

```
1  package other;
2  import java.io.File;
3  import java.io.FileNotFoundException;
4  import java.io.FileOutputStream;
5  import java.io.FileWriter;
6  import java.io.IOException;
7  import java.io.OutputStream;
8  import java.io.OutputStreamWriter;
9  import java.io.Writer;
10 import java.util.ArrayList;
11 import java.util.HashMap;
12 import java.util.Iterator;
13 import java.util.Map;
14 import java.util.Map.Entry;
15 import java.util.Set;
16 import org.dom4j.Attribute;
17 import org.dom4j.Document;
18 import org.dom4j.Element;
19 import org.dom4j.io.OutputFormat;
20 import org.dom4j.io.SAXReader;
21 import org.dom4j.io.XMLWriter;
22 import com.test.bpelbean.Message;
23 import com.test.bpelbean.Value;
24 import com.test.bean.Activity;
25 import com.test.bean.Atom;
26 import com.test.bean.Structure;
27 public class ParseBpel {
28     public static Structure activity=new Structure("begin");
29     public static ArrayList<String> If=new ArrayList<String>();
30     public static HashMap<String,String> If_=new HashMap<String,String>();
31     public static HashMap<String,String> If_Branch=new HashMap<String,String>();
32     public static ArrayList<String> collection=new ArrayList<String>();
33     public static void main(String[] args) throws FileNotFoundException, IOException{
34         String filename="D:\\Workspace1\\DebugTest\\bpel\\TravelAgency.bpel";
35         File file = new File(filename);
36         Document document = openXMLFile(file.getAbsolutePath());
37         Element root = document.getRootElement();
38         for (Iterator iter = root.elementIterator(); iter.hasNext();){
39             Element element = (Element) iter.next();
40             String attr=element.attributeValue("name");
41             System.out.println(attr);
42             for(Iterator iter1 = element.elementIterator(); iter1.hasNext();){
43                 Element element1 = (Element) iter1.next();
44                 String attr1=element1.attributeValue("name");
45                 System.out.println("    "+attr1);
46             }
47             System.out.println(attr1);
48         }
49     }
50     public static Document openXMLFile(String filePath) {
```

```
51     Document document = null;
52     SAXReader reader = new SAXReader();
53     try {
54         File file = new File(filePath);
55         document = reader.read(file);
56     } catch (Exception e) {
57         e.printStackTrace();
58     }
59     return document;
60 }
61 public static boolean readwsdlfile(String filepath) throws FileNotFoundException,
62 IOException {
63     try {
64         File file = new File(filepath);
65         if (!file.isDirectory()) {
66             System.out.println("absolutePath=" + file.getAbsolutePath());
67             System.out.println("name=" + file.getName());
68             String absolutePath = file.getAbsolutePath();
69             Document document = openXMLFile(absolutePath);
70             parseWSDL(document);
71         } else if (file.isDirectory()) {
72             String[] fileList = file.list();
73             for (int i = 0; i < fileList.length; i++) {
74                 File readfile = new File(filepath + "\\" + fileList[i]);
75                 if (!readfile.isDirectory()) {
76                     if (fileList[i].contains(".wsdl")) {
77                         String absolutePath = readfile.getAbsolutePath();
78                         Document document = openXMLFile(absolutePath);
79                         parseWSDL(document);
80                     }
81                 } else if (readfile.isDirectory()) {
82                     readwsdlfile(filepath + "\\" + fileList[i]);
83                 }
84             }
85         }
86     } catch (FileNotFoundException e) {
87         System.out.println("readfile() Exception:" + e.getMessage());
88     }
89     if_if();
90     return true;
91 }
92 public static void parseWSDL(Document document) {
93     Element root = document.getRootElement();
94     for (Iterator iter = root.elementIterator(); iter.hasNext();){
95         Element element = (Element) iter.next();
96         String attr=element.attributeValue("name");
97         if((attr!=null)&&attr.equals("main")){
98             parseMain(element,activity);
99         }
100     }
```

```
101         for(Activity component : activity.components){
102             if(component.getClass().getName().equals("com.test.bean.Structure")){
103                 If_.put(component.getChild(),component.getName());
104             }
105         }
106     }
107     public static void parseMain(Element node,Activity act){
108         for (Iterator iter = node.elementIterator(); iter.hasNext();){
109             Element element = (Element) iter.next();
110             String type=element.getName();
111             switch(type){
112                 case "assign":
113                 case "invoke":
114                 case "receive":
115                 case "reply":
116                     parseYuan(element,act);
117                     break;
118                 case "flow":
119                     parseFlow(element,act);
120                     break;
121                 case "sequence":
122                     parseSeq(element,act);
123                     break;
124                 case "if":
125                     parseIf(element,act);
126                     break;
127             }
128         }
129         private static void parseSeq(Element node,Activity act){
130             Structure seq=new Structure(node.getName());
131             for (Iterator iter = node.elementIterator(); iter.hasNext();){
132                 Element element = (Element) iter.next();
133                 parseMain(element,act);
134             }
135         }
136         private static void parseFlow(Element node,Activity act){
137             String attr=node.attributeValue("name");
138             System.out.println("    flow"+attr);
139             if(attr!=null)
140                 collection.add(attr);
141             for (Iterator iter = node.elementIterator(); iter.hasNext();){
142                 Element element = (Element) iter.next();
143                 parseMain(element,act);
144             }
145         }
146         private static void parseYuan(Element node,Activity act) {
147             String attr=node.attributeValue("name");
148             String key=act.getName();
149             if(If_Branch.containsKey(act.getName())){
150                 String value=If_Branch.get(key);
```

```
151         If_Branch.put(act.getName(), value+attr+"#");
152     }
153     act.add(new Atom(attr));
154     if(attr!=null)
155         collection.add(attr);
156 }
157 private static void parseElse(Element node,Activity act){
158     for (Iterator iter = node.elementIterator(); iter.hasNext();){
159         Element element = (Element) iter.next();
160         parseMain(element,act);
161     }
162 }
163 private static void parseIf(Element node,Activity act) {
164     String attr=node.attributeValue("name");
165     String key=act.getName();
166     Structure _if=new Structure(attr);
167     activity.add(_if);
168     if(If_Branch.containsKey(act.getName())){
169         String value=If_Branch.get(key);
170         If_Branch.put(act.getName(), value+attr+"#");
171     }
