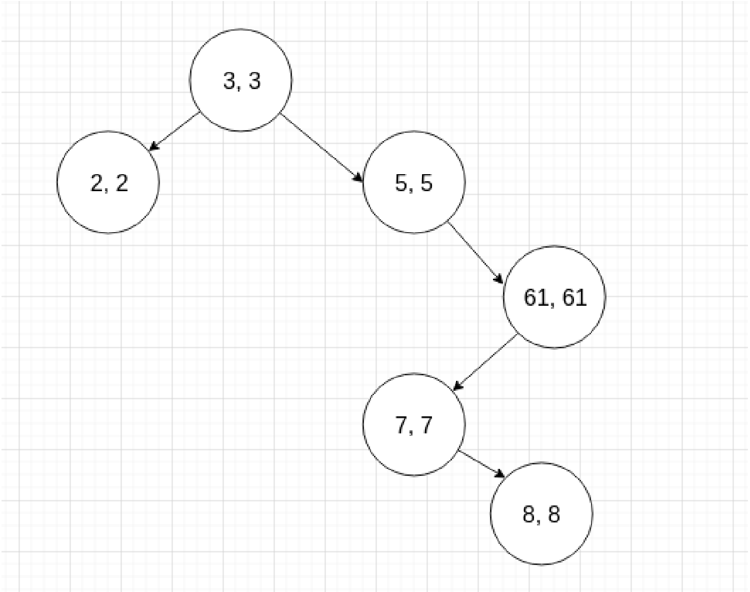
UNIT TEST DESIGNS

Data structures classes

BinarySearchTree class:

Name	Class	Stage
setup1	BinarySearchTree	

Name	Class	Stage
setup2	BinarySearchTree	

Name	Class	Stage
setup3	BinarySearchTree	

Test goal: Verify if the method addNode is able to set new nodes in the tree				
Class	**Method**	**Stage**	**Input values**	**Result**

BinarySearchTree	addNode()	setup1	element = 3
			key = 3
			
			element = 5
			key = 5
				False, which means that the tree set in the stage1 it has a root and right child not null

Test goal: Verify if the method delete is able to set the weight of the tree and delete correctly an specific node				
Class	**Method**	**Stage**	**Input values**	**Result**
BinarySearchTree	delete()	setup2	element = 5	1, which is the new weight of the tree set in the stage2 after deleting its right child

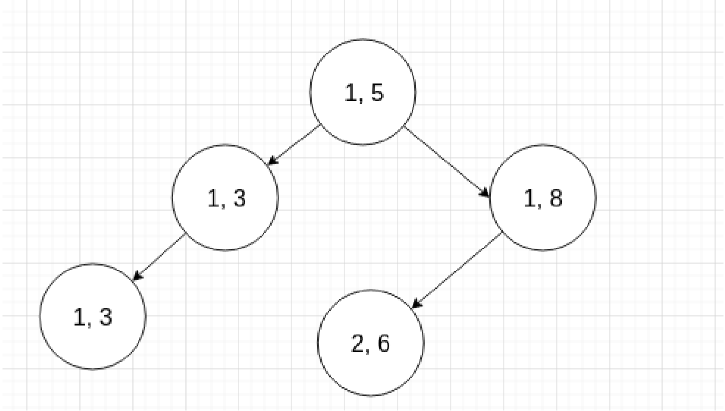
Test goal: Verify if the method search returns the correct value that want to be searched				
Class	**Method**	**Stage**	**Input values**	**Result**
BinarySearchTree	search()	setup2	key = 5	True, which means that the node searched is not null

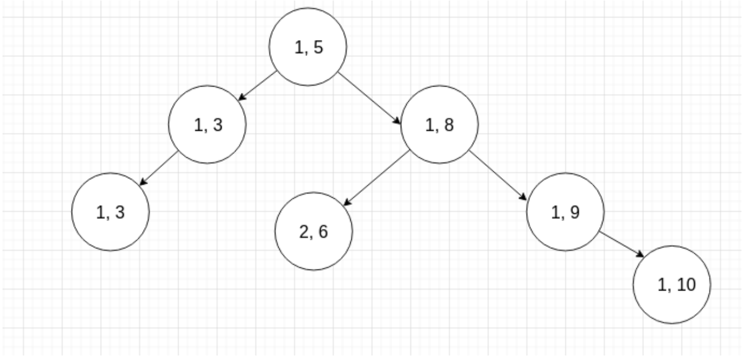
Test goal: Verify is the method is able to find the correct value of the successor for an specific node				
Class	**Method**	**Stage**	**Input values**	**Result**
BinarySearchTree	successor()	setup3	key = 2	3, which is the correct successor of the node whit key 2

