软件构造第4次实验

实验报告

**班级**：软件工程14-3

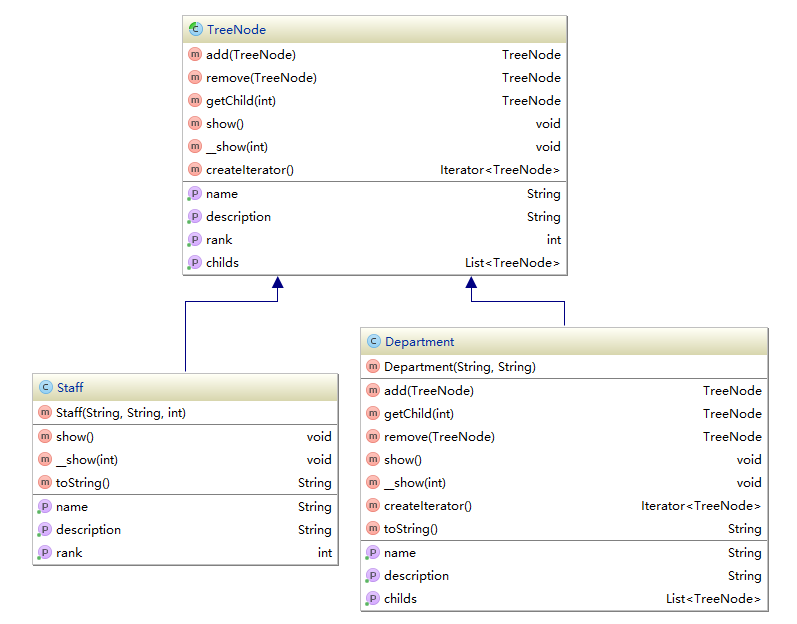
**学号**：201401061038

**姓名**：张正锟

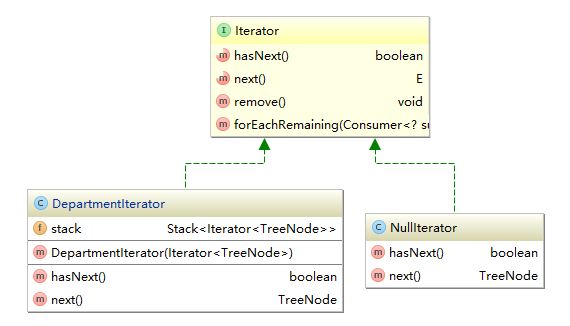
2016年4月17日

# 设计

## 公司员工树形结构设计类图



## 公司员工迭代器设计类图



# 实现

## 树形结构节点实现

**package** com.zzkun.company;  
  
**import** java.util.Iterator;  
**import** java.util.List;  
  
*/\*\*  
 \* Created by kun on 2016/4/13.  
 \*/***public abstract class** TreeNode {  
 */\*\*  
 \* 在当前节点添加子节点  
 \** ***@param node*** *子节点  
 \*/* **public** TreeNode add(TreeNode node) {  
 **throw new** UnsupportedOperationException();  
 }  
  
 */\*\*  
 \* 删除当前节点的子节点  
 \** ***@param node*** *子节点  
 \*/* **public** TreeNode remove(TreeNode node) {  
 **throw new** UnsupportedOperationException();  
 }  
  
 */\*\*  
 \* 得到第i个位置的子节点  
 \** ***@param i*** *位置  
 \** ***@return*** *第i个位置的子节点  
 \*/* **public** TreeNode getChild(**int** i) {  
 **throw new** UnsupportedOperationException();  
 }  
  
 **public** List<TreeNode> getChilds() {  
 **throw new** UnsupportedOperationException();  
 }  
  
 */\*\*  
 \* 获取当前节点的名称  
 \** ***@return*** *当前节点的名称  
 \*/* **public** String getName() {  
 **throw new** UnsupportedOperationException();  
 }  
  
 */\*\*  
 \* 获取当前节点的描述  
 \** ***@return*** *当前节点的描述  
 \*/* **public** String getDescription() {  
 **throw new** UnsupportedOperationException();  
 }  
  
 */\*\*  
 \* 获取员工排名  
 \** ***@return*** *员工排名  
 \*/* **public int** getRank() {  
 **throw new** UnsupportedOperationException();  
 }  
  
 */\*\*  
 \* 递归打印当前节点及子节点  
 \*/* **public void** show() {  
 **throw new** UnsupportedOperationException();  
 }  
  