172     If_Branch.put(attr, "");
173     If.add(attr);
174     collection.add(attr);
175     for (Iterator iter = node.elementIterator(); iter.hasNext();){
176         Element element = (Element) iter.next();
177         String type=element.getName();
178         if(type.equals("condition"))
179             continue;
180         if(type.equals("else"))
181             parseElse(element,_if);
182         parseMain(element,_if);
183     }
184 }
185 public static void if_if(){
186     int n=ParseBpel.If_Branch.size();
187     String[] key=new String[n];
188     String[] value=new String[n];
189     int i=0;
190     for(Entry<String, String> entry: ParseBpel.If_Branch.entrySet()){
191         key[i]=entry.getKey();
192         value[i++]=entry.getValue();
193     }
194     for(i=0;i<n;i++){
195         for(int j=0;j<n;j++){
196             if(i==j)
197                 continue;
198             if(value[i].contains(key[j])){
199                 ParseBpel.If_Branch.put(key[i],
200 ParseBpel.If_Branch.get(key[i])+value[j]);
```

```
201         }
202     }
203 }
204 }
205 }
206 package other;
207 import java.io.File;
208 import com.test.XMLHelper_Ran;
209 public class Name {
210     public static void main(String args[]) throws Exception{
211         String output="E:\\毕设实验\\Quote\\path.txt";
212         File file = new File("C:\\Users\\Administrator\\Desktop\\实验\\quotemutant\\");
213         File[] files = file.listFiles();
214         for(int i = 0; i < files.length; i++){
215             System.out.println(files[i].getName());
216             XMLHelper_Ran.stringexport(files[i].getName()+"\n",output);
217         }
218     }
219 }
220 package other;
221 import java.io.BufferedReader;
222 import java.io.File;
223 import java.io.FileInputStream;
224 import java.io.InputStreamReader;
225 import java.util.regex.Pattern;
226 import com.test.XMLHelper_Ran;
227 public class Quote_Add {
228     public static void main(String[] args) throws Exception{
229         System.out.println("begin:");
230         String output="E:\\毕设实验\\Quote\\path_new444.txt";
231         String path1="E:\\毕设实验\\Quote\\path_Add.txt";
232         String path2="E:\\毕设实验\\Quote\\path_new4.txt";
233         File file1=new File(path1);
234         InputStreamReader isr1=new InputStreamReader(new FileInputStream(file1));
235         BufferedReader br1=new BufferedReader(isr1);
236         File file2=new File(path2);
237         InputStreamReader isr2=new InputStreamReader(new FileInputStream(file2));
238         BufferedReader br2=new BufferedReader(isr2);
239         String line1;
240         while((line1=br1.readLine())!=null){
241             boolean b=Pattern.matches(".*bpel", line1);
242             if(b){
243                 System.out.println(line1);
244                 XMLHelper_Ran.stringexport(line1+"\n",output);
245                 String line2;
246                 while((line2=br2.readLine())!=null){
247                     if(line2.equals(line1)){
248                         System.out.println("    haha"+line2);
249                     }else{
250                         boolean b2=Pattern.matches(".*bpel", line2);
```

```
251             if(b2){
252                 break;
253             }else{
254                 System.out.println(line2);
255                 XMLHelper_Ran.stringexport(line2+"\n",output);
256             }
257         }
258     }
259 }
260 else{
261     System.out.println(line1);
262     XMLHelper_Ran.stringexport(line1+"\n",output);
263 }
264 }
265 }
266 }
267 package other;
268 import java.io.BufferedReader;
269 import java.io.File;
270 import java.io.FileInputStream;
271 import java.io.InputStreamReader;
272 import java.util.ArrayList;
273 import java.util.regex.Pattern;
274 import com.test.XMLHelper_Ran;
275 public class Quote_PathAdd2 {
276     public static void main(String[] args) throws Exception{
277         System.out.println("begin:");
278         String output="E:\\毕设实验\\Quote\\path_Add3.txt";
279         String path1="E:\\毕设实验\\Quote\\path.txt";
280         String path2="E:\\毕设实验\\Quote\\path_Quote_Mutant3.txt";
281         int[] array={ 1,5,4,7,11 };
282         ArrayList list=new ArrayList();
283         for(int num: array)
284             list.add(num);
285         File file1=new File(path1);
286         InputStreamReader isr1=new InputStreamReader(new FileInputStream(file1));
287         BufferedReader br1=new BufferedReader(isr1);
288         File file2=new File(path2);
289         InputStreamReader isr2=new InputStreamReader(new FileInputStream(file2));
290         BufferedReader br2=new BufferedReader(isr2);
291         String line1;
292         while((line1=br1.readLine())!=null){
293             boolean b=Pattern.matches(".*bpel", line1);
294             if(b){
295                 System.out.println(line1);
296                 XMLHelper_Ran.stringexport(line1+"\n",output);
297                 String line2;
298                 int count=0;
299                 while((line2=br2.readLine())!=null){
300                     if(line2.equals(line1)){
```

```
301         count=0;
302     }else{
303         boolean b2=Pattern.matches(".*bpe1", line2);
304         if(b2){
305             count=0;
306             break;
307         }else{
308             count++;
309             if(list.contains(count)){
310                 System.out.println("add "+line2);
311                 XMLHelper_Ran.stringexport(line2+"\n",output);
312             }
313         }
314     }
315 }
316 }
317 else{
318     System.out.println("1 "+line1);
319     XMLHelper_Ran.stringexport(line1+"\n",output);
320 }
321 }
322 }
323 }
324 package other;
325 import java.io.BufferedReader;
326 import java.io.File;
327 import java.io.FileInputStream;
328 import java.io.InputStreamReader;
329 import java.util.ArrayList;
330 import java.util.regex.Pattern;
331 import com.test.XMLHelper_Ran;
332 public class Smart_PathAdd2 {
333     public static void main(String[] args) throws Exception{
334         System.out.println("begin:");
335         String output="E:\\毕设实验\\SmartShelf\\path_temp3.txt";
336         String path1="E:\\毕设实验\\SmartShelf\\path.txt";
337         String path2="E:\\毕设实验\\SmartShelf\\path_old3.txt";
338         int i=5;
339         int[] array={ 21,22,23,24};
340         ArrayList list=new ArrayList();
341         for(int num: array)
342             list.add(num);
343         File file1=new File(path1);
344         InputStreamReader isr1=new InputStreamReader(new FileInputStream(file1));
345         BufferedReader br1=new BufferedReader(isr1);
346         File file2=new File(path2);
347         InputStreamReader isr2=new InputStreamReader(new FileInputStream(file2));
348         BufferedReader br2=new BufferedReader(isr2);
349         String line1;
350         while((line1=br1.readLine())!=null){
```

```
351         boolean b=Pattern.matches(".*bpel", line1);
352         if(b){
353             System.out.println(line1);
354             XMLHelper_Ran.stringexport(line1+"\n",output);
355             String line2;
356             int count=0;
357             while((line2=br2.readLine())!=null){
358                 if(line2.equals(line1)){
359                     count=0;
360                 }else{
361                     boolean b2=Pattern.matches(".*bpel", line2);