Test goal: Verify is the method is able to find the correct value of the successor for an specific node				
Class	**Method**	**Stage**	**Input values**	**Result**
BinarySearchTree	successor()	setup3	key = 5	7, which is the correct successor of the node whit key 5

Test goal: Verify is the method is able to find the correct value of the successor for an specific node				
Class	**Method**	**Stage**	**Input values**	**Result**
BinarySearchTree	successor()	setup3	key = 3	5, which is the correct successor of the node whit key 3

AVLTree class:

Name	Class	Stage
setup1	AVLTree	

Name	Class	Stage
setup2	AVLTree	

Test goal: check if the method is able to indicate if the tree is balanced				
Class	**Method**	**Stage**	**Input values**	**Result**
AVLTree	isBalanced()	setup1	none	True, because the tree created in the stage1 is balanced, by the criteria of ist rolling factor

<table>
<tr><td colspan="5"><b>Test goal: check if the method is able to indicate if the tree is balanced</b></td></tr>
<tr><th>Class</th><th>Method</th><th>Stage</th><th>Input values</th><th>Result</th></tr>
<tr><td>AVLTree</td><td>isBalanced()</td><td>setup2</td><td>none</td><td>True, because the tree created in the stage2 is balanced, by the criteria of its rolling factor</td></tr>
</table>

<table>
<tr><td colspan="5"><b>Test goal: Verify if the method addNode is able to set new nodes in the tree, respecting the criterion of the balance factor of the AVL tree</b></td></tr>
<tr><th>Class</th><th>Method</th><th>Stage</th><th>Input values</th><th>Result</th></tr>
<tr><td>AVLTree</td><td>addNode()</td><td>setup1</td><td>element = 2
key = 11</td><td>the element of the node searched is the same as the one of the node added to the tree</td></tr>
</table>

<table>
<tr><td colspan="5"><b>Test goal: Verify if the method delete is able to delete correctly an specific node</b></td></tr>
<tr><th>Class</th><th>Method</th><th>Stage</th><th>Input values</th><th>Result</th></tr>
<tr><td>AVLTree</td><td>delete()</td><td>setup2</td><td>element = 10</td><td>9, which is the new node with the highest value of the tree, because the node with the value 10 was deleted and it used to be the highest one.</td></tr>
</table>

<table>
<tr><td colspan="5"><b>Test goal: Verify if the method search returns the correct value that want to be searched</b></td></tr>
<tr><th>Class</th><th>Method</th><th>Stage</th><th>Input values</th><th>Result</th></tr>
<tr><td>AVLTree</td><td>search()</td><td>setup1</td><td>key = 6</td><td>2, which is the value of the node with key 6. That means that the search was made successfully</td></tr>
</table>

<table>
<tr><th colspan="5">Test goal: Check if the tree returns the lowest value expected before setting the tree in an stage</th></tr>
<tr><th>Class</th><th>Method</th><th>Stage</th><th>Input values</th><th>Result</th></tr>
<tr><td>AVLTree</td><td>min()</td><td>setup1</td><td>none</td><td>1, which belongs to the lowest key in the tree set in the stage1</td></tr>
</table>

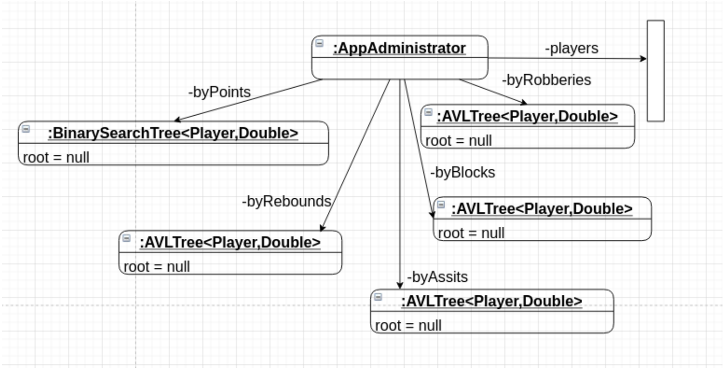
<table>
<tr><th colspan="5">Test goal: Check if the tree returns the highest value expected before setting the tree in an stage</th></tr>
<tr><th>Class</th><th>Method</th><th>Stage</th><th>Input values</th><th>Result</th></tr>
<tr><td>AVLTree</td><td>max()</td><td>setup1</td><td>none</td><td>8, which belongs to the highest key in the tree set in the stage1</td></tr>
</table>

