FUNCTIONAL REQUIREMENTS, TEST DESIGN CASES AND CLASS DIAGRAM

MEMBERS:

Gabriel Suárez - **A00368589 SIS**

Alejandro Varela - **A00369019 TEL & SIS**

Luis Alfonso Murcia Hernández - **A00369008 TEL**

TEACHER:

Anibal Sosa Aguirre

ALGORITHMS AND DATA STRUCTURES

ICESI UNIVERSITY
2021-2

FUNCTIONAL REQUIREMENTS

Functional Requirements: Integrative Task 2

The program must be able to:

RF1: ***Allow*** data entry, either in bulk with .csv files or through an interface

RF2: ***Delete*** data in case it is already unnecessary in any way within the data collection tool

RF3: ***Modify*** data in case it has been exceeded in any way or an error was made at the time of entry.

RF4: ***Make*** player queries using the statistical categories included as search criteria, find those players who have scored 10 points per game, or those with more than 20 rebounds per game.

RF5: ***Include*** the data per player of the following items: name, age, team and 5 statistics points per game, rebounds per game, assists per game, steals per game, blocks per game

RF6: ***Access*** efficiently for the performance of the application due to the large amount of data that the application must store.

RF7: ***Retrieve*** players according to the selected search category and the value given for it, it must be considered that queries are not necessarily requested where the attribute satisfies an equality. The search criteria can be any of the statistical attributes, but for four of them the search must be very fast

RF8: ***Show*** the time it takes to make a query.

RF9: ***Allow*** customer to search on two statistical criteria using ABB as structure for index management

RF10*: Execute* player queries according to all criteria, but only four of them (on statistical attributes) should be efficient.

TEST DESIGN CASES

Scenarios:

Name	Class	Scenary
setUpEscenary1	BSTree	BSTree about points, Empty tree
setUpEscenary2	BSTree	BSTree about points, An object null
setUpEscenary1	RBTree	Empty
setUpEscenary1	AVLTree	AVLTree, about points with three players
setUpEscenary1	model	Three players

Test Objective: Go through the whole tree in order and return the value.

Class	Method	Scenary	Input	Output
BSTree	inOrder()	setupScenary 1	Player player1= new Player("David",28,"Chicago Bulls", 345); Player player2= new Player("Jacobo",20,"Chicago Bulls", 118); Player player3= new Player("Juan",24,"Chicago Bulls", 403);	118 345 403

Test Objective: Go through the whole tree in order and return the order.

Class	Method	Scenary	Input	Output
BSTree	inOrder()	setupScenary 2	Player player1= null; Player player2= new Player("Jacobo",20,"Chicago Bulls", 118); Player player3= new Player("Juan",24,"Chicago Bulls", 203);	118 403

Test Objective: Search one element into the tree and return the value.

Class	Method	Scenary	Input	Output
BSTree	Search()	setupScenary 1	Player player1= new Player("David",28,"Chicago Bulls", 345); Player player2= new Player("Jacobo",20,"Chicago Bulls", 118); Player player3= new Player("Juan",24,"Chicago Bulls", 203);	345, player1 118, player2 203, player3

Test Objective: Search one element into the tree and return the value.

Class	Method	Scenary	Input	Output
BSTree	Search()	setupScenary 2	Player player1= new Player("David",28,"Chicago Bulls", 345); Player player2= new Player("Jacobo",20,"Chicago Bulls", 118); Player player3 = null	345, player1 118, player2 Return null.

Test Objective: Search the minimum value in the whole tree.

Class	Method	Scenary	Input	Output
BSTree	mínimum()	setupScenary 1	Player player1= new Player("David",28,"Chicago Bulls", 345); Player player2= new Player("Jacobo",20,"Chicago Bulls", 118); Player player3= new Player("Juan",24,"Chicago Bulls", 203);	118

Test Objective: Search the minimum value in the whole tree.

Class	Method	Scenary	Input	Output
BSTree	mínimum()	setupScenary 2	200	null

Test Objective: Search the maximum value in the whole tree.