362                     if(b2){
363                         count=0;
364                         break;
365                     }else{
366                         count++;
367                         if(list.contains(count)){
368                             System.out.println("add "+line2);
369                             XMLHelper_Ran.stringexport(line2+"\n",output);
370                         }
371                     }
372                 }
373             }
374         }
375         else{
376             System.out.println("1"+line1);
377             XMLHelper_Ran.stringexport(line1+"\n",output);
378         }
379     }
380 }
381 }
382 package other;
383 import java.io.BufferedReader;
384 import java.io.File;
385 import java.io.FileInputStream;
386 import java.io.InputStreamReader;
387 import java.util.regex.Pattern;
388 import com.test.XMLHelper_Ran;
389 public class SmartShelf_Add {
390     public static void main(String[] args) throws Exception{
391         System.out.println("begin:");
392         String output="E:\\毕设实验\\SmartShelf\\path_temp3333.txt";
393         String path1="E:\\毕设实验\\SmartShelf\\path_temp333.txt";
394         String path2="E:\\毕设实验\\SmartShelf\\path_Add.txt";
395         File file1=new File(path1);
396         InputStreamReader isr1=new InputStreamReader(new FileInputStream(file1));
397         BufferedReader br1=new BufferedReader(isr1);
398         File file2=new File(path2);
399         InputStreamReader isr2=new InputStreamReader(new FileInputStream(file2));
400         BufferedReader br2=new BufferedReader(isr2);
```



```
401     String line1;
402     while((line1=br1.readLine())!=null){
403         boolean b=Pattern.matches(".*bpel", line1);
404         if(b){
405             System.out.println(line1);
406             XMLHelper_Ran.stringexport(line1+"\n",output);
407             String line2;
408             while((line2=br2.readLine())!=null){
409                 if(line2.equals(line1)){
410                     System.out.println("    haha"+line2);
411                 }else{
412                     boolean b2=Pattern.matches(".*bpel", line2);
413                     if(b2){
414                         break;
415                     }else{
416                         System.out.println(line2);
417                         XMLHelper_Ran.stringexport(line2+"\n",output);
418                     }
419                 }
420             }
421         }
422         else{
423             System.out.println(line1);
424             XMLHelper_Ran.stringexport(line1+"\n",output);
425         }
426     }
427 }
428 }
429 package other;
430 import java.io.BufferedReader;
431 import java.io.File;
432 import java.io.FileInputStream;
433 import java.io.InputStreamReader;
434 import java.util.regex.Pattern;
435 import com.test.XMLHelper_Ran;
436 public class Travel_Add {
437     public static void main(String[] args) throws Exception{
438         System.out.println("begin:");
439         String output="E:\\毕设实验\\Travel\\path_new22222.txt";
440         String path1="E:\\毕设实验\\Travel\\path_new2.txt";
441         String path2="E:\\毕设实验\\Travel\\path_Add.txt";
442         File file1=new File(path1);
443         InputStreamReader isr1=new InputStreamReader(new FileInputStream(file1));
444         BufferedReader br1=new BufferedReader(isr1);
445         File file2=new File(path2);
446         InputStreamReader isr2=new InputStreamReader(new FileInputStream(file2));
447         BufferedReader br2=new BufferedReader(isr2);
448         String line1;
449         while((line1=br1.readLine())!=null){
450             boolean b=Pattern.matches(".*bpel", line1);
```

```

451         if(b){
452             System.out.println(line1);
453             XMLHelper_Ran.stringexport(line1+"\n",output);
454             String line2;
455             while((line2=br2.readLine())!=null){
456                 if(line2.equals(line1)){
457                     System.out.println("    haha"+line2);
458                 }else{
459                     boolean b2=Pattern.matches(".*bpe1", line2);
460                     if(b2){
461                         break;
462                     }else{
463                         System.out.println(line2);
464                         XMLHelper_Ran.stringexport(line2+"\n",output);
465                     }
466                 }
467             }
468         }
469         else{
470             System.out.println(line1);
471             XMLHelper_Ran.stringexport(line1+"\n",output);
472         }
473     }
474 }
475 }
476 package lizi;
477 import java.util.ArrayList;
478 import java.util.HashMap;
479 public class SusSetReduce {
480     public HashMap<String,Integer> TruePath=new HashMap<String,Integer>();
481     public HashMap<String,Integer> FalsePath=new HashMap<String,Integer>();
482     public Parse parse;
483     public static ArrayList S=new ArrayList();
484     public void susipion(){
485         S.add(null);
486     }
487     public void predicate(){
488         Node node;
489     }
490     public void atom(){
491         Node node;
492     }
493 }
494 package lizi;
495 import java.io.File;
496 import java.util.ArrayList;
497 public class RunTestCase {
498     Parse parse;
499     public File file=parse.BPELProgram;
500     public boolean deploy(){

```

```
501         return false;
502     }
503     public String sendMessage(ArrayList list){
504         return "";
505     }
506     public boolean compareResult(String s1,String s2){
507         return true;
508     }
509 }
510 package lizi;
511 import java.util.ArrayList;
512 import java.util.HashMap;
513 import java.util.TreeMap;
514 public class Predicate_sort {
515     SusSetReduce pd;
516     public TreeMap<Double, String> map=new TreeMap<Double,String>();
517     public TreeMap<Double, String> sober(HashMap TruePath,HashMap FalsePath){
518         ArrayList list=SusSetReduce.S;
519         return map;
520     }
521 }
522 package lizi;
523 import java.io.File;
524 import java.io.FileNotFoundException;
525 import java.io.IOException;
526 import java.util.ArrayList;
527 import org.dom4j.Document;
528 public class Parse {
529     public File BPELProgram;
530     public ArrayList<Node> list;
531     public File getFile(){
532         return this.BPELProgram;
533     }
534     public boolean readwsdlfile(String filepath) throws FileNotFoundException, IOException {
535         return true;
536     }
537     public void parseWSDL(Document document){
538         Node node;
539     }
540     public Node getChildren(Node node){
541         return node;
542     }
543 }
544 package lizi;
545 import java.util.ArrayList;
546 public class Output {
547     private Predicate_sort s;
548     private ArrayList list;
549     public void result(){
550     }
```

```
551 }
552 package lizi;
553 import java.util.ArrayList;
554 public class Node {
555     private int id;
556     private String name;
557     private String type;
558     private int responseId;
559     private int childNumber;
560     private int beforeNumber;
561     private ArrayList beforeNodes = new ArrayList();
562     private ArrayList afterNodes = new ArrayList();
563     private ArrayList childNodes = new ArrayList();
564     Node(){
565     }
566     public int getBeforeNumber() {
567         return beforeNumber;
