**Name Surname =** Gül Eda Aydemir

**Student Number =** 2015510013

**B-Tree**

In this lab session, you will practice B-tree.

Below you wil find the java code of the classes Btree, Node, Student, and Test. Copy this code to Eclips IDE and fill the missing parts with correct statements. (missing parts are green)

**Student Class**

**public** **class** Student {

**public** **long** number;

**public** String name;

**public** String surname;

**public** Student(**long** number, String name, String surname) {

String a=name,b=surname;

**this**.number = number;

**if**(name!=**null** && name.length()<10){

**for** (**int** i = 0; i < 10-name.length(); i++)

{

a= "." + a;

}

}

**this**.name = a;

**if**(surname!=**null**&&surname.length()<10){

**for** (**int** i = 0; i < 10-surname.length(); i++)

{

b= "." + b;

}

}

**this**.surname = b;

}

**public** **long** getNumber() {

**return** number;

}

**public** **void** setNumber(**long** number) {

**this**.number = number;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getSurname() {

**return** surname;

}

**public** **void** setSurname(String surname) {

**this**.surname = surname;

}

}

**Node Class**

**public** **class** Node {

**public** **int** mNumKeys = 0;

**public** Node[] mChildNodes = **new** Node[2 \* Btree.***T***];

**public** **boolean** mIsLeafNode;

**public** Student[] student=**new** Student[2 \* Btree.***T*** - 1];

**public** **int** pageNumber=0;

**public** Node() {

**for** (**int** i = 0; i < student.length; i++) {

student[i]=**new** Student(0, **null**,**null**);

}

**for** (**int** i = 0; i < mChildNodes.length; i++) {

mChildNodes[i]=**null**;

}

mIsLeafNode=**true**;

}

}

**Btree Class**

**public** **class** Btree {

**public** **static** Node *mRootNode*;

**public** **final** **static** **int** ***T*** = 2;

**public** Btree() **throws** Exception {

*mRootNode* = **new** Node();

*mRootNode*.mIsLeafNode = **true**;

}

**public** **void** Bulk\_Insert(File file) **throws** NumberFormatException, Exception {

FileReader frdr = **new** FileReader(file);

BufferedReader br = **new** BufferedReader(frdr);

String str;

String [] informs={};

**while**((str=br.readLine())!=**null**){

informs=str.split(",");

Insert(**new** Student(Long.*parseLong*(informs[0]),informs[1], informs[2]));

}

}

**public** **void** Insert(Student s) **throws** Exception {

Node rootNode = *mRootNode*;

**if** (rootNode.mNumKeys == (2 \* ***T*** - 1)) {

Node newRootNode = **new** Node();

*mRootNode* = newRootNode;

newRootNode.mIsLeafNode = **false**;

*mRootNode*.mChildNodes[0] = rootNode;

splitChildNode(newRootNode, 0, rootNode);

insertIntoNonFullNode(rootNode, s);

} **else** {

insertIntoNonFullNode(rootNode, s);

}

}

**void** splitChildNode(Node parentNode, **int** i, Node node) **throws** Exception

{

Node newNode = **new** Node();

newNode.mIsLeafNode = node.mIsLeafNode;

newNode.mNumKeys = ***T***-1;

**for** (**int** j = 0; j < ***T***-1; j++) {

newNode.student[j] = node.student[j + ***T***];

}

**if** (!node.mIsLeafNode) {

**for** (**int** j = 0; j < ***T*** + 1; j++) {

newNode.mChildNodes[j] = node.mChildNodes[j + ***T*** - 1];

}

**for** (**int** j = ***T***; j <= node.mNumKeys; j++) {

node.mChildNodes[j] = **null**;

}

}

node.mNumKeys = ***T*** - 1;

**for** (**int** j = parentNode.mNumKeys; j >= i + 1; j--) {

parentNode.mChildNodes[j + 1]=parentNode.mChildNodes[j];

}

parentNode.mChildNodes[i + 1] = newNode;

**for** ( int j = parentNode.mNumKeys-1; j >= i ; j-- ) {

parentNode.student[j+1]=parentNode.student[j];

}

parentNode.student[i] = node.student[***T***-1];

parentNode.mNumKeys++;

}

**void** insertIntoNonFullNode(Node node, Student stu) **throws** Exception

{

**int** i = node.mNumKeys - 1;

**if** (node.mIsLeafNode) {

**while** (i >= 0 && stu.number < node.student[i].number) {

node.student[i + 1] = node.student[i];

i--;

}

i++;

node.student[i] = stu;

node.mNumKeys++;

} **else** {

**while** (i >= 0 && stu.number < node.student[i].number) {

i--;

}

i++;

**if** (node.mChildNodes[i]!=**null**){

Node child=node.mChildNodes[i];

**if** (child!=**null**){

**if** (child.mNumKeys == (2 \* ***T*** - 1)) {

splitChildNode(node, i, child);

**if** (stu.number > node.student[i].number) {

i++;

}

}

insertIntoNonFullNode(node.mChildNodes[i], stu); }

}

}

}

**public** Student search(Node node, Student stu) {

**int** i = 0;

**while** (i < node.mNumKeys && stu.number > node.student[i].number) {

i++;

}

**if** (i < node.mNumKeys && stu.number == node.student[i].number) {

**return** node.student[i];

}

**if** (node.mIsLeafNode) {

**return** **null**;

} **else** {

Node child;

**try** {

**if** (node.mChildNodes[i]!=**null**){

child = node.mChildNodes[i];

**if** (child!=**null**)

**return** search(child, stu);

}

} **catch** (Exception e) {

e.printStackTrace();

}

}

**return** **null**;

}

}

**Test Class**

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Btree myBtree;

**try** {

myBtree = **new** Btree();

myBtree.Bulk\_Insert(**new** File("sdata.txt"));

System.***out***.println("search 3935245592,jtqfwphtfv,iadcptmxie");

Student st= myBtree.search(Btree.*mRootNode*, **new**

Student(Long.*parseLong*("3935245592"),"",""));

**if** (st !=**null** )

System.***out***.println("found "+st.number+ ""+st.name+""+st.surname);

**else**

System.***out***.println("Not found");

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**Task – 2**

Copy your codes below. Make your own statements red and upload this document.