1/16 树的节点

树的节点实现参考链接：<http://blog.csdn.net/shanzhizi/article/details/13022749>

这是一颗比较传统的多叉树代码，在后续与图形化连接的过程中会有一定的改动

打开javatree项目，我们在MainActivity所在的文件夹里增加一个新的java class叫做TreeNode，这个类能在MainActivity里不需要import就能用

![C:\Users\1535725170\AppData\Roaming\Tencent\Users\1535725170\QQ\WinTemp\RichOle\](G1QS1A9I@JX~V%D~IO@QO.png](data:image/png;base64,)

在TreeNode里面，我们加入以下简单的代码

**public class** TreeNode {  
 *//节点的id* **private** String **nodeId**;  
 *//父节点的id* **private** String **parentId**;  
 *//节点的内容* **private** String **text**;  
 *//构造函数* **public** TreeNode(String nodeId){  
 **this**.**nodeId** = nodeId;  
 }  
 *//构造函数* **public** TreeNode(String nodeId,String parentId){  
 **this**.**nodeId** = nodeId;  
 **this**.**parentId** = parentId;  
 }  
 *//设置节点的id* **public void** setNodeId(String nodeId){  
 **this**.**nodeId** = nodeId;  
 }  
 *//得到节点的id* **public** String getNodeId(){  
 **return nodeId**;  
 }  
}

🡨---------------------------------------------------------🡪

接下来我们在MainActivity的onClick函数里面加入如下测试代码



这里有两个注意点:

一个是类的实例化代码 如TreeNode xxx(随便命名) = new TreeNode(参数)

另一个是涉及String的操作，为了保险起见，对得到的字符串可以加一个.toString（）函数。

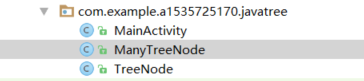
发现是可以正常运行的，所以在TreeNode里对parentNode和text进行相同的set和get函数

*//设置父节点的id***public void** setParentId(String parentId){  
 **this**.**parentId** = parentId;  
}  
  
*//得到父节点的id***public** String getParentId(){  
 **return parentId**;  
}  
  
*//设置节点的text***public void** setText(String text){  
 **this**.**text** = text;  
}  
  
*//得到节点的text***public** String getText(){  
 **return text**;  
}

这样，一个最最简单的节点就这样做好了，接下来我们以这个简单的节点为基础，建立多叉树节点

代码也是很简单感人，唯一需要的是去了解一个List，ArrayList在java里的应用以及一些类转换的骚操作

代码我就直接放这里了，和上一段代码一样易懂。新建一个ManyTreeNode。



**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Created by 1535725170 on 2018/1/18.  
 \*/***public class** ManyTreeNode {  
  
 **private** TreeNode **data**;  
  
 **private** List<ManyTreeNode> **childList**;  
  
 *//to bo used* **private** ManyTreeNode **parentNode**;  
  
 **public** ManyTreeNode(TreeNode data){  
 **this**.**data**=data;  
 **this**.**childList** = **new** ArrayList<ManyTreeNode>();  
 }  
  
 **public** ManyTreeNode(TreeNode data,List<ManyTreeNode>childList){  
 **this**.**data**=data;  
 **this**.**childList**=childList;  
 }  
  
 **public** TreeNode getData(){  
 **return data**;  
 }  
  
 **public** List<ManyTreeNode>getChildList(){  
 **return childList**;  
 }  
  
 **public void** setChildList(List<ManyTreeNode>childList) {  
 **this**.**childList** = childList;  
 }  
  
 **public void** setDataId(String newId){  
 **this**.**data**.setNodeId(newId);  
 }  
  
}

以上的两个包List和ArrayList是在打代码打到相应的List和ArrayList的时候自动导入的，如果你按照我的一句一句打代码的话，前面是不需要import的，这是安卓的便捷之处。