568     }
569     public void setBeforeNumber(int beforeNumber) {
570         this.beforeNumber = beforeNumber;
571     }
572     public ArrayList getChildNodes() {
573         return childNodes;
574     }
575     public void setChildNodes(ArrayList childNodes) {
576         this.childNodes = childNodes;
577     }
578     public int getId() {
579         return id;
580     }
581     public void setId(int id) {
582         this.id = id;
583     }
584     public String getName() {
585         return name;
586     }
587     public void setName(String name) {
588         this.name = name;
589     }
590     public String getType() {
591         return type;
592     }
593     public void setType(String type) {
594         this.type = type;
595     }
596     public int getResponseId() {
597         return responseId;
598     }
599     public void setResponseId(int responseId) {
600         this.responseId = responseId;
```

```
601     }
602     public int getChildNumber() {
603         return childNumber;
604     }
605     public void setChildNumber(int childNumber) {
606         this.childNumber = childNumber;
607     }
608     public ArrayList getBeforeNodes() {
609         return beforeNodes;
610     }
611     public void setBeforeNodes(ArrayList beforeNodes) {
612         this.beforeNodes = beforeNodes;
613     }
614     public ArrayList getAfterNodes() {
615         return afterNodes;
616     }
617     public void setAfterNodes(ArrayList afterNodes) {
618         this.afterNodes = afterNodes;
619     }
620     public String toString() {
621         /*String ret = this.getId() + " " + this.getResponseId() + " "
622             + this.getAfterNodes().size() + " " + this.getType() + " "
623             + this.getName() + " "+this.getPredict();
624         conditions = this.getConditions();
625         return ret;
626     }
627     public int getbeforeNumber(Node node) {
628         return node.beforeNodes.size();
629     }
630 }
631 package com.test;
632 import java.io.File;
633 import java.io.FileNotFoundException;
634 import java.io.FileOutputStream;
635 import java.io.FileWriter;
636 import java.io.IOException;
637 import java.io.OutputStream;
638 import java.io.OutputStreamWriter;
639 import java.io.Writer;
640 import java.util.ArrayList;
641 import java.util.HashMap;
642 import java.util.Iterator;
643 import java.util.Map;
644 import java.util.Map.Entry;
645 import java.util.Set;
646 import org.dom4j.Attribute;
647 import org.dom4j.Document;
648 import org.dom4j.Element;
649 import org.dom4j.io.OutputFormat;
650 import org.dom4j.io.SAXReader;
```

```
651 import org.dom4j.io.XMLWriter;
652 import com.test.bpelbean.Message;
653 import com.test.bpelbean.Value;
654 public class XMLHelper_Ran {
655     public static HashMap<String, ArrayList<String>> hashMap = new HashMap<String,
656     ArrayList<String>>();
657     public static HashMap<String, String> hashBpelHashMap = new HashMap<String,
658     String>();
659     public static HashMap<String, ArrayList<String>> finalHashMap = new HashMap<String,
660     ArrayList<String>>();
661     public static void main(String args[]) throws FileNotFoundException, IOException {
662         String bpelPath = "D:\\Workspace1\\DebugTest\\bpel\\SmartShelfProcess.bpel";
663         readwsdlfile(bpelPath);
664         Set set=hashMap.entrySet();
665         for(Object o:set){
666             Map.Entry entry=(Map.Entry)o;
667             String key=(String) entry.getKey();
668             ArrayList list=(ArrayList) entry.getValue();
669             System.out.println(key+" "+list);
670         }
671     }
672     public static Document openXMLFile(String filePath) {
673         Document document = null;
674         SAXReader reader = new SAXReader();
675         try {
676             File file = new File(filePath);
677             document = reader.read(file);
678         } catch (Exception e) {
679             e.printStackTrace();
680         }
681         return document;
682     }
683     public static boolean writeXMLFile(Document document, String filePath) {
684         boolean flag = false;
685         XMLWriter writer = null;
686         OutputFormat format = OutputFormat.createPrettyPrint();
687         format.setEncoding("UTF-8");
688         try {
689             File file = new File(filePath);
690             writer = new XMLWriter(new FileWriter(file), format);
691             writer.write(document);
692             writer.flush();
693             writer.close();
694             flag = true;
695         } catch (Exception e) {
696             flag = false;
697             e.printStackTrace();
698         }
699         return flag;
700     }
```

```
701 public static boolean readwsdlfile(String filepath)
702     throws FileNotFoundException, IOException {
703     try {
704         File file = new File(filepath);
705         if (!file.isDirectory()) {
706             System.out.println("absolutePath=" + file.getAbsolutePath());
707             System.out.println("name=" + file.getName());
708             String absolutePath = file.getAbsolutePath();
709             Document document = openXMLFile(absolutePath);
710             parseWSDL(document);
711         } else if (file.isDirectory()) {
712             String[] fileList = file.list();
713             for (int i = 0; i < fileList.length; i++) {
714                 File readfile = new File(filepath + "\\" + fileList[i]);
715                 if (!readfile.isDirectory()) {
716                     if (fileList[i].contains(".wsdl")) {
717                         String absolutePath = readfile.getAbsolutePath();
718                         Document document = openXMLFile(absolutePath);
719                         parseWSDL(document);
720                     }
721                 } else if (readfile.isDirectory()) {
722                     readwsdlfile(filepath + "\\" + fileList[i]);
723                 }
724             }
725         }
726     } catch (FileNotFoundException e) {
727         System.out.println("readfile() Exception:" + e.getMessage());
728     }
729     return true;
730 }
731 public static void parseWSDL(Document document) {
732     Element root = document.getRootElement();
733     ArrayList<Message> arrayList = new ArrayList<Message>();
734     ArrayList<Value> values = new ArrayList<Value>();
735     for (Iterator iter = root.elementIterator(); iter.hasNext();) {
736         Element element = (Element) iter.next();
737         if (element.getName().equals("message")) {
738             Message message = new Message();
739             Attribute attribute = element.attribute("name");
740             String key = attribute.getValue();
741             Element partElement = element.element("part");
742             Attribute elAttribute = partElement.attribute("element");
743             String refString = elAttribute.getValue();
744             String a[] = refString.split(":");
745             message.setName(key);
746             message.setRefsString(a[1]);
747             arrayList.add(message);
748         } else if (element.getName().equals("types")) {
749             Element tyElement = element.element("schema");
750             for (Iterator iterator = tyElement.elementIterator(); iterator
```

```