<table>
<tr><th colspan="5">Test goal: Verify if the the method is able to calcule a correct value for the tree’s height set in the stage</th></tr>
<tr><th>Class</th><th>Method</th><th>Stage</th><th>Input values</th><th>Result</th></tr>
<tr><td>AVLTree</td><td>heigth()</td><td>setup1</td><td>none</td><td>3, which is the correct value for the height of the tree set in the stage1</td></tr>
</table>

<table>
<tr><th colspan="5">Test goal: Verify if the the method is able to calcule a correct value for tree’s rolling factor set in the stage</th></tr>
<tr><th>Class</th><th>Method</th><th>Stage</th><th>Input values</th><th>Result</th></tr>
<tr><td>AVLTree</td><td>getRollingfactor()</td><td>setup1</td><td>none</td><td>0, that represents the difference between the high of the right subtree and the left subtree</td></tr>
</table>

Model classes

AppAdministrator class:

Name	Class	Stage
setup1	AppAdministrator	 :AppAdministrator -players -byRobberies -byPoints :AVLTree<Player,Double> root = null :BinarySearchTree<Player,Double> root = null -byBlocks -byRebounds :AVLTree<Player,Double> root = null :AVLTree<Player,Double> root = null -byAssits :AVLTree<Player,Double> root = null

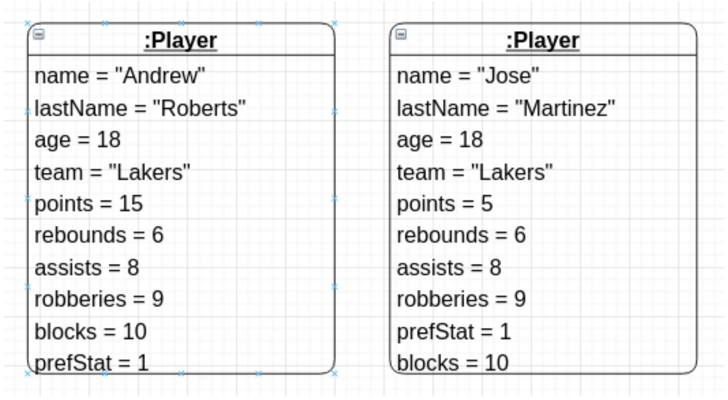
Test goal: Verify is the method is able to add a player to the list with the correct value of its attributes				
Class	**Method**	**Stage**	**Input values**	**Result**

AppAdministrator	addPlayer()	setup1	name = "Jose",
			lastName = "Martinez"
			age = 18
			team = "Leakers"
			points = 5
			rebounds = 6
			assits = 8
			robberies = 9
			blocks = 10
				True, which means that the player was added correctly to the list of players by checking if the name of the first element of the list is the same as the name of the player added. The list now has a size of 1

Test goal: Check if the method is able to import information from external data				
Class	**Method**	**Stage**	**Input values**	**Result**
AppAdministrator	imporPlayerrs()	setup1	fileName = "src/data/Dataset.csv"	The method was able to import the information from the database and the list of the class administrator has a new size of 200.000 with all the information of the players and all its atributes