 **public void** \_\_show(**int** dent) {  
 **throw new** UnsupportedOperationException();  
 }  
  
 **public** Iterator<TreeNode> createIterator() {  
 **throw new** UnsupportedOperationException();  
 }  
}

## 员工节点实现

**public class** Staff **extends** TreeNode {  
  
 String **name**;  
 String **description**;  
 **int rank**;  
  
 **public** Staff(String name, String description, **int** rank) {  
 **this**.**name** = name;  
 **this**.**description** = description;  
 **this**.**rank** = rank;  
 }  
  
 @Override  
 **public** String getName() {  
 **return name**;  
 }  
  
 @Override  
 **public** String getDescription() {  
 **return description**;  
 }  
  
 @Override  
 **public int** getRank() {  
 **return rank**;  
 }  
  
 @Override  
 **public void** show() {  
 \_\_show(0);  
 }  
  
 @Override  
 **public void** \_\_show(**int** dent) {  
 String pre = **""**;  
 **for**(**int** i = 0; i < dent; ++i) pre += **" "**;  
 System.***out***.println(pre + getName() + **"(Rank"** + getRank() + **"): "** + getDescription());  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Staff{"** +  
 **"name='"** + **name** + **'\''** +  
 **", description='"** + **description** + **'\''** +  
 **", rank="** + **rank** +  
 **'}'**;  
 }  
}

## 公司部门节点实现

**public class** Department **extends** TreeNode {  
  
 List<TreeNode> **list** = **new** ArrayList<>();  
 String **name**;  
 String **description**;  
  
 **public** Department(String name, String description) {  
 **this**.**name** = name;  
 **this**.**description** = description;  
 }  
  
 @Override  
 **public** TreeNode add(TreeNode node) {  
 **list**.add(node);  
 **return this**;  
 }  
  
 @Override  
 **public** TreeNode getChild(**int** i) {  
 **return list**.get(i);  
 }  
  
 @Override  
 **public** List<TreeNode> getChilds() {  
 **return list**;  
 }  
  
 @Override  
 **public** TreeNode remove(TreeNode node) {  
 **list**.remove(node);  
 **return this**;  
 }  
  
 @Override  
 **public** String getName() {  
 **return name**;  
 }  
  
 @Override  
 **public** String getDescription() {  
 **return description**;  
 }  
  
 @Override  
 **public void** show() {  
 \_\_show(0);  
 }  
  
 @Override  
 **public void** \_\_show(**int** dent) {  
 System.***out***.println();  
 String pre = **""**;  
 **for**(**int** i = 0; i < dent; ++i) pre += **" "**;  
 System.***out***.println(pre + **"-------"** + getName() + **"-------"**);  
 System.***out***.println(pre + **"描述："** + getDescription());  
 **for**(TreeNode node : **list**) {  
 node.\_\_show(dent + 1);  
 }  
 }  
  
 @Override  
 **public** Iterator<TreeNode> createIterator() {  
 **return new** DepartmentIterator(**list**.iterator());  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Department{"** +  
 **"list="** + **list** +  
 **", name='"** + **name** + **'\''** +  
 **", description='"** + **description** + **'\''** +  
 **'}'**;  
 }  
}

## 公司部门迭代器实现

**public class** DepartmentIterator **implements** Iterator<TreeNode> {  
  
 Stack<Iterator<TreeNode>> **stack** = **new** Stack<>();  
  
 **public** DepartmentIterator(Iterator<TreeNode> iterator) {  
 **stack**.push(iterator);  
 }  
  
 @Override  
 **public boolean** hasNext() {  
 **if**(**stack**.empty()) {  
 **return false**;  
 }  
 Iterator<TreeNode> iterator = **stack**.peek();  
 **if**(!iterator.hasNext()) {  
 **stack**.pop();  
 **return** hasNext();  
 }  
 **return true**;  
 }  
  
 @Override  
 **public** TreeNode next() {  
 **if**(!hasNext()) {  
 **return null**;  
 }  
 Iterator<TreeNode> iterator = **stack**.peek();  
 TreeNode node = iterator.next();  
 **if**(node **instanceof** Department) {  
 **stack**.push(node.getChilds().iterator());  
 **return** next();  
 }  
 **return** node;  
 }  
}