这里面我还没有打parentNode父节点有关的代码，因为现在只是测试网上的代码用的，到后期会有根据图形化界面的需求改动内部代码的。

注释我后期会直接打到项目里，其实是没必要的，很容易看懂这段代码。

下面是上述代码的测试代码，跑的还不错



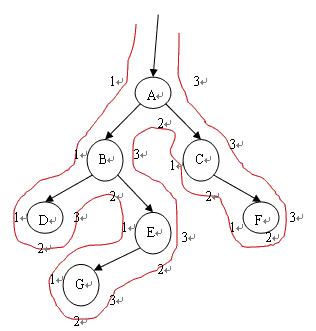
最后是多叉树代码，用来管理内部的多叉树节点的

这部分和网上的有很大的差别，主要是我自己写的，顺便改了很多TreeNode和ManyTreeNode中的代码，以尽可能地满足图形化编程代码板块的需要。程序放在最后了。

一番操作和debug之后，经过大改造的树的管理和遍历完成了。ManyNodeTree中的代码我不给过多讲解，只要能看懂网上的代码，自然能看懂我的代码，其中逻辑很简单，其中最难的函数是先序遍历函数iteratorTree（），用的是递归的算法。

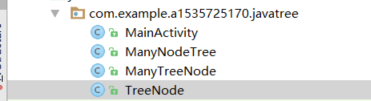
如果算法基础不牢固的，建议去查找递归算法的一些例子。

这里也可以不需要了解的十分详细，只要知道这是用某种顺序来打印出一棵树的就行。

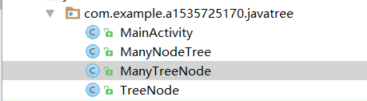


上图是前序遍历的简单例子：根据红线从左边进去，从右边出来的过程中，经历节点的顺序就是读取树的节点的顺序。

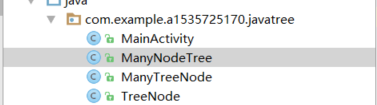
接下来给出数据结构的完整代码



**public class** TreeNode {  
 *//节点的id* **private** String **nodeId**;  
  
 *//父节点的id* **private** String **parentId**;  
  
 *//节点的内容* **private** String **text**;  
  
 *//构造函数* **public** TreeNode(String nodeId){  
 **this**.**nodeId** = nodeId;  
 }  
  
 *//构造函数* **public** TreeNode(String nodeId,String parentId){  
 **this**.**nodeId** = nodeId;  
 **this**.**parentId** = parentId;  
 }  
  
 *//设置节点的id* **public void** setNodeId(String nodeId){  
 **this**.**nodeId** = nodeId;  
 }  
  
 *//得到节点的id* **public** String getNodeId(){  
 **return nodeId**;  
 }  
  
 *//设置父节点的id* **public void** setParentId(String parentId){  
 **this**.**parentId** = parentId;  
 }  
  
 *//得到父节点的id* **public** String getParentId(){  
 **return parentId**;  
 }  
  
 *//设置节点的text* **public void** setText(String text){  
 **this**.**text** = text;  
 }  
  
 *//得到节点的text* **public** String getText(){  
 **return text**;  
 }  
  
}

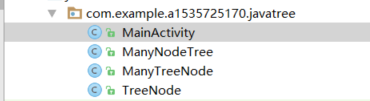


**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* Created by 1535725170 on 2018/1/18.  
 \*/***public class** ManyTreeNode {  
  
 *//树节点的信息* **private** TreeNode **data**;  
  
 *//子节点集合* **private** List<ManyTreeNode> **childList**;  
  
 *//to bo used* **private** ManyTreeNode **parentNode**;  
  
 *//构造函数* **public** ManyTreeNode(TreeNode data){  
 **this**.**data**=data;  
 **this**.**childList** = **new** ArrayList<ManyTreeNode>();  
 }  
  
 *//构造函数* **public** ManyTreeNode(TreeNode data,List<ManyTreeNode>childList){  
 **this**.**data**=data;  
 **this**.**childList**=childList;  
 }  
  
 *//得到该节点的信息* **public** TreeNode getData(){  
 **return data**;  
 }  
  
 *//得到该节点的子节点集合* **public** List<ManyTreeNode>getChildList(){  
 **return childList**;  
 }  
  