751         .hasNext();) {
752         Value value = new Value();
753         Element elementOut = (Element) iterator.next();
754         String refName = elementOut.attribute("name").getValue();
755         ArrayList<String> arrayList2 = new ArrayList<String>();
756         Element elementSequence = elementOut.element("complexType")
757             .element("sequence");
758         if (elementSequence != null) {
759             for (Iterator iterator2 = elementSequence
760                 .elementIterator(); iterator2.hasNext();) {
761                 Element elementInner = (Element) iterator2.next();
762                 String value1 = elementInner.attribute("name")
763                     .getValue();
764                 arrayList2.add(value1);
765             }
766         }
767         value.setRefName(refName);
768         value.setValue(arrayList2);
769         values.add(value);
770     }
771 }
772 }
773 addtoHashMap(arrayList, values);
774 }
775 private static void addtoHashMap(ArrayList<Message> arrayList,
776     ArrayList<Value> values) {
777     for (int i = 0; i < arrayList.size(); i++) {
778         String name = arrayList.get(i).getName();
779         String refname = arrayList.get(i).getRefsString();
780         for (int j = 0; j < values.size(); j++) {
781             String reString = values.get(j).getRefName();
782             if (refname.equals(reString)) {
783                 hashMap.put(name, values.get(j).getValue());
784             }
785         }
786     }
787 }
788 public static void parseBpelVariable(String filePath) {
789     Document document = openXMLFile(filePath);
790     Element root = document.getRootElement();
791     Element vsElement = root.element("variables");
792     for (Iterator iterator = vsElement.elementIterator(); iterator
793         .hasNext();) {
794         Element vElement = (Element) iterator.next();
795         String messageType = null;
796         if (vElement.attribute("messageType") != null) {
797             messageType = vElement.attribute("messageType").getValue();
798             messageType = messageType.split(":")[1];
799         }
800

```



```

801         String name = vElement.attribute("name").getValue();
802         hashBpelHashMap.put(messageType, name);
803     }
804     bpelWsdIParse();
805 }
806 public static void bpelWsdIParse() {
807     for (Entry<String, ArrayList<String>> entry : hashMap.entrySet()) {
808         String meString = entry.getKey();
809         ArrayList<String> valueList = entry.getValue();
810         for (Entry<String, String> entry2 : hashBpelHashMap.entrySet()) {
811             String meString2 = entry2.getKey();
812             String nameString = entry2.getValue();
813             if (meString.equals(meString2)) {
814                 finalHashMap.put(nameString, valueList);
815                 System.out.println(nameString+" "+valueList+"|!!!!!!!!!!!!!!!!!!!!");
816             }
817         }
818     }
819 }
820 public static boolean stringexport(String parameter1, String path) throws Exception {
821     boolean flag = false;
822     OutputStream os;
823     try {
824         os = new FileOutputStream(new File(path), true);
825         Writer fos = new OutputStreamWriter(os);
826         fos.write(parameter1);
827         fos.flush();
828         fos.close();
829         flag = true;
830     } catch (FileNotFoundException e) {
831         e.printStackTrace();
832     }
833     return flag;
834 }
835 }
836 package com.test;
837 import java.util.Calendar;
838 import org.apache.axiom.om.OMAbstractFactory;
839 import org.apache.axiom.om.OMElement;
840 import org.apache.axiom.om.OMFactory;
841 import org.apache.axiom.om.OMNamespace;
842 import org.apache.axis2.AxisFault;
843 import org.apache.axis2.addressing.EndpointReference;
844 import org.apache.axis2.client.Options;
845 import org.apache.axis2.client.ServiceClient;
846 public class TravelClient
847 {
848     public synchronized static OMElement sendmessage(String string, int amount) throws
849     Exception
850     {

```

```
851     OMElement res=null;
852     ServiceClient sc=null;
853     try {
854         sc = new ServiceClient();
855         Options opts = new Options();
856         opts.setTo(new EndpointReference(
857             "http://localhost:8080/ode/processes/TravelAgency"));
858         opts.setAction("www.ustb.edu.cn/bpel/travelagency/process");
859         sc.setOptions(opts);
860         long startTime = Calendar.getInstance().getTimeInMillis();
861         res = sc.sendReceive(createPayLoad(string,amount));
862         long endTime = Calendar.getInstance().getTimeInMillis();
863         System.out.println(string+"&&" + amount);
864         System.out.println(res);
865
866     } catch (AxisFault e) {
867         e.printStackTrace();
868     } finally{
869         sc.cleanupTransport();
870     }
871     return res;
872 }
873 public static OMElement createPayLoad(String string,int parameter2){
874     OMFactory fac = OMAbstractFactory.getOMFactory();
875     OMNamespace omNs =
876     fac.createOMNamespace("www.ustb.edu.cn/bpel/travelagency", "ustb");
877     OMNamespace omNs1 =
878     fac.createOMNamespace("www.ustb.edu.cn/bpel/travelagency", "ustb");
879     OMElement method = fac.createOMElement("TravelAgencyRequest",omNs);
880     OMElement value1 = fac.createOMElement("name",omNs1);
881     OMElement value2 = fac.createOMElement("amount",omNs1);
882     value1.setText(string+ "");
883     value2.setText(parameter2+ "");
884     method.addChild(value1) ;
885     method.addChild(value2) ;
886     System.out.println(method);
887     return method;
888 } ;
889 }
890 package com.test;
891 import java.util.HashMap;
892 import java.util.Iterator;
893 import java.util.Map;
894 import org.dom4j.Document;
895 import org.dom4j.Element;
896 public class StringMapper {
897     private static Map<String, String> stringMap = new HashMap<String, String>();
898     static {
899         Document document = XMLHelper_Ran.openXMLFile("string_en.xml");
900         Element root = document.getRootElement();
```

```
901         for (Iterator iter = root.elementIterator(); iter.hasNext();) {
902             Element element = (Element) iter.next();
903             String name = null;
904             String value = null;
905             for (Iterator ii = element.elementIterator(); ii.hasNext();) {
906                 Element e = (Element) ii.next();
907                 if (e.getName().equals("name")) {
908                     name = e.getText();
909                 } else if (e.getName().equals("value")) {
910                     value = e.getText();
911                 }
912             }
913             if (name != null && value != null) {
914                 stringMap.put(name, value);
915             }
916         }
917     }
918     public static String get(String name) {
919         return stringMap.get(name);
920     }
921 }
922 package com.test;
923 import com.test.update.InventoryUpdate;
924 import com.test.update.ProductUpdate;
925 import com.test.update.ShelfUpdate;
926 import com.test.update.WarehouseUpdate;
927 public class smartUpdate {
928     public static void update(){
929         InventoryUpdate inventory=new InventoryUpdate();
930         inventory.inventoryupdate();