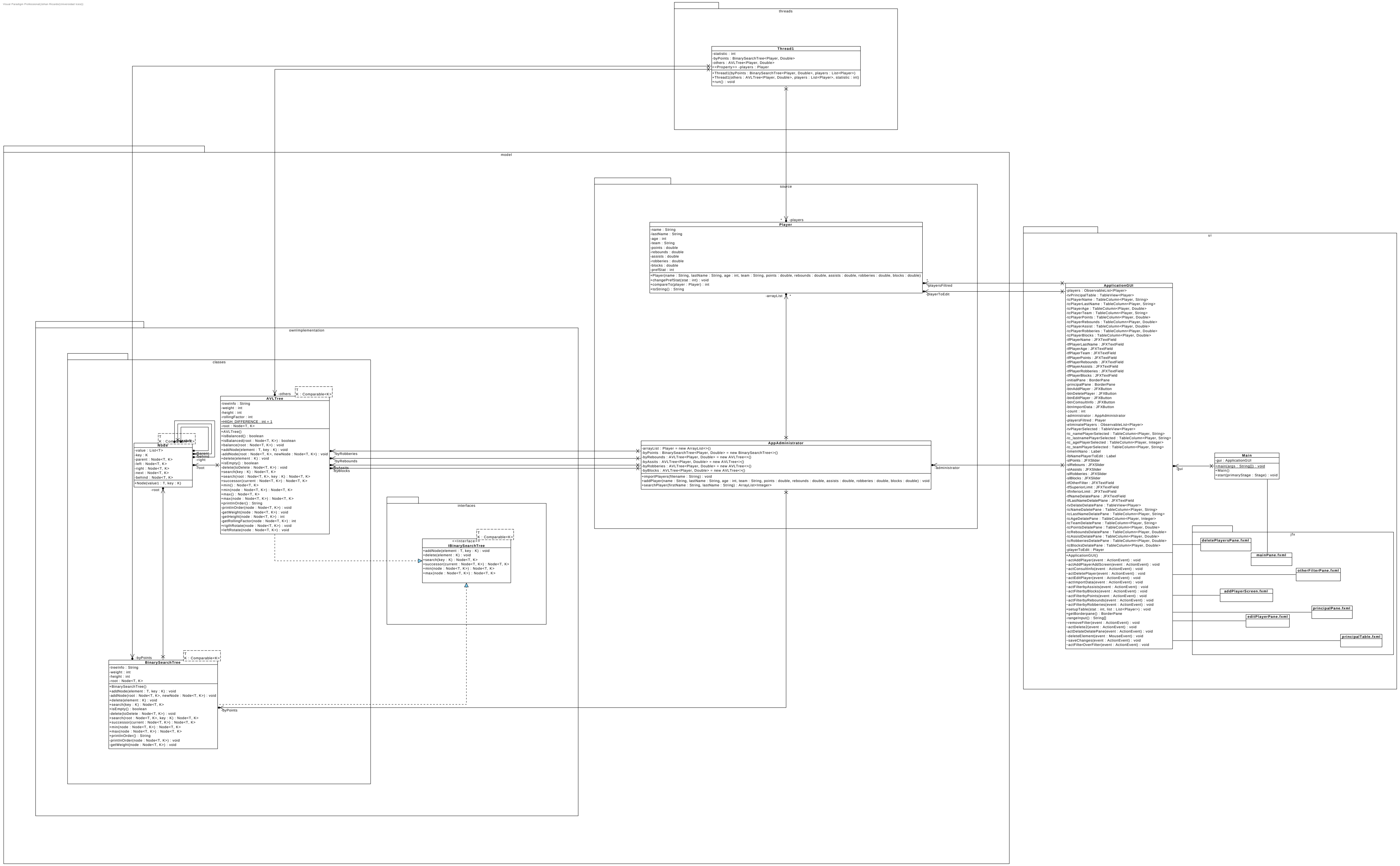
Player class:

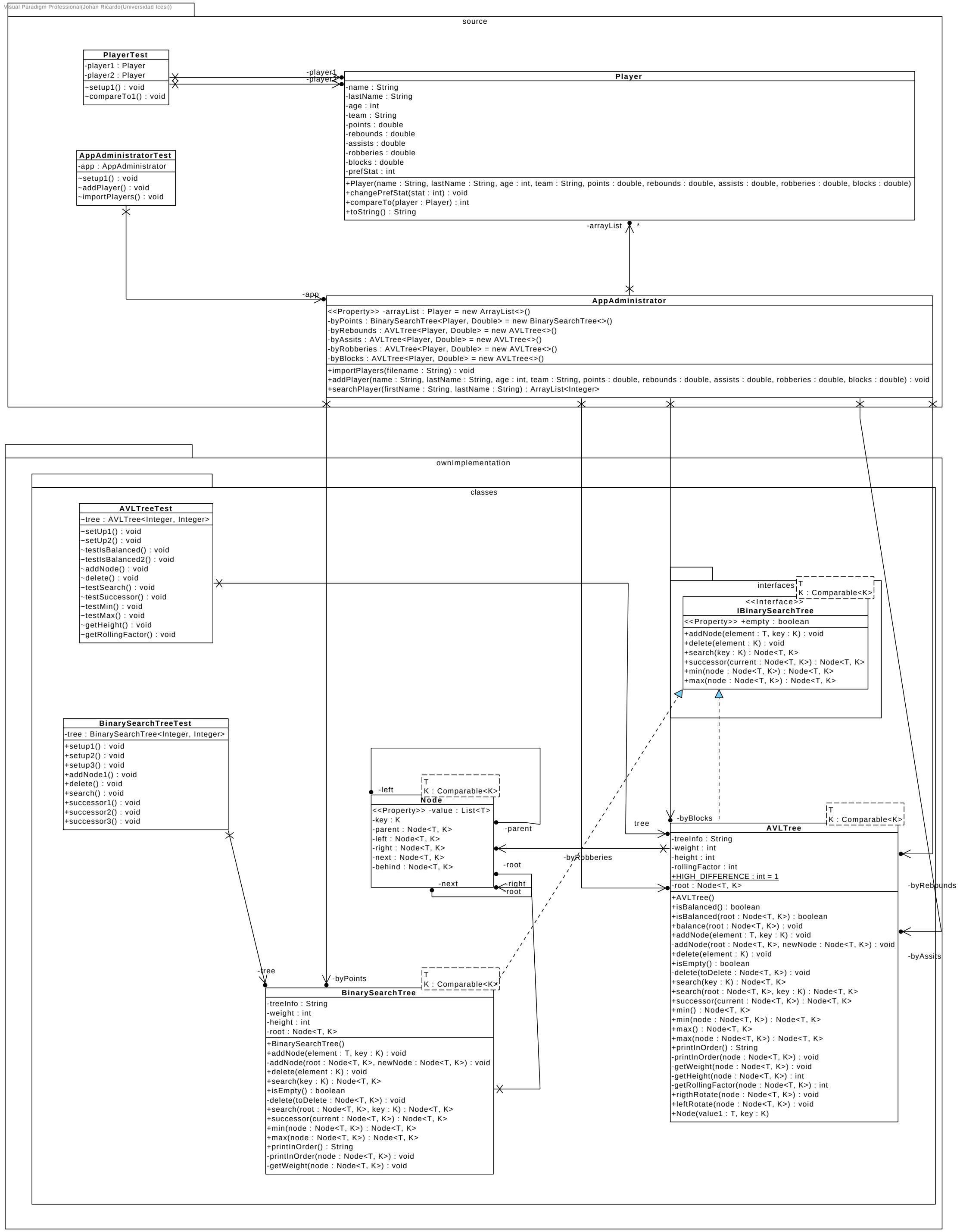
Name	Class	Stage

setup1	Player	
		:Player
		name = "Andrew"
		lastName = "Roberts"
		age = 18
		team = "Lakers"
		points = 15
		rebounds = 6
		assists = 8
		robberies = 9
		blocks = 10
		prefStat = 1
		:Player
		name = "Jose"
		lastName = "Martinez"
		age = 18
		team = "Lakers"
		points = 5
		rebounds = 6
		assists = 8
		robberies = 9
		prefStat = 1
		blocks = 10

Test goal: Verify if the overridden method comparteTo works correctly				
Class	**Method**	**Stage**	**Input values**	**Result**
Player	compareTo()	setup1	Player = player2	The method returns a value higher than 0, which represents that the player2 is higher than the player.

				That means that the overridden method works correctly.