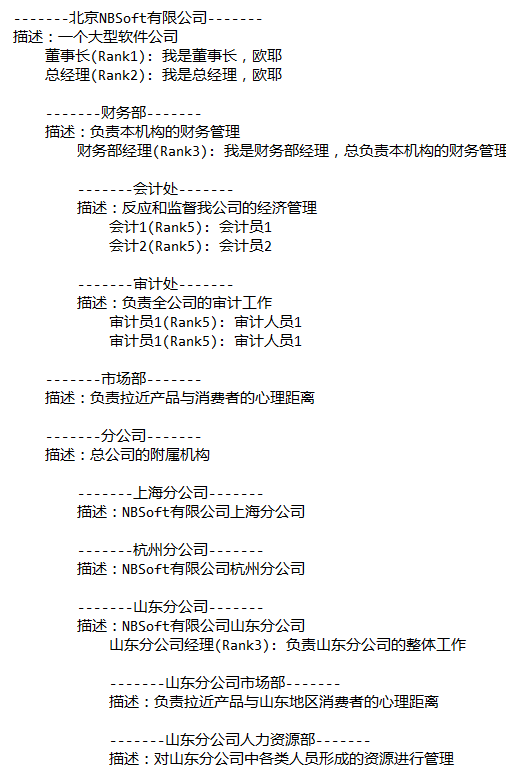
## 空迭代器实现

**public class** NullIterator **implements** Iterator<TreeNode> {  
  
 @Override  
 **public boolean** hasNext() {  
 **return false**;  
 }  
  
 @Override  
 **public** TreeNode next() {  
 **return null**;  
 }  
}

# 测试

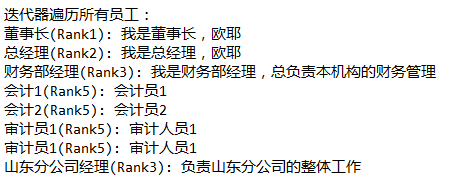
## 输出测试

TreeNode root = **new** Department(**"北京NBSoft有限公司"**, **"一个大型软件公司"**);  
  
root.add(**new** Staff(**"董事长"**, **"我是董事长，欧耶"**, 1)).add(**new** Staff(**"总经理"**, **"我是总经理，欧耶"**, 2));  
Department caiwu = **new** Department(**"财务部"**, **"负责本机构的财务管理"**);  
Department shichang = **new** Department(**"市场部"**, **"负责拉近产品与消费者的心理距离"**);  
Department fengongsi = **new** Department(**"分公司"**, **"总公司的附属机构"**);  
root.add(caiwu).add(shichang).add(fengongsi);  
  
caiwu.add(**new** Staff(**"财务部经理"**, **"我是财务部经理，总负责本机构的财务管理"**, 3));  
Department kuaijichu = **new** Department(**"会计处"**, **"反应和监督我公司的经济管理"**);  
Department shenjichu = **new** Department(**"审计处"**, **"负责全公司的审计工作"**);  
caiwu.add(kuaijichu).add(shenjichu);  
kuaijichu.add(**new** Staff(**"会计1"**, **"会计员1"**, 5)).add(**new** Staff(**"会计2"**, **"会计员2"**, 5));  
shenjichu.add(**new** Staff(**"审计员1"**, **"审计人员1"**, 5)).add(**new** Staff(**"审计员1"**, **"审计人员1"**, 5));  
  
fengongsi.add(**new** Department(**"上海分公司"**, **"NBSoft有限公司上海分公司"**));  
fengongsi.add(**new** Department(**"杭州分公司"**, **"NBSoft有限公司杭州分公司"**));  
Department sdfengongsi = **new** Department(**"山东分公司"**, **"NBSoft有限公司山东分公司"**);  
fengongsi.add(sdfengongsi);  
sdfengongsi.add(**new** Staff(**"山东分公司经理"**, **"负责山东分公司的整体工作"**, 3));  
sdfengongsi.add(**new** Department(**"山东分公司市场部"**, **"负责拉近产品与山东地区消费者的心理距离"**));  
sdfengongsi.add(**new** Department(**"山东分公司人力资源部"**, **"对山东分公司中各类人员形成的资源进行管理"**));  
  
root.show();



## 迭代器遍历所有员工

System.***out***.println(**"迭代器遍历所有员工："**);  
Iterator<TreeNode> iterator = root.createIterator();  
**while**(iterator.hasNext()) {  
 TreeNode node = iterator.next();  
 **if**(node != **null**) node.show();  
}



## 迭代器遍历所有分公司员工

System.***out***.println(**"迭代器遍历所有分公司员工："**);  
iterator = fengongsi.createIterator();  
**while**(iterator.hasNext()) {  
 TreeNode node = iterator.next();  
 **if**(node != **null**) node.show();  
}



## 迭代器遍历所有等级高于3的员工

System.***out***.println(**"迭代器遍历所有等级高于3的员工："**);  
iterator = root.createIterator();  
**while**(iterator.hasNext()) {  
 TreeNode node = iterator.next();  
 **if**(node != **null** && node.getRank() <= 3) node.show();  
}

