|  |
| --- |
| Procedure addPlayer(Player) |
|  |
| IF ContainsInPlayers(Player) THEN |
| ERROR "PLayer is already in the system" |
| END IF |
| ELSE |
| index <-- 0 |
| IF NonEmpty(availableIndexs) THEN |
| newIndex <-- Poll(availableIndex) |
|  |
| ELSE |
| IF ContainsKeyinPlayers(LASTINDEX) THEN |
|  |
| newIndex <-- PLayerAdded(LASTINDEX) |
| PutinPlayers(LASTINDEX,PLAYER+1) |
|  |
|  |
| ELSE |
| Put(LASTINDEX, 0) |
| END IF |
| END IF |
| END IF |
|  |
|  |
| Put(PlayerName, newIndex) |
| SavePLayer(PLayer, newIndex) |
|  |
| RBTreeADDPlayer(Player, newIndex) |
| AVLTreeADDPlayer(Player, newIndex) |
| BTSTreeADDPlayer(PLayer, newIndex) |
| PersistenceOff() |
|  |
| END Procedure |

# FIBA PSEUDOCODES

\

|  |
| --- |
| Procedure removePlayer (NamePlayer) |
| result <-- The name of the PLayer NamePLayer has been remove |
|  |
| IF playersAdded(NamePLayer) Then |
| player <-- null |
| TRY |
| player <-- getPlayer(PlayersAdded(NamePlayer)) |
|  |
| TRY END |
|  |
| CATCH (EXCEPTION) |
| result <-- ERROR APPEAR |
| CATCH END |
|  |
| AvailableIndexADD(PLayersAdded(namePlayer)) |
| playersADDEDRemove(NamePLayer) |
|  |
| AVLTreeRemovePlayer(NamePlayer) |
| BTSTreeRemovePlayer(NamePlayer) |
| RBTreeRemovePlayer(NamePlayer) |
|  |
| ELSE |
|  |
| result <-- The pLayer Name was not found |
| END IF |
|  |
|  |
| return result |
| END PROCEDURE |

|  |
| --- |
| Procedure modifyPlayerAtribute(PlayerName, attributeToChange,newValue) |
|  |
| result <-- DONE |
|  |
| IF PlayersAddedContain(PlayerName) Then |
|  |
| player <-- null |
|  |
| TRY |
| Player <-- getPlayer(PLayersAddedGet(PlayerName)) |
|  |
| Catch (Exception) |
|  |
| result <-- An Error Impeded to done that change |
|  |
| END CATCH |
|  |
| switch (attributeToChange) |
|  |
| CASE PLayerChangeName |
|  |
| PLayerSetName(newValue) |
| CASE PlayerChangeGender |
|  |
| PLayerSetGender(NewValueCharAt(0)) |
|  |
| CASE PlayerChangeAge |
| PLayerSetAge(IntegerParseInt(newValue)) |
|  |
| CASE PlayerChangeGamesPlayed |
|  |
| playerSetGamesPlayed(IntergerPaserInt(newValue)) |
|  |
| CASE PlayerChangeMinutesPlayed |
| playerSetMinutesPlayed(DoubleParseDouble(newValue)) |
|  |
| CASE PlayerChangeFieldGoalsPercentage |
| playerSetFieldGoalsPercentage(DoubleParseDouble(newValue)) |
|  |
| CASE PlayerChangeThreeThrowPercentage |
|  |
| PLayerSetFieldGoalsPercentage(DoubleParseDouble(newValue)) |
|  |
| CASE PlayerChangeFreeThrowPrecentage |
| playerSetFreeThrowPercentage(DoubleParseDouble(newValue)) |
|  |
| CASE PLayerChangePersonalFouls |
|  |
| PLayerSetPersonalFouls(IntegerParseInteger(newValue)) |
|  |
| CASE PlayerChangePlayerImpactEstimate |
| playerSetPLayerImpactEstimate(DoubleParseDOuble(newValue)) |
|  |
|  |
| CASE PlayerChangeOfensiveReboundPercentage |
|  |
| playerSetOffensiveReboundPercentage(DoubleparseDouble(newValue)) |
|  |
| CASE PlayerChangeTurnoverPercentage |
| playerSetTurnoverPercentage(DoubleParseDouble(newValue)) |
|  |
| IF attributeToChangeEquals(PLayerChangeFieldGoalsPercentage) Then |
|  |
| attributeToChangeEquals(PlayerChangeThreeThrwPercentage) |
|  |
| switch(attributeTochange) |
|  |
| Case PLayerChangeFieldGoalsPercentage |
| RBTreeModifyValue(player, newvalue, RedBlackNodeFieldGoalsPercentage ) |
| BTSTreeModifyValue(player, newValue, BTSNodeThreePointFieldGoalsPercentage ) |
|  |
| Case PlayerChangeThreeThrowPercentage |
|  |
| RBTreeModifyValue(player, newValue, RedBlackNodeThreePointFieldGoalPercentage) |
| BTSTreeModifyValue(player, newValue, BTSNodeThreePointFieldGoalsPercentage) |
|  |
| ELSE IF attributeToChangeEquals(PLayerChangeFreeThrowPercentage) Then |
|  |
|  |
|  |
| switch(attributeTochange) |
|  |
| Case PLayerChangeFreeThrowPercentage |
| AVLTreeModifyValue(player, newvalue, AVLNodeFreeThrowPercentage ) |
| BTSTreeModifyValue(player, newValue, BTSNodeFreeThrowPercentage ) |
|  |
|  |
| Case PlayerChangePersonalFouls |
|  |
| AVLTreeModifyValue(player, newValue, AVLNodePersonalFouls) |
| BTSTreeModifyValue(player, newValue, BTSNodePersonalFouls) |
|  |
| try savePlayer(player, PLayerAddedGet(PlayerName)) |
|  |
| catch (Exception) |
|  |
|  |
| result <-- An IPException impeded to done that change |
|  |
| END TRY |
|  |
| ELSE |
| result <-- PLayer Does not has Added |
|  |
|  |
| END IF |
| END IF |
|  |
| return result |
| END PROCEDURE |