threads
Thread1
-statistic : int
-byPoints : BinarySearchTree<Player, Double>
-others : AVLTree<Player, Double>
<<Property>> -players : Player
+Thread1(byPoints : BinarySearchTree<Player, Double>, players : List<Player>)
+Thread1(others : AVLTree<Player, Double>, players : List<Player>, statistic : int)
+run() : void
model
source
*
-players
Player
-name : String
-lastName : String
-age : int
-team : String
-points : double
-rebounds : double
-assists : double
-robberies : double
-blocks : double
-prefStat : int
+Player(name : String, lastName : String, age : int, team : String, points : double, rebounds : double, assists : double, robberies : double, blocks : double)
+changePrefStat(stat : int) : void
+compareTo(player : Player) : int
+toString() : String
*
-playersFiltred
-playerToEdit
-arrayList
*
AppAdministrator
-arrayList : Player = new ArrayList<>()
-byPoints : BinarySearchTree<Player, Double> = new BinarySearchTree<>()
-byRebounds : AVLTree<Player, Double> = new AVLTree<>()
-byAssits : AVLTree<Player, Double> = new AVLTree<>()
-byRobberies : AVLTree<Player, Double> = new AVLTree<>()
-byBlocks : AVLTree<Player, Double> = new AVLTree<>()
+importPlayers(filename : String) : void
+addPlayer(name : String, lastName : String, age : int, team : String, points : double, rebounds : double, assists : double, robberies : double, blocks : double) : void
+searchPlayer(firstName : String, lastName : String) : ArrayList<Integer>
-administrator
ownImplementation
classes
-others
T
K : Comparable<K>
AVLTree
-treeInfo : String
-weight : int
-height : int
-rollingFactor : int
+HIGH_DIFFERENCE : int = 1
-root : Node<T, K>
+AVLTree()
+isBalanced() : boolean
+isBalanced(root : Node<T, K>) : boolean
+balance(root : Node<T, K>) : void
+addNode(element : T, key : K) : void
-addNode(root : Node<T, K>, newNode : Node<T, K>) : void
+delete(element : K) : void
+isEmpty() : boolean
-delete(toDelete : Node<T, K>) : void
+search(key : K) : Node<T, K>
+search(root : Node<T, K>, key : K) : Node<T, K>
+successor(current : Node<T, K>) : Node<T, K>
+min() : Node<T, K>
+min(node : Node<T, K>) : Node<T, K>
+max() : Node<T, K>
+max(node : Node<T, K>) : Node<T, K>
+printInOrder() : String
-printInOrder(node : Node<T, K>) : void
-getWeight(node : Node<T, K>) : void
-getHeight(node : Node<T, K>) : int
-getRollingFactor(node : Node<T, K>) : int
+rigthRotate(node : Node<T, K>) : void
+leftRotate(node : Node<T, K>) : void
-byRobberies
-byRebounds
-byAssits
-byBlocks
T
K : Comparable<K>
-next
-left
Node
-value : List<T>
-key : K
-parent : Node<T, K>
-left : Node<T, K>
-right : Node<T, K>
-next : Node<T, K>
-behind : Node<T, K>
+Node(value1 : T, key : K)
-parent
-behind
-right
-root
-root
interfaces
T
K : Comparable<K>
<<Interface>>
IBinarySearchTree
+addNode(element : T, key : K) : void
+delete(element : K) : void
+search(key : K) : Node<T, K>
+successor(current : Node<T, K>) : Node<T, K>
+min(node : Node<T, K>) : Node<T, K>
+max(node : Node<T, K>) : Node<T, K>
-byPoints
T
K : Comparable<K>
BinarySearchTree
-treeInfo : String
-weight : int
-height : int
-root : Node<T, K>
+BinarySearchTree()
+addNode(element : T, key : K) : void
-addNode(root : Node<T, K>, newNode : Node<T, K>) : void
+delete(element : K) : void
+search(key : K) : Node<T, K>
+isEmpty() : boolean
-delete(toDelete : Node<T, K>) : void
+search(root : Node<T, K>, key : K) : Node<T, K>
+successor(current : Node<T, K>) : Node<T, K>
+min(node : Node<T, K>) : Node<T, K>