931         ProductUpdate product=new ProductUpdate();
932         product.productupdate();
933         ShelfUpdate shelf=new ShelfUpdate();
934         try {
935             shelf.shelfupdate();
936         } catch (Exception e) {
937             e.printStackTrace();
938         }
939         WarehouseUpdate warehouse=new WarehouseUpdate();
940         try {
941             warehouse.warehouseupdate();
942             Thread.sleep(1000);
943         } catch (Exception e) {
944             e.printStackTrace();
945         }
946     }
947 }
948 package com.test;
949 import java.util.Calendar;
950 import org.apache.axiom.om.OMAbstractFactory;
```

```

951 import org.apache.axiom.om.OMElement;
952 import org.apache.axiom.om.OMFactory;
953 import org.apache.axiom.om.OMNamespace;
954 import org.apache.axis2.AxisFault;
955 import org.apache.axis2.addressing.EndpointReference;
956 import org.apache.axis2.client.Options;
957 import org.apache.axis2.client.ServiceClient;
958 public class quoteprocessclient
959 {
960     public synchronized static OMElement sendmessage(String name, int amount) throws
961     Exception
962     {
963         OMElement res=null;
964         ServiceClient sc =null;
965         try {
966             sc = new ServiceClient();
967             Options opts = new Options();
968             opts.setTo(new EndpointReference(
969                 "http://localhost:8080/ode/processes/QuoteProcess"));
970             opts.setAction("www.ustb.edu.cn/bpel/quoteprocess/process");
971             sc.setOptions(opts);
972             long startTime = Calendar.getInstance().getTimeInMillis();
973             res = sc.sendReceive(createPayLoad(name,amount));
974             long endTime = Calendar.getInstance().getTimeInMillis();
975             System.out.println(name+"&&" +amount);
976             System.out.println(res);
977         } catch (AxisFault e) {
978             e.printStackTrace();
979         } finally{
980             sc.cleanupTransport();
981         }
982         return res;
983     }
984     public static OMElement createPayLoad(String parameter1,int parameter2){
985         OMFactory fac = OMAbstractFactory.getOMFactory();
986         OMNamespace omNs =
987         fac.createOMNamespace("www.ustb.edu.cn/bpel/quoteprocess", "ustb");
988         OMNamespace omNs1 =
989         fac.createOMNamespace("www.ustb.edu.cn/bpel/quoteprocess", "ustb");
990         OMElement method = fac.createOMElement("QuoteProcessRequest",omNs);
991         OMElement value1 = fac.createOMElement("Name",omNs1);
992         OMElement value2 = fac.createOMElement("Amount",omNs1);
993         value1.setText(parameter1);
994         value2.setText(parameter2+"");
995         method.addChild(value1) ;
996         method.addChild(value2) ;
997         System.out.println("????:"+method);
998         return method;
999     }
1000 }

```

```
1001 package com.test;
1002 import java.util.ArrayList;
1003 import java.util.Iterator;
1004 import java.util.List;
1005 import org.apache.axiom.om.OMElement;
1006 import org.apache.axiom.om.OMNode;
1007 public class OmelementParse
1008 {
1009     public static List<String> getresult(OMElement element)
1010     {
1011         if (element == null){
1012             return null;
1013         }
1014         Iterator iterator = element.getChildElements();
1015         List<String> list = new ArrayList<String>();
1016         while (iterator.hasNext())
1017         {
1018             OMNode omNode = (OMNode) iterator.next();
1019             if (omNode.getType() == OMNode.ELEMENT_NODE){
1020                 OMElement omElement = (OMElement) omNode;
1021                 String temp=omElement.getText().trim();
1022                 list.add(temp);
1023             }
1024         }
1025         return list;
1026     }
1027 }
1028 package com.test;
1029 import java.util.Calendar;
1030 import org.apache.axiom.om.OMAbstractFactory;
1031 import org.apache.axiom.om.OMElement;
1032 import org.apache.axiom.om.OMFactory;
1033 import org.apache.axiom.om.OMNamespace;
1034 import org.apache.axis2.AxisFault;
1035 import org.apache.axis2.addressing.EndpointReference;
1036 import org.apache.axis2.client.Options;
1037 import org.apache.axis2.client.ServiceClient;
1038 public class MessageClient
1039 {
1040     public synchronized static OMElement sendmessage(String name, int amount) throws
1041     Exception
1042     {
1043         OMElement res=null;
1044         ServiceClient sc=null;
1045         try {
1046             sc = new ServiceClient();
1047             Options opts = new Options();
1048             opts.setTo(new EndpointReference(
1049                 "http://localhost:8080/ode/processes/SmartShelfProcess"));
1050             opts.setAction("ustb.bpel.org/process");
```

```
1051         sc.setOptions(opts);
1052         long startTime = Calendar.getInstance().getTimeInMillis();
1053         res = sc.sendReceive(createPayLoad(name,amount));
1054         long endTime = Calendar.getInstance().getTimeInMillis();
1055         System.out.println(name+"&&"+"amount");
1056         System.out.println("SmartShelf-----"+res);
1057     } catch (AxisFault e) {
1058         e.printStackTrace();
1059     }finally{
1060         sc.cleanupTransport ();
1061     }
1062     return res;
1063 }
1064 public static OMElement createPayLoad(String parameter1,int parameter2){
1065     OMFactory fac = OMAbstractFactory.getOMFactory();
1066     OMNamespace omNs = fac.createOMNamespace("ustb.bpel.org", "ustb");
1067     OMNamespace omNs1 = fac.createOMNamespace("ustb.bpel.org", "ustb");
1068     OMElement method = fac.createOMElement("SmartShelfProcessRequest",omNs);
1069     OMElement value1 = fac.createOMElement("name",omNs1);
1070     OMElement value2 = fac.createOMElement("amount",omNs1);
1071     value1.setText(parameter1);
1072     value2.setText(parameter2+"");
1073     method.addChild(value1) ;
1074     method.addChild(value2) ;
1075     System.out.println(method);
1076     return method;
1077 } ;
1078 }
1079 package com.test;
1080 import java.io.File;
1081 public class getFiles {
1082     public static void main(String[] args) {
1083         File files = new File(
1084             "C:\\Users\\Administrator\\Desktop\\实验\\travelmutant\\");
1085         File[] filess = files.listFiles();
1086         for (int i = 0; i < filess.length; i++){
1087             System.out.println(filess[i]);
1088         }
1089         System.out.println(filess.length);
1090     }
1091 }
1092 package com.test;
1093 import java.io.BufferedInputStream;
1094 import java.io.BufferedOutputStream;
1095 import java.io.BufferedReader;
1096 import java.io.File;
1097 import java.io.FileInputStream;
1098 import java.io.FileNotFoundException;
1099 import java.io.FileReader;
1100 import java.io.IOException;
```

```
1101 import java.io.PrintStream;
1102 import java.io.RandomAccessFile;
1103 import java.io.Reader;
1104 import java.util.ArrayList;
1105 import java.util.HashMap;
1106 import javax.swing.JTextArea;
1107 import com.test.bpelbean.TestcaseNode;
1108 public class FileControl {
1109     public static String re=null;
1110     public static Boolean deletefile( String deployname, String abstractfilename) throws
1111     InterruptedException {
1112         boolean flag = false;