|  |
| --- |
| Procedure searchIntervalQuery ( dafaultSearch, initialValue, finalValue, itemtype) |
|  |
| result <-- ArrayListNuevo |
|  |
| itemArray <-- ArrayListNuevo |
|  |
| IF defaultSearch THEN |
|  |
| IF itemType == RedBlackNodeFieldGoalsPercentage THEN |
|  |
| itemType == RedBlackNodeThreePointFieldGoalsPercentage |
|  |
| valueInitial <-- DoubleParseDouble (InitialValue) |
| valueFinal <-- DoubleParseDouble( finalValue) |
| itemsArray <-- searchIntervalValues (RBTree, valueInitial, ValueFinal, itemType) |
|  |
| ELSE IF itemType ==AVLNodePersonalFouls THEN |
|  |
| valueInitial <-- IntegerParseInt(initialValue); |
| valueFinal <-- IntegerParseInt(finalValue); |
| itemitsArray <-- searchIntervalValues(AVlTree, valueInitial, valueFinal, itemType); |
|  |
|  |
| ELSE IF item==AVLNodeFreeThrowPercentage THEN |
|  |
| valueInitial <-- Double.parseDouble(initialValue); |
| valueFinal <-- Double.parseDouble(finalValue); |
| itemsArray <-- searchIntervalValues(AVlTree, valueInitial, valueFinal, itemType); |
|  |
| END IF |
| END IF |
|  |
| ELSE |
| IF itemType == BTSNodeFIELDGOALSPERCENTAGE OR itemType == BTSNodeTHREEPOINTFIELDGOALSPERCENTAGE |
| OR itemType == BTSNodeTHREEPOINTFIELDGOALSPERCENTAGE THEN |
|  |
| valueInitial <-- DoubleParseDouble(initialValue); |
| valueFinal <-- DoubleParseDouble(finalValue); |
| itemsArray <-- searchIntervalValues(BTSTree, valueInitial, valueFinal, itemType); |
|  |
|  |
| ELSE IF itemType == BTSNodePersonalFouls THEN |
|  |
| valueInitial <-- IntegerParseInt(initialValue); |
| valueFinal <-- IntegerParseInt(finalValue); |
| itemsArray <-- searchIntervalValues(BTSTree, valueInitial, valueFinal, itemType); |
|  |
|  |
| END IF |
|  |
| TRY |
| result <-- generateReport(itemsArray) |
|  |
| catch (Exception) Then |
| Exception |
|  |
| END TRY |
|  |
|  |
| return result |
|  |
|  |
| END PROCEDURE |

