Class MorseCodeTree implements
LinkedConverterTreeInterface<String>

-root: TreeNode<Sting>

+MorseCodeTreeQ{
this.buildTreeQ;
}

-

+addNode(TreeNode<String> root, String code, String letter): void
create temp node equal to root
if code.length is equal to 1 temp is new TreeNode(letter);if
code.equals(".") root.setleft(temp);
else root.setRight(temp);
return
if code.charAt(O) == '.' addNode(temp.getLeftQ, code
.substring(l), letter);
else if code.charAt(O) == '-' addNode(temp.getRightQ,code.s
ubstr ing(l), letter); else throw new
NoSuchElementExceptionO

+buildTree():void setRoot(new
TreeNode<String>("");
insen(".", "e");
insen("-", "t");
insen("..", "i");
insen(".-", "a");
insen("-.", "n");
insen("--", "m");
insen("...", "s");
insen("..-", "u");
insen(".-.", "r");
insen(".--", "w");
insen("-..", "d");
insen("-.-", "k");
insen("--.", "g");
insen("---", "o");
insen("....", "h");
insen("...-", 'V');
insen(".-.", "r");
insen(". . ", "I");
insen(".--.", "p");
insen(".·-·", "j");
insen("-..", "b");
insen("-..-", "x");
insen("-.-.", "c");
insen("-- ", y°);
insen("···", "z'?;
insen("-- ", "q");

+fetch(String code)if code
equals *"/"* **return** ""
else return fetchNode(getRoot(),code)

+fetchNode(TreeNode<String>root, String code)
Create a temp TreeNode<String> equal to rootif code's
length == o
retu rn root.getDataQ;if
code.charAt(O) == '.'
return fetchNode(temp.getleft(), code.substring(l));else if
code.charAl(O) == '-'
return fetchNode(temp.getRightQ, code.substring(l);else
throw new NoSuchElementException()

+getRoot(): TreeNode<String
return this.root;

+insert(String code, String letter): MorseCodeTree
addNode(getRoot(), code, letter);
return this;

+LNRoutputTraversal(TreeNode<String>root, Arrayl ist<String< list):void
if root == nullreturn:
LNRoutputTraversal(root.getl eft(), list);
list.add(root.getData());
LNRoutputTraversal(root.getRight(),list);

+setRoot(TreeNode<String> newNode) this.root
= newNode;

+toArrayli stQ: Arrayli st<String>
Arrayli st<String> list = new Arrayli st<String>();LNRoutputTraversal(getRoot(),
list);
return list;

<< interfa ce>>
LinkedConverterTreelnterface

+ getRootQ:TreeNode<T>
+insen(T code, T
result):LinkedConverterTreeelnterface<T>
+addNode(TreeNode<T> root, T code, T
letter): void
+fetch(Str ing code): T
+ fetchNode(TreeNode<T> root, T code):
+delete(T data):
LinkedConverterTreelnterface<T>
+update(): LinkedConvenerTreelnterface
+toArrayli st):Arraylist<T>

Class TreeNode

- data T;
+ leftChild:
TreeNode
+rightChild:
TreeNode

TreeNode(TData)
constructor set left and right child
to nullwith the data for the node
equal to data
TreeNode(TreeNode<T> node)
constructor set left and right child to
nodedata and data equal to node data

+set eft(TreeNodeT> left):
void set this.left to left

+setRight(TreeNode<T>right):
void set this.right to right
+getR1ghtQ:
TreeNodereturn
this.rightChild
+getleft():
TreeNodereturn
this.leftChild
+getDataQ: T
return this.data
+hasRightQ:
boolean return
rightChild != null;
+hasleftQ:
boolean return
leftChild != null;

Class MorseCodeConvener

- **static tree: MorseCodeTree**
- static answer: String
static codes: String[]

+static String convenToEnglish(File
codeFile) throws
FileNotFoundException:
String

tree= new MorseCodeTreeQ;
ans="";
try{

Scanner keyboard = new
Scanner(codeFile);

while(keyboard.hasNextLineQ){

codes=

keyboardnextlineQsplit(" ");

for(int i = O; i < codes .length;
i++){ans+=
tree.fetch(codesi][];
)
}
keyboard.closeQ;
returnans;
}
catch(FileNotFoundExcept ion
e){ throw new
FileNotFoundExcepiton();
}

+ static convenToEnglish(String
code):
String
tree= new MorseCodeTreeQ;
ans="";
codes= code.split(" ");
for(int i = O; i< codes .leng th;
i++){
ans+= tree.fetch(codes[i]);
}
return ans;

+ printTreeQ: String

tree= new

MorseCodeTreeQ;
String s = "";
for(String letter:
tree.toArraylistQ){
s+= letter + " ";
}
return s.substring(O, s.lengthQ -1);