1113         String path = "D:/Programs/tomcat 6old/webapps/ode/WEB-INF/processes";
1114         File directory=new File(path);
1115         if (!directory.isDirectory()) {
1116             System.out.println("删除文件夹找不到");
1117             flag = false;
1118         }else{
1119             flag=parse(directory,deployname,abstractfilename);
1120         }
1121         return flag;
1122     }
1123     public static void readFileByLines(String fileName, JTextArea expectArea) {
1124         File file = new File(fileName);
1125         BufferedReader reader = null;
1126         try {
1127             System.out.println("以行为单位读取文件内容，一次读一整行：");
1128             reader = new BufferedReader(new FileReader(file));
1129             String tempString = null;
1130             int line = 1;
1131             while ((tempString = reader.readLine()) != null) {
1132                 expectArea.append(tempString+"\r\n");
1133                 System.out.println("line " + line + ": " + tempString);
1134                 line++;
1135             }
1136             reader.close();
1137         } catch (IOException e) {
1138             e.printStackTrace();
1139         } finally {
1140             if (reader != null) {
1141                 try {
1142                     reader.close();
1143                 } catch (IOException e1) {
1144                 }
1145             }
1146         }
1147     }
1148     public static void readFileByLines(String fileName,HashMap<String,Integer> Path,String
1149     flag,JTextArea expectArea) {
1150         File file = new File(fileName);
```

```
1151     BufferedReader reader = null;
1152     try {
1153         System.out.println("以行为单位读取文件内容，一次读一整行：");
1154         reader = new BufferedReader(new FileReader(file));
1155         String tempString = null;
1156         int line = 1;
1157         while ((tempString = reader.readLine()) != null) {
1158             if(tempString.contains(flag)){
1159                 if(Path.containsKey(tempString)){
1160                     Path.put(tempString, Path.get(tempString)+1);
1161                 }else{
1162                     Path.put(tempString, 1);
1163                 }
1164                 expectArea.append(tempString+"\r\n");
1165                 System.out.println("line " + line + ": " + tempString);
1166                 line++;
1167             }
1168         }
1169         reader.close();
1170     } catch (IOException e) {
1171         e.printStackTrace();
1172     } finally {
1173         if (reader != null) {
1174             try {
1175                 reader.close();
1176             } catch (IOException e1) {
1177             }
1178         }
1179     }
1180 }
1181 public static ArrayList readFile(String fileName, JTextArea expectArea) {
1182     File file = new File(fileName);
1183     BufferedReader reader = null;
1184     ArrayList list=new ArrayList();
1185     try{
1186         System.out.println("以行为单位读取文件内容，一次读一整行：");
1187         reader = new BufferedReader(new FileReader(file));
1188         String tempString = null;
1189         int line = 1;
1190         tempString=reader.readLine();
1191         System.out.println(reader.readLine());
1192         System.out.println(tempString);
1193         String[] str=tempString.split("#");
1194         list.add("entry");
1195         list.add("receiveInput");
1196         expectArea.append("entry");
1197         expectArea.append("receiveInput");
1198         for(int i=0;i<str.length;i++){
1199             expectArea.append(str[i]+" ");
1200             list.add(str[i]);
```



```
1201         if(i%5==0){
1202             expectArea.append("\n");
1203         }
1204     }
1205     expectArea.append("replyOutput");
1206     list.add("replyOutPut");
1207 }catch (IOException e) {
1208     e.printStackTrace();
1209 }finally {
1210     if (reader != null) {
1211         try {
1212             reader.close();
1213         } catch (IOException e1) {
1214             }
1215         }
1216     }
1217     return list;
1218 }
1219 public static ArrayList readFileEndLine(String fileName) {
1220     File file = new File(fileName);
1221     BufferedReader reader = null;
1222     ArrayList list=new ArrayList();
1223     try {
1224         System.out.println("以行为单位读取文件内容，一次读一整行：");
1225         reader = new BufferedReader(new FileReader(file));
1226         String tempString = null;
1227         int line = 1;
1228         String s=null;
1229         while((s=reader.readLine())!=null){
1230             s=tempString;
1231         }
1232         System.out.println(tempString);
1233         String[] str=tempString.split("#");
1234         list.add("entry");
1235         list.add("receiveInput");
1236         for(int i=0;i<str.length;i++){
1237             list.add(str[i]);
1238         }
1239         list.add("replyOutPut");
1240     } catch (IOException e) {
1241         e.printStackTrace();
1242     } finally {
1243         if (reader != null) {
1244             try {
1245                 reader.close();
1246             } catch (IOException e1) {
1247                 }
1248         }
1249     }
1250     return list;
```

```

1251     }
1252     public static void parseWsdPath(String deployname,File directory){
1253         if(directory.isDirectory()){
1254             File[] files=directory.listFiles();
1255             for(int i=0;i<files.length;i++){
1256                 if(files[i].isFile()){
1257                     String curName=files[i].getName();
1258                     if(curName.equals(deployname)){
1259                         System.out.println(files[i].getParentFile().getAbsolutePath());
1260                         re=files[i].getParentFile().getAbsolutePath();
1261                     }
1262                 }else{
1263                     File directorynew=new File(directory+"/"+files[i].getName());
1264                     parseWsdPath(deployname,directorynew);
1265                 }
1266             }
1267         }
1268     }
1269     private static boolean parse(File directory, String deletename,String abstractfilename) throws
1270     InterruptedException {
1271         boolean flag=false;
1272         if(directory.isDirectory()){
1273             File[] files = directory.listFiles();
1274             for (int i = 0; i < files.length; i++) {
1275                 if (files[i].isFile()) {
1276                     String oldname = files[i].getName();
1277                     if (oldname.equals(deletename)) {
1278                         files[i].delete();
1279                         Thread.sleep(1000);
1280                         try {
1281                             copyfile(files[i].getAbsolutePath(),abstractfilename);
1282                         } catch (Exception e) {
1283                             e.printStackTrace();
1284                         }
1285                         System.out.println("删除 bpel 文件成功");
1286                         String path = "D:/Programs/tomcat
1287old/webapps/ode/WEB-INF/processes";
1288                         File odepath=new File(path);
1289                         File[] outer=odepath.listFiles();
1290                         for(int j=0;j<outer.length;j++){
1291                             System.out.println(outer[j].getName());
1292                             if(outer[j].isFile()){