|  |
| --- |
| Procedure searchIntervalValues (Tree, initialValue, finalValuable, itemType) |
|  |
| result <-- ArrayListNew |
|  |
| IF Tree Instanceof RedBlackTree THEN |
|  |
| result <-- RBTreeSearchRange (initialValue, finalValue, itemType) |
|  |
|  |
| ELSE IF tree Instanceof AVLTree THEN |
|  |
| result <-- AVLTreeSearchRange(initialValue, FinalValue, itemType) |
|  |
|  |
| ELSE IF tree Instanceof BinarySearchTree Then |
|  |
| result <-- BTSTreeSearchRange( initialValue, finalValue, itemType) |
|  |
|  |
| END IF |
|  |
| return result |
| END PROCEDURE |
|  |

|  |
| --- |
| Procedure searchValueQuery ( defaultSearch, value,itemType ) |
|  |
| result <-- ArrayListNew |
|  |
| IF defaultSearch Then |
|  |
| IF itemType == RedBlackNodeFieldGoalsPercentage THEN |
|  |
| value1 <-- DoubleParseDouble(value) |
| itemsArray <-- searchValue(RBTree, value1, itemType) |
|  |
|  |
|  |
| ELSE IF itemType== AVLNodePersonalFouls THEN |
|  |
| int value1 <-- IntegerParseInt(value); |
| itemsArray <-- searchValue(AVlTree, value1, itemType); |
|  |
| ELSE IF itemType == AVLNodeFREETHROWPERCENTAGE THEN |
|  |
| value1 <-- Double.parseDouble(value); |
| itemsArray <-- searchValue(AVlTree, value1, itemType); |
|  |
| END IF |
|  |
| ELSE |
| IF itemType == BTSNodeFIELDGOALSPERCENTAGE OR itemType == BTSNodeTHREEPOINTFIELDGOALSPERCENTAGE |
| OR itemType == BTSNodeTHREEPOINTFIELDGOALSPERCENTAGE THEN |
|  |
| value1 <-- DoubleparseDouble(value); |
| itemsArray <-- searchValue(BTSTree, value1, itemType); |
|  |
| ELSE IF itemType == BTSNodePERSONALFOULS THEN |
|  |
| value1 <-- IntegerParseInt(value); |
| itemsArray <-- searchValue(BTSTree, value1, itemType); |
|  |
|  |
|  |
| END IF |
|  |
| TRY |
| result <-- generateReport(itemsArray); |
|  |
| CATCH(EXCEPTION) |
|  |
| ERROR |
|  |
| END TRY |
|  |
| return result |
|  |
| END PROCEDURE |

|  |
| --- |
| Procedure searchValue (Tree, Value, ItemType) |
|  |
| result <-- ArrayListNew |
|  |
| IF tree instanceof RedBlackTree THEN |
|  |
| result <-- RBTreeSearchValue(value, itemType) |
|  |
| ELSE IF tree Instanceof AVLTree THEN |
|  |
| result <-- AVLTreeSearchValue(value, itemType) |
|  |
| ELSE IF tree Instanceof BinarySearchTree THEN |
|  |
| result <-- BTSTreeSearchValue(value, itemType) |
|  |
| END IF |
|  |
| retunr result |
|  |
| END PROCEDURE |

|  |
| --- |
| Procedure searchPlayer (Name) |
|  |
| Player <-- null |
|  |
| IF PlayerAddedCointains(Name) THEN |
|  |
| Player <-- getPlayer(PlayerAddedsget(Name)) |
|  |
| ELSE |
|  |
| PlayerNotFoundException <-- The PLayer was not found |
|  |
|  |
| END IF |
|  |
| retunr PLayer |
|  |
| END PROCEDURE |