Class	Method	Scenary	Input	Output
BSTree	máximum()	setupScenary 1	Player player1= new Player("David",28,"Chicago Bulls", 345); Player player2= new Player("Jacobo",20,"Chicago Bulls", 118); Player player3= new Player("Juan",24,"Chicago Bulls", 403);	403

Test Objective: Search the maximum value in the whole tree.

Class	Method	Scenary	Input	Output
BSTree	máximum()	setupScenary 2	200	null

Test Objective: Find the succesor of a specific key

Class	Method	Scenary	Input	Output
BSTree	succesor()	setupScenary 1	345	403

Test Objective: Find the succesor of a specific key

Class	Method	Scenary	Input	Output
BSTree	succesor()	setupScenary 2	200	null

Test Objective: Insert one element in the tree

Class	Method	Scenary	Input	Output
BSTree	add()	setupScenary 1	Player player1= new Player("David",28,"Chicago Bulls", 345);	The player1 was added successfully.
			Player player2= new Player("Jacobo",20,"Chicago Bulls", 118);	The player2 was added successfully.
			Player player3= new Player("Juan",24,"Chicago Bulls", 403);	The player3 was added successfully.

Test Objective: Insert one element in the tree

Class	Method	Scenary	Input	Output
BSTree	add()	setupScenary 2	Player player1= new Player("David",28,"Chicago Bulls", 345);	The player1 was added successfully.
			Player player2= new Player("Jacobo",20,"Chicago Bulls", 118);	The player2 was added successfully.
			Player player3= null.	The player3 was not added.

Test Objective:

Class	Method	Scenary	Input	Output
BSTree	delete()	setupScenary 1	118	The player 2 was deleted successfully.

Test Objective:

Class	Method	Scenary	Input	Output
BSTree	delete()	setupScenary 2	Null	Not is possible delete on element null.

Test Objective:

Class	Method	Scenary	Input	Output
RBTree	isRed()	setupScenary 1	3 2 1	True False True

Test Objective:

Class	Method	Scenary	Input	Output
RBTree	add()	setupScenary 1	3 2 1 Add 1, add 2, add 3	2 1 , 3

Test Objective:

Class	Method	Scenary	Input	Output
RBTree	delete()	setupScenary 1	3 2 1 Delete 2	1 0, 3

Test Objective: Add one element in the AVLTree and after rebalancing the same tree

Class	Method	Scenary	Input	Output
AVLTree	add()	setupScenary 1	3 2 1 Add 1, add 2, add 3	2 1 , 3

Test Objective: Delete one element in the AVLTree and after rebalancing the same tree

Class	Method	Scenary	Input	Output
AVLTree	delete()	setupScenary 1	3 2 1 0 Delete 2	1 0, 3

Test Objective: Try to filter the data for one defined parameter

Class	Method	Scenary	Input	Output
FibaDataCenter	filterData()	setupScenary 1	1, 100, 350	Player1, player2

Test Objective: Show the parameters, data size, numbers of teams and average age in the application.

Class	Method	Scenary	Input	Output
FibaDataCenter	updateIndicators()	setupScenary 1	118 345 403	3, 1, 24