+max(node : Node<T, K>) : Node<T, K>
+printInOrder() : String
-printInOrder(node : Node<T, K>) : void
-getWeight(node : Node<T, K>) : void
-byPoints
ui
ApplicationGUI
-players : ObservableList<Player>
-tvPrincipalTable : TableView<Player>
-tcPlayerName : TableColumn<Player, String>
-tcPlayerLastName : TableColumn<Player, String>
-tcPlayerAge : TableColumn<Player, Double>
-tcPlayerTeam : TableColumn<Player, String>
-tcPlayerPoints : TableColumn<Player, Double>
-tcPlayerRebounds : TableColumn<Player, Double>
-tcPlayerAssist : TableColumn<Player, Double>
-tcPlayerRobberies : TableColumn<Player, Double>
-tcPlayerBlocks : TableColumn<Player, Double>
-tfPlayerName : JFXTextField
-tfPlayerLastName : JFXTextField
-tfPlayerAge : JFXTextField
-tfPlayerTeam : JFXTextField
-tfPlayerPoints : JFXTextField
-tfPlayerRebounds : JFXTextField
-tfPlayerAssists : JFXTextField
-tfPlayerRobberies : JFXTextField
-tfPlayerBlocks : JFXTextField
-initialPane : BorderPane
-principalPane : BorderPane
-btnAddPlayer : JFXButton
-btnDeletePlayer : JFXButton
-btnEditPlayer : JFXButton
-btnComsultInfo : JFXButton
-btnImportData : JFXButton
-count : int
-administrator : AppAdministrator
-playersFiltred : Player
-eliminatePlayers : ObservableList<Player>
-tvPlayerSelected : TableView<Player>
-tc_namePlayerSelected : TableColumn<Player, String>
-tc_lastnamePlayerSelected : TableColumn<Player, String>
-tc_agePlayerSelected : TableColumn<Player, Integer>
-tc_teamPlayerSelected : TableColumn<Player, String>
-timeInNano : Label
-lbNamePlayerToEdit : Label
-slPoints : JFXSlider
-slRebouns : JFXSlider
-slAssists : JFXSlider
-slRobberies : JFXSlider
-slBlocks : JFXSlider
-tfOtherFilter : JFXTextField
-tfSuperiorLimit : JFXTextField
-tfInferiorLimit : JFXTextField
-tfNameDelatePane : JFXTextField
-tfLastNameDelatePlane : JFXTextField
-tvDelateDelatePane : TableView<Player>
-tcNameDaletePane : TableColumn<Player, String>
-tcLastNameDelatePane : TableColumn<Player, String>
-tcAgeDelatePane : TableColumn<Player, Integer>
-tcTeamDelatePane : TableColumn<Player, String>
-tcPointsDelatePane : TableColumn<Player, Double>
-tcReboundsDelatePane : TableColumn<Player, Double>
-tcAssistDelatePane : TableColumn<Player, Double>
-tcRobberiesDelatePane : TableColumn<Player, Double>
-tcBlocksDelatePane : TableColumn<Player, Double>
-playerToEdit : Player
+ApplicationGUI()
~actAddPlayer(event : ActionEvent) : void
~actAddPlayerAddScreen(event : ActionEvent) : void
~actConsultInfo(event : ActionEvent) : void
~actDeletePlayer(event : ActionEvent) : void
~actEditPlayer(event : ActionEvent) : void
~actImportData(event : ActionEvent) : void
~actFilterbyAssists(event : ActionEvent) : void
~actFilterbyBlocks(event : ActionEvent) : void
~actFilterbyPoints(event : ActionEvent) : void
~actFilterbyRebounds(event : ActionEvent) : void
~actFilterbyRobberies(event : ActionEvent) : void
+setupTable(stat : int, list : List<Player>) : void
+getBorderpane() : BorderPane
-rangeInput() : String[]
~removeFilter(event : ActionEvent) : void
~actDelete2(event : ActionEvent) : void
-actDelateDelatePane(event : ActionEvent) : void
~deleteElement(event : MouseEvent) : void
~saveChanges(event : ActionEvent) : void
~actFilterOverFilter(event : ActionEvent) : void
-gui
Main
-gui : ApplicationGUI
+main(args : String[]) : void
+Main()
+start(primaryStage : Stage) : void
jfx
deletePlayersPane.fxml
mainPane.fxml
otherFilterPane.fxml
addPlayerScreen.fxml
principalPane.fxml
editPlayerPane.fxml
principalTable.fxml