# RED AND BLACK TREE

|  |
| --- |
| Procedure addPLayerItems (PLayer, txtIndex) |
|  |
| fieldGoalsPercentageItem <-- Item<Double>New(playerGetFieldGoalsPercentage(), txtIndex); |
| threePointsFieldItem <-- Item<Double>New (playerGetThreePointsFieldPercentage(), txtIndex); |
|  |
| node <-- NodeNew<>(fieldGoalsPercentageItem, RedBlackNode.RED\_NODE); |
| r2 <-- null; |
|  |
| IF RootFieldGoalsIndex == null THEN |
|  |
| RootFieldGoalsIndex <-- node; |
| RootFieldGoalsIndexSetColor(RedBlackNodeBLACKNODE); |
| ELSE |
| r2 <--RootFieldGoalsIndexInser(Node) |
| END IF |
|  |
| IF rootFieldGoalsIndex = r2 AND r2GetFather() == null THEN |
|  |
| r2 |
|  |
| ELSE |
| RootFieldGoalsIndex |
| END IF |
|  |
| node <-- (RedBlackNode<Double>) NodeNew<>(threePointsFieldItem, RedBlackNodeREDNODE); |
| r2 <-- null; |
|  |
| IF RootThreePointsIndex == null THEN |
|  |
| RootThreePointsIndex <-- node; |
| RootThreePointsIndexSetColor(RedBlackNodeBLACKNODE); |
| ELSE |
|  |
| r2 <--RootThreePointsIndexInsert(Node) |
|  |
| END IF |
|  |
| IF this.rootThreePointsIndex = r2 AND r2GetFather() == null THEN |
|  |
| r2 |
|  |
| ELSE |
| RootFieldGoalsIndex; |
|  |
|  |
| END IF |
|  |
| END PROCEDURE |

|  |
| --- |
| Procedure removePlayer (PLayer, txtIndex) |
|  |
| fieldGoalsPercentageItem <-- Item<Double>New(playerGetFieldGoalsPercentage(), txtIndex); |
| threePointsFieldItem <-- Item<Double>New (playerGetThreePointsFieldPercentage(), txtIndex); |
|  |
| node <-- NodeNew<>(fieldGoalsPercentageItem, RedBlackNode.RED\_NODE); |
| r2 <-- null; |
|  |
| IF RootFieldGoalsIndex == null THEN |
|  |
| EmptyTreeException The red black tree that contains field goals percentage index is empty |
|  |
| IF RootFieldGoalsIndexItemsHasSamePlayerAndValue(RootFieldGoalsIndexGetItem(), fieldGoalsPercentageItem) |
| AND RootFieldGoalsIndexRightSonLeaf() AND RootFieldGoalsIndexLeftSonLeaf() THEN |
|  |
|  |
| RootFieldGoalsIndex <-- null; |
|  |
| ELSE |
|  |
| cast <-- RootFieldGoalsIndexGetNodeWithEqualsValuesAndSamePlayer(fieldGoalsPercentageItem); |
| r2 <-- castRemoveRB(); |
| IF RootFieldGoalsIndex = r2 AND r2GetFather()==null |
| r2 |
| ELSE |
| RootFieldGoalsIndex |
|  |
| END IF |
|  |
|  |
| IF RootThreePointsIndex <-- null THEN |
|  |
| EmptyTreeException The red black tree that contains three point field goals percentage index is empty |
|  |
| IF RootThreePointsIndexItemsHasSamePlayerAndValue(RootThreePointsIndexGetItem(), threePointsFieldItem) |
| AND RootThreePointsIndexRightSonLeaf() AND RootThreePointsIndexLeftSonLeaf() THEN |
|  |
|  |
| RootThreePointsIndex <-- null; |
|  |
| ELSE |
|  |
|  |
| cast <-- RootThreePointsIndexGetNodeWithEqualsValuesAndSamePlayer(threePointsFieldItem); |
| r2 <-- castRemoveRB(); |
|  |
| IF RootThreePointsIndex <-- r2!=null AND r2GetFather()==null THEN |
|  |
| r2 |
|  |
| ELSE |
| RootThreePointsIndex |
|  |
|  |
| END IF |
| END IF |
| END IF |
| END IF |
|  |
|  |
| END PROCEDURE |