CLASS DIAGRAM

FibaController
-mainPane : Pane
-btnCloseLogin : Circle
-btnMinimize : Circle
-palette : JFXToggleButton
-tblPlayers : Text
-lblTeams : Text
-lblAge : Text
-tblPlayers : TableView<Player>
-tblcPlayer : TableColumn<String, Player>
-tblcAge : TableColumn<Integer, Player>
-tblcTeam : TableColumn<String, Player>
-tblcPoints : TableColumn<String, Player>
-tblcAsists : TableColumn<Integer, Player>
-tblcBlocks : TableColumn<Integer, Player>
-tblcRebounds : TableColumn<Integer, Player>
-tblcSteals : TableColumn<Integer, Player>
-tblcActions : TableColumn<Player, String>
-lblTime : Label
+pBar : JFXProgressBar
-assists : JFXSlider
-blocks : JFXSlider
-rebounds : JFXSlider
-steals : JFXSlider
-lblAssits : Label
-lblBlocks : Label
-lblRebounds : Label
-lblSteals : Label
-txtName : JFXTextField
-txtTeam : JFXTextField
-txtAge : JFXTextField
-txtPoints : JFXTextField
-cancel : JFXButton
-save : JFXButton
-edit : JFXButton
-lblDialogName : Label
-txtUntil : JFXTextField
-txtSince : JFXTextField
-modal : Stage
-searchType : int
-pFiba : FibaDataCenter
-selected : Player
-txtLinear : JFXTextField
+FibaController(pFiba : FibaDataCenter)
+handleMouseClick(event : MouseEvent) : void
-getWindow() : Stage
+changePalette(event : ActionEvent) : void
-loadModal(route : Route) : Stage
+exportData(event : MouseEvent) : void
-fileChooser() : File
+importData(event : MouseEvent) : void
+newPlayer(event : ActionEvent) : void
+cancel(event : ActionEvent) : void
-initDialog(type : int, lbl : String) : void
+pointSearch(event : ActionEvent) : void
+assistSearch(event : ActionEvent) : void
+blockSearch(event : ActionEvent) : void
+reboundSearch(event : ActionEvent) : void
+stealSearh(event : ActionEvent) : void
+search(event : ActionEvent) : void
+onTablePlayers(data : ArrayList<Player>) : void
-renderShelveActions() : void
+refresh(event : MouseEvent) : void
-updateLabels() : void
-validateFileds() : boolean
+editPlayer(event : ActionEvent) : void
+savePlayer(event : ActionEvent) : void

-fController

Observer
-fController : FibaController
-obs : Observable
+Observer(fController : FibaController, fDataCenter : FibaDataCenter, br : BufferedReader)
+init() : void
+oninit() : void
+onFinish() : void

-fibaController

Main
-RENDER_COMPONENTES : int = 4000
-fibaController : FibaController
+Main()
+main(args : String[]) : void
+start(primaryStage : Stage) : void
+init() : void

-obs

Observable
-br : BufferedReader
-fDataCenter : FibaDataCenter
<<Property>> -listener : IObserver
+Observable(fDataCenter : FibaDataCenter, br : BufferedReader)
+importData() : void
+run() : void

<<Interface>>
IObserver
+onInit() : void
+onFinish() : void

-listener

FibaDataCenterTest
-players : Player[]
-fibaDataCenter : FibaDataCenter = new FibaDataCenter()
-getPlayers() : void
+setUpScenary1() : ITree<Integer, Player>
+filterData() : void
+updateIndicators() : void

-selected
-players
-players

Player
-name : String
-age : int
-team : String
-point : int
-bounces : int
-assists : int
-steals : int
-blocks : int
+Player(name : String, age : int, team : String, point : int, assists : int, blocks : int, bounces : int, steals : int)
+toString() : String

-players

BSTreeTest
-players : Player[]
-getPlayers() : void
+setUpScenary1() : ITree<Integer, Player>
+setUpScenary2() : ITree<Integer, Player>
+inOrder() : void
+inOrder2() : void
+search() : void
+search2() : void
+succesor() : void
+succesor2() : void
+add() : void
+add2() : void
+deletion() : void
+deletion2() : void

RBTreeTest
-players : Player[]
-getPlayers() : void
+setUpScenary1() : RBTree<Integer, Player>
+add() : void
+delete() : void

collections

AVLTreeTest
-players : Player[]
-getPlayers() : void
+setUpScenary1() : AVLTree<Integer, Player>
+add() : void
+delete() : void

K : Comparable<K>
V

<<Interface>>
ITree
+inOrder() : ArrayList<V>
+search(key : K, value : V) : BSNode<K, V>
+filter(filter : ArrayList<V>, key : K) : void
+sucessor(node) : BSNode<K, V>
+add(key : K, value : V) : BSNode<K, V>
+delete(key : K, value : V) : BSNode<K, V>

-trees

-fDataCenter

FibaDataCenter
-trees : ITree<Integer, Player>[]
-memory : Player
+FibaDataCenter()
-initTree() : void
+deletePlayer(py : Player) : void
-addPlayer(py : Player) : void
+filterData(searchType : int, since : int, until : int) : ArrayList<Player>
+updateIndicators(data : ArrayList<Player>) : int[]

