public interface INode

public boolean addChild(String ID, String parentID)

public INode find(String value)

public INode getParent

public int size

public String toString

public String getId

public void printTree

public class TreeDataStructure implements INode

private String ID

private INode parent

private INode leftChild

private INode rightChild

public TreeDataStructure

public TreeDataStructure(String ID)

public TreeDataStructure(String ID, INode parent)

public TreeDataStructure(String ID, INode parent, INode leftChild)

public TreeDataStructure(String ID, INode parent, INode leftChild,

INode rightChild)

public boolean addChild(String ID, String parentID)

if ID.equals(parentID)

"Cannot add a node of same ID to parent ID."

return false

if this.getId is null

this.setID(ID)

return true

if this.getId.equals(parentID)

if this.hasLeftChild && this.hasRightChild

"Parent already has 2 children."

return false

if this.hasLeftChild

this.rightChild = new TreeDataStructure

return this.rightChild.addChild(ID, parentID)

this.leftChild = new TreeDataStructure

return this.leftChild.addChild(ID, parentID)

if this.find(parentID) is not null

return this.find(parentID).addChild(ID, parentID)

else

return false

public INode find(String value)

INode result = null

if value.equals(this.ID)

return this

if this.leftChild is null

result = this.leftChild.find(value)

if result is not null

return result

if this.rightChild is not null

result = this.rightChild.find(value)

return result

public INode getParent

return parent

public int size

int size = 1

if this.hasLeftChild

size add this.leftChild.size

if this.hasRightChild

size add this.rightChild.size

return size

public String toString

return this.ID

public String getId

return this.ID

public void printTree

if this.hasLeftChild && this.hasRightChild

Print root and children

this.leftChild.printTree

this.rightChild.printTree

else if this.hasLeftChild

Print root and left child

this.leftChild.printTree

else if this.hasRightChild

Print root and right child

this.rightChild.printTree

else

“", this.toString)

public INode getLeftChild

return this.leftChild

public INode getRightChild

return this.rightChild

public boolean hasParent

return this.parent is not null

public boolean hasLeftChild

return this.leftChild is not null

public boolean hasRightChild

return this.rightChild is not null

private void setID(String ID) ­­

this.ID = ID

public class BinaryTree

private static Scanner sc

public static void main(String[] args)

TreeDataStructure root = new TreeDataStructure(”A”)

root.addChild (”B", "A”)

root.addChild (”C", "A”)

root.addChild (”D", "B”)

root.addChild (”E", "B”)

root.addChild (”F“, "C”)

root.addChild (”G", "C”)

root.addChild (”H", "D”)

root.addChild (”I", "D”)

root.addChild (”J", "E”)

root.addChild (”K", "E”)

root.addChild (”L", "F“)

root.printTree

“There are # nodes in this tree”

while option isn’t 0

printMenu

option = promptForMenuOption

if option is 1

addNode(root)

if option is 2

getTreeSize(root)

if option is 3

findNode(root)

public static void printMenu

"Please select from one of the following options:"

"1. Add Node"

"2. Tree Size"

"3. Find Node"

"0. Exit"

"-> "

public static void addNode(INode root)

"Please input the node you want to add:"

String newNodeID = promptForCharacter

"Please input the parent node of :", newNodeID

String parentNodeID = promptForCharacter

if (root.find(parentNodeID) is null)

“Parent node was not found!”

if (root.addChild(newNodeID, parentNodeID))

“Node successfully added!”

root.printTree

public static void findNode(INode root)

“Please input the node you want to add:”

String searchCharacter = promptForCharacter

INode node = root.find(searchCharacter)

if (node is null)

“Node does not exist.", searchCharacter)

else

“Node found!", searchCharacter)

public static void getTreeSize(INode root)

“Please input the root node:”

String rootCharacter = promptForCharacter

INode rootNode = root.find(rootCharacter)

if (rootNode is null)

“Node does not exist.", rootCharacter)

return

“There are nodes in that tree.", rootNode.size)

rootNode.printTree

public static String promptForCharacter

Scanner sc = new Scanner(System.in)

String character = ""

if (sc.hasNext)

String input = sc.next

if (input.length > 0 && input.length < 2)

character = input

else

“Invalid input. Please enter a valid character:”

promptForCharacter

return character.toUpperCase

public static int promptForMenuOption

Scanner sc = new Scanner(System.in)

if (sc.hasNextInt)

int input = sc.nextInt

if (input is greater than or equal to 0 && input is less than or equal to 3)

return input

else

“Invalid input. Please enter a valid menu option.”

return -1

return -1