 *//设置该节点的子节点集合* **public void** setChildList(List<ManyTreeNode>childList) {  
 **this**.**childList** = childList;  
 }  
  
 *//设置该节点的信息中的节点id* **public void** setDataId(String newId){  
 **this**.**data**.setNodeId(newId);  
 }  
  
 *//设置父节点* **public void** setParentNode(ManyTreeNode parentNode){  
 **this**.**parentNode**=parentNode;  
 }  
  
 **public** ManyTreeNode getParentNode(){  
 **return parentNode**;  
 }  
  
}



**public class** ManyNodeTree {  
 **private** ManyTreeNode **root**;  
  
 **private** ManyTreeNode **tmp**;  
  
 **private int floor**;  
  
 **public** ManyNodeTree(){  
 **root** = **new** ManyTreeNode(**new** TreeNode(**"root"**));  
 **tmp** = **root**;  
 **floor** = 0;  
 }  
  
 **public** ManyTreeNode getRoot(){  
 **return root**;  
 }  
  
 **public** ManyTreeNode getTmp(){  
 **return tmp**;  
 }  
  
 **public void** addChild(ManyTreeNode manyTreeNode){  
 **this**.**tmp**.getChildList().add(manyTreeNode);  
 manyTreeNode.setParentNode(**this**.**tmp**);  
 }  
  
 **public void** beChild(**int** n){  
 **if**(**tmp**.getChildList()!=**null**&&**tmp**.getChildList().size()>n) {  
 ManyTreeNode newNode = **this**.**tmp**.getChildList().get(n);  
 **tmp** = newNode;  
 **floor** = **floor** + 1;  
 }  
 }  
  
 **public void** beParent(){  
 **if**(**tmp**.getData().getNodeId()!=**"root"**){  
 ManyTreeNode newNode = **tmp**.getParentNode();  
 **tmp** = newNode;  
 **floor** = **floor** + 1;  
 }  
 }  
  
 **public** String iteratorTree(ManyTreeNode manyTreeNode){  
 StringBuilder buffer = **new** StringBuilder();  
 buffer.append(**"\n"**);  
 **if**(manyTreeNode!=**null**){  
 **for**(ManyTreeNode index : manyTreeNode.getChildList()){  
 buffer.append(index.getData().getNodeId()+**","**);  
  
 **if**(index.getChildList()!=**null**&&index.getChildList().size()>0){  
 buffer.append(iteratorTree(index));  
 }  
 }  
 }  
 buffer.append(**"\n"**);  
 **return** buffer.toString();  
 }  
  
}

以下是全部的测试代码



**import** android.os.Bundle;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** MainActivity **extends** AppCompatActivity **implements** View.OnClickListener {  
  
 Button **bt**;*//定义按钮* TextView **tx**;*//定义显示结果的text* @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **bt** = (Button)findViewById(R.id.***bt***);*//实例化按钮* **tx** = (TextView)findViewById(R.id.***text***);*//实例化text* **bt**.setOnClickListener(**this**);*//给按钮设置点击事件* }  
  
 @Override  
 **public void** onClick(View v) {  
 ManyNodeTree manyNodeTree = **new** ManyNodeTree();  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"1"**)));  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"2"**)));  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"3"**)));  
 manyNodeTree.beChild(1);  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"4"**)));  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"5"**)));  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"6"**)));  
 manyNodeTree.beParent();  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"d"**)));  
 manyNodeTree.beChild(2);  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"7"**)));  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"8"**)));  
 manyNodeTree.addChild(**new** ManyTreeNode(**new** TreeNode(**"9"**)));  
  
  
 String output=**tx**.getText().toString();  
 **switch** (v.getId()){  
 **case** R.id.***bt***:  
 output = manyNodeTree.iteratorTree(manyNodeTree.getRoot());  
 **break**;  
 **default**:  
 **break**;  
 }  
 **tx**.setText(output);  
 }  
}

这样大概的树就做好了