-pFiba

Main
-RENDER_COMPONENTES : int = 4000
-fitaController: FitaController
-stage: Stage
+main(String[] : String[]) : void
+start(primaryStage : Stage) : void
+init() : void
Splash
-splash : Stage
+displaySplash(primaryStage : Stage) : void
+handleStateChange(notification)(info : StateChangeNotification) : void
ui
images
styles
screens
dashboard.fxml
display.fxml
header.fxml
mainFrame.fxml
player.fxml
fitaController
FitaController
-mainFrame : Pane
-btnClose/Login : Circle
-btnMinimize : Circle
-palette : JFXTogglleButton
-btnPlayers : Tool
-btnTeam : Tool
-btnAge : Tool
-tblPlayers : TableView<Player>
-tblcPlayer : TableColumn<String, Player>
-tblcAge : TableColumn<Integer, Player>
-tblcTeam : TableColumn<String, Player>
-tblcPoints : TableColumn<String, Player>
-tblcAssists : TableColumn<Integer, Player>
-tblcBlocks : TableColumn<Integer, Player>
-tblcRebounds : TableColumn<Integer, Player>
-tblcSteals : TableColumn<Integer, Player>
-tblcActions : TableColumn<Player, String>
-btnTime : Label
-txtNar : JFXProgressBar
-assists : JFXSlider
-rebounds : JFXSlider
-blocksSlc : JFXSlider
-steals : JFXSlider
-lblAssists : Label
-lblBlocks : Label
-lblRebounds : Label
-lblSteals : Label
-txtName : JFXTextField
-txtTeam : JFXTextField
-txtAge : JFXTextField
-txtPoints : JFXTextField
-cancel : JFXButton
-btnFile : JFXButton
-add : JFXButton
-lblDialogName : Label
-txtLimit : JFXTextField
-txtSince : JFXTextField
-txtUntil : JFXTextField
-searchType : int
-pfTree : FitaDataCenter
-selected : Player
-txtLinear : JFXTextField
+FitaController(pfTree : FitaDataCenter)
+handleMouseClick(event : MouseEvent) : void
+initialize() : void
+changePlate(event : ActionEvent) : void
-loadModel(route : Route) : Stage
-exportData(event : MouseEvent) : void
-fileChooser() : File
+importPlayers(event : MouseEvent) : void
+moveFile(event : ActionEvent) : void
+cancel(event : ActionEvent) : void
-infoDialog(type : int, fil : String) : void
+pointSearch(event : ActionEvent) : void
+assistSearch(event : ActionEvent) : void
+blockSearch(event : ActionEvent) : void
+reboundSearch(event : ActionEvent) : void
+stealSearch(event : ActionEvent) : void
+search(event : ActionEvent) : void
+showTable(players : ObservableList<Player>) : void
-renderShelves(hcm)() : void
+refresh(event : MouseEvent) : void
-lpiSlideLabel(hi) : void
-addBalefFile(fil) : boolean
+addPlayer(event : ActionEvent) : void
+saveFile(event : ActionEvent) : void
-prepareFileyerEditor() : void
+searchLinear(event : ActionEvent) : void
Observer
#Controller : FitaController
#Ocs : Observable
+Observer(Controller : FitaController, fitaDataCenter : FitaDataCenter, br : BufferedReader)
+init() : void
+start() : void
+setFinish() : void
thread
Observable
-br : BufferedReader
-fitaCenter : FitaDataCenter
<<Property>> #_listener : IObserver
+Observable(fitaCenter : FitaDataCenter, br : BufferedReader)
+registerData() : void
+run() : void
fitaCtrl
fitaCenter
Player
-name : String
-age : int
-team : String
-point : int
-bounces : int
-rebounds : int
-steals : int
-blocks : int
-id : int
+Player(name : String, age : int, team : String, point : int, assists : int, blocks : int, bounces : int, steals : int)
+toString() : String
selected
players
players
FitaDataCenter