source
PlayerTest
-player1 : Player
-player2 : Player
~setup1() : void
~compareTo1() : void
-player1
-player2
Player
-name : String
-lastName : String
-age : int
-team : String
-points : double
-rebounds : double
-assists : double
-robberies : double
-blocks : double
-prefStat : int
+Player(name : String, lastName : String, age : int, team : String, points : double, rebounds : double, assists : double, robberies : double, blocks : double)
+changePrefStat(stat : int) : void
+compareTo(player : Player) : int
+toString() : String
AppAdministratorTest
-app : AppAdministrator
~setup1() : void
~addPlayer() : void
~importPlayers() : void
-arrayList
*
-app
AppAdministrator
<<Property>> -arrayList : Player = new ArrayList<>()
-byPoints : BinarySearchTree<Player, Double> = new BinarySearchTree<>()
-byRebounds : AVLTree<Player, Double> = new AVLTree<>()
-byAssits : AVLTree<Player, Double> = new AVLTree<>()
-byRobberies : AVLTree<Player, Double> = new AVLTree<>()
-byBlocks : AVLTree<Player, Double> = new AVLTree<>()
+importPlayers(filename : String) : void
+addPlayer(name : String, lastName : String, age : int, team : String, points : double, rebounds : double, assists : double, robberies : double, blocks : double) : void
+searchPlayer(firstName : String, lastName : String) : ArrayList<Integer>
ownImplementation
classes
AVLTreeTest
~tree : AVLTree<Integer, Integer>
~setUp1() : void
~setUp2() : void
~testIsBalanced() : void
~testIsBalanced2() : void
~addNode() : void
~delete() : void
~testSearch() : void
~testSuccessor() : void
~testMin() : void
~testMax() : void
~getHeight() : void
~getRollingFactor() : void
interfaces
T
K : Comparable<K>
<<Interface>>
IBinarySearchTree
<<Property>> +empty : boolean
+addNode(element : T, key : K) : void
+delete(element : K) : void
+search(key : K) : Node<T, K>
+successor(current : Node<T, K>) : Node<T, K>
+min(node : Node<T, K>) : Node<T, K>
+max(node : Node<T, K>) : Node<T, K>
BinarySearchTreeTest
-tree : BinarySearchTree<Integer, Integer>
+setup1() : void
+setup2() : void
+setup3() : void
+addNode1() : void
+delete() : void
+search() : void
+successor1() : void
+successor2() : void
+successor3() : void
-left
T
K : Comparable<K>
Node
<<Property>> -value : List<T>
-key : K
-parent : Node<T, K>
-left : Node<T, K>
-right : Node<T, K>
-next : Node<T, K>
-behind : Node<T, K>
-parent
-root
-next
-right
-root
-byRobberies
tree
-byBlocks
T
K : Comparable<K>
AVLTree
-treeInfo : String
-weight : int
-height : int
-rollingFactor : int
+HIGH_DIFFERENCE : int = 1
-root : Node<T, K>
+AVLTree()
+isBalanced() : boolean
+isBalanced(root : Node<T, K>) : boolean
+balance(root : Node<T, K>) : void
+addNode(element : T, key : K) : void
-addNode(root : Node<T, K>, newNode : Node<T, K>) : void
+delete(element : K) : void
+isEmpty() : boolean
-delete(toDelete : Node<T, K>) : void
+search(key : K) : Node<T, K>
+search(root : Node<T, K>, key : K) : Node<T, K>
+successor(current : Node<T, K>) : Node<T, K>
+min() : Node<T, K>
+min(node : Node<T, K>) : Node<T, K>
+max() : Node<T, K>
+max(node : Node<T, K>) : Node<T, K>
+printInOrder() : String
-printInOrder(node : Node<T, K>) : void
-getWeight(node : Node<T, K>) : void
-getHeight(node : Node<T, K>) : int
-getRollingFactor(node : Node<T, K>) : int
+rigthRotate(node : Node<T, K>) : void
+leftRotate(node : Node<T, K>) : void
+Node(value1 : T, key : K)
-byRebounds
-byAssits
-tree
-byPoints
T
K : Comparable<K>
BinarySearchTree
-treeInfo : String
-weight : int
-height : int
-root : Node<T, K>
+BinarySearchTree()
+addNode(element : T, key : K) : void
-addNode(root : Node<T, K>, newNode : Node<T, K>) : void
+delete(element : K) : void
+search(key : K) : Node<T, K>
+isEmpty() : boolean
-delete(toDelete : Node<T, K>) : void
+search(root : Node<T, K>, key : K) : Node<T, K>
+successor(current : Node<T, K>) : Node<T, K>
+min(node : Node<T, K>) : Node<T, K>
+max(node : Node<T, K>) : Node<T, K>
+printInOrder() : String
-printInOrder(node : Node<T, K>) : void
-getWeight(node : Node<T, K>) : void