-trees : ITree<Integer, Player>[]
-memory : Player
+FitaDataCenter()
+setTree() : void
+deletePlayer(p1p : Player) : void
+addPlayer(p1p : Player) : void
+fillterDateSearch(Type : int, since : int, until : int) : ArrayList<Player>
+updateIndicators(data : ArrayList<Player>) : int[]
+getPlayers() : ArrayList<Player>
+add(p : Player) : void
+searchLinear(prop : String) : ArrayList<Player>
pfTree
medal
mainFrame
RBTree
-RED : boolean = true
-BLACK : boolean = false
-rbRoot : RBNode<K, V>
+RBTree()
-leftRotate(p : RBNode) : RBNode<K, V>
-rightRotate(p : RBNode) : RBNode<K, V>
-flipColors(node : RBNode) : void
-fixAfterInsertion(z : RBNode) : void
+add(key : K, value : V) : RBNode<K, V>
-addEnode : RBNode, key : K, value : V) : RBNode<K, V>
+inOrder() : ArrayList<V>
#inOrder(current : BSNode<K, V>) : void
-transplant(u : RBNode<K, V>, v : RBNode<K, V>) : void
-rightRotate(p : RBNode<K, V>) : RBNode<K, V>
-flipColors(node : RBNode<K, V>) : void
-isRed(node : BSNode<K, V>) : boolean
-addEnode : RBNode<K, V>, key : K, value : V) : RBNode<K, V>
rbTree
RBNode
-color : boolean
+RBNode(key : K, value : V)
+getColor() : boolean
rbRoot
ITree
<<interface>>
+inOrder() : ArrayList<V>
+search(key : K, value : V) : BSNode<K, V>
+fillterDate : ArrayList<V>(key : K) : void
+successor(node) : BSNode<K, V>
+add(key : K, value : V) : BSNode<K, V>
+delete(key : K, value : V) : BSNode<K, V>
+successor(node : BSNode<K, V>) : BSNode<K, V>
BSTree
-fnodes : ArrayList<V>
-froot : BSNode<K, V>
+inOrder() : ArrayList<V>
#inOrder(current : BSNode<K, V>) : void
+search(key : K, value : V) : BSNode<K, V>
#search(current : BSNode<K, V>, key : K) : BSNode<K, V>
#search(current : BSNode<K, V>, key : K, value : V) : BSNode<K, V>
+fillterDate(key : K) : ArrayList<V>
#fillterDate(ArrayList<V> current : BSNode<K, V>, key : K) : void
+successor(current : BSNode<K, V>) : BSNode<K, V>
+add(key : K, value : V) : BSNode<K, V>
#add(newNode : BSNode<K, V>) : BSNode<K, V>
+delete(key : K, value : V) : BSNode<K, V>
bsTree
BSNode
-key : K
-value : V
-parent : BSNode<K, V>
-right : BSNode<K, V>
-left : BSNode<K, V>
+BSNode(key : K, value : V)
node
AVLTree
-mezcla : int, b : int) : int
-height(node : AVLNode<K, V>) : int
-updateHeight(node : AVLNode<K, V>) : void
-leftRotate(p : AVLNode<K, V>) : void
-rightRotate(p : AVLNode<K, V>) : void
+add(key : K, value : V) : AVLNode<K, V>
#addEnode(key : K, value : V) : AVLNode<K, V>
+getBalanceFactor(node : AVLNode<K, V>) : int
+rebalance(node : AVLNode<K, V>, key : K) : void
avlTree
AVLNode
-height : int
+AVLNode(key : K, value : V)
<<Instrumentation>>
Route
-route : String
+Route(route : String)
SPLASH
MAINFRAME
LOGIN
DASH
PLAYERS
SEARCH
RBTreeTest
-players : Player[]
+getPlayers() : void
+setUpScenary1() : RBTree<Integer, Player>
+add() : void
+delete() : void
BSTreeTest
-players : Player[]
+getPlayers() : void
+setUpScenary1() : ITree<Integer, Player>
+setUpScenary2() : ITree<Integer, Player>
+inOrder() : void
+preOrder() : void
+search() : void
+add() : void
+successor() : void
+successor1() : void
+add() : void
+add2() : void
+delete() : void
+deletion2() : void
AVLTreeTest
-players : Player[]
+getPlayers() : void
+setUpScenary1() : AVLTree<Integer, Player>
+add() : void
+delete() : void