1293                                 String
1294deployname=files[i].getParentFile().getName()+".deployed";
1295                                 System.out.println(outer[j].getName()+" "+deployname);
1296                                 if(deployname.equals(outer[j].getName())){
1297                                     System.out.println("删除.deployed 文件成功");
1298                                     outer[j].delete();
1299                                     Thread.sleep(100);
1300                                     flag=true;

```

```

1301         }
1302     }
1303 }
1304 }
1305     }else {
1306         File directorynew=new File(directory+"/"+files[i].getName());
1307         parse(directorynew, deletename, abstractfilename);
1308     }
1309 }
1310 }
1311 return flag;
1312 }
1313 public static Boolean copyfile(String topath, String frompath) throws Exception
1314 {
1315     boolean flag=false;
1316     System.out.println(topath+"~~~~~"+frompath);
1317     File oldfile=new File(topath);
1318     File newfile=new File(frompath);
1319     System.out.println(newfile.exists()+"^^^^^^^^"+newfile.isFile());
1320     if(oldfile.exists()){
1321         oldfile.delete();
1322     }
1323     oldfile.createNewFile();
1324     if((!newfile.exists())||(!newfile.isFile()))
1325     {
1326         System.out.println("路径不是一个 directory 或者文件夹不存在");
1327         flag=false;
1328     }else{
1329         FileInputStream fin = new FileInputStream(newfile.getAbsolutePath());
1330         BufferedInputStream bin = new BufferedInputStream(fin);
1331         PrintStream pout = new PrintStream(oldfile);
1332         BufferedOutputStream bout = new BufferedOutputStream(pout);
1333         int total =bin.available();
1334         int count;
1335         while((count = bin.available())!= 0)
1336         {
1337             int c = bin.read();
1338             bout.write((char)c);
1339         }
1340         bout.close();
1341         pout.close();
1342         bin.close();
1343         fin.close();
1344         flag=true;
1345     }
1346     return flag;
1347 }
1348 public static String comparefile(String oldpath, String newpath) throws Exception
1349 {
1350     XMLHelper_Ran export = new XMLHelper_Ran();
    String result="";

```

```
1351     boolean flag=false;
1352     File oldfile=new File(oldpath);
1353     File newfile=new File(newpath);
1354     if(!oldfile.isFile())
1355     {
1356         System.out.println("源文件不是一个文件");
1357         flag=false;}
1358     if(!newfile.isFile())
1359     {
1360         System.out.println("对比文件不是一个文件");
1361         flag=false;
1362     }
1363     try {
1364         Reader fin1 = new FileReader(oldfile);
1365         Reader fin2 = new FileReader(newfile);
1366         BufferedReader bin1 = new BufferedReader(fin1);
1367         BufferedReader bin2 = new BufferedReader(fin2);
1368         String s1=null;
1369         String s2=null;
1370         while(((s1=bin1.readLine())!=null))
1371         {
1372             s2=bin2.readLine();
1373             if(!s1.equals(s2)) {
1374                 XMLHelper_Ran.stringexport(newfile.getAbsolutePath()+"\n",
1375                 "C:\\Users\\Administrator\\Desktop\\实验
1376 \\smarshelfSlicing\\smartcompare1.txt");
1377                 XMLHelper_Ran.stringexport("原始版本:  "+s1+"\n"+"故障版本:
1378 "+s2+"\n",
1379                 "C:\\Users\\Administrator\\Desktop\\实验
1380 \\smarshelfSlicing\\smartcompare1.txt");
1381                 XMLHelper_Ran.stringexport("\n\n",
1382                 "C:\\Users\\Administrator\\Desktop\\实验
1383 \\smarshelfSlicing\\smartcompare1.txt");
1384                 result="different";
1385                 break;
1386             }else
1387             {
1388                 result="same";
1389             }
1390         }
1391     } catch (FileNotFoundException e) {
1392         e.printStackTrace();
1393     }
1394     return result;
1395 public TestcaseNode gettestcase(String testcasepath)
1396 {
1397     String line = "";
1398     TestcaseNode testcasenode = null;
1399     String[] temp1;
1400     int[] temp = null;
```

```
1401         @SuppressWarnings("rawtypes")
1402         ArrayList templist = new ArrayList();
1403         try {
1404             RandomAccessFile rf = new RandomAccessFile(testcasepath, "rw");
1405             try {
1406                 for (line = rf.readLine(); line != null; line = rf.readLine()) {
1407                     temp1 = line.split("#");
1408                     for(int i=0;i<temp1.length;i++)
1409                         temp[i]=Integer.parseInt(temp1[i]);
1410                     System.out.println("temp'length:" + temp.length);
1411                     if(temp.length==2)
1412                     {
1413                     }
1414                 }
1415             } catch (IOException e) {
1416                 e.printStackTrace();
1417             }
1418         } catch (FileNotFoundException e) {
1419             e.printStackTrace();
1420         }
1421         return testcasenode;
1422     }
1423 }
1424 package com.test;
1425 import java.io.IOException;
1426 public class EngineImpl implements Engine {
1427     public boolean deploy(String file1,String abstractfilename) throws InterruptedException {
1428         boolean flag=false;
1429         System.out.println(abstractfilename.endsWith("."));
1430         flag=FileControl.deletefile(file1,abstractfilename);
1431         return flag;
1432     }
1433     public boolean start() {
1434         String line = "";
1435         boolean flag=false;
1436         Runtime runtime = Runtime.getRuntime();
1437         try {
1438             Process process = runtime.exec("D://Programs//tomcat 6old//bin//startup.bat");
1439             flag=true;
1440         } catch (IOException e) {
1441             e.printStackTrace();
1442         }
1443         return flag;
1444     }
1445 }
1446 package com.test;
1447 public interface Engine {
1448     public boolean start();
1449     public boolean deploy(String file1, String abstractfilename) throws InterruptedException;
1450 }
```

```
1451 package com.test;
1452 import java.util.Iterator;
1453 import org.dom4j.Attribute;
1454 import org.dom4j.Element;
1455 public class ElementWrapper {
1456     Element element;
1457     private String xPath;
1458     public ElementWrapper() {
1459     }
1460     public ElementWrapper(Element element) {
1461         this.element = element;
1462     }
1463     public Element getElement() {
1464         return element;
1465     }
1466     public void setElement(Element element) {
1467         this.element = element;
1468     }
1469     public String getXPath() {
1470         return this.xPath;
1471     }
1472     public void setXPath(String xPath) {
1473         this.xPath = xPath;
1474     }
1475     public String toString() {
1476         StringBuffer sb = new StringBuffer();
1477         sb.append(element.getName() + " ");
1478         for (Iterator iter = element.attributeIterator(); iter.hasNext();) {
1479             Attribute attribute = (Attribute) iter.next();
1480             sb.append(attribute.getName()).append("=")
1481                 .append(attribute.getValue()).append(" ");
1482         }
1483         return sb.toString();
1484     }
1485 }
1486 package com.test;
1487 import java.io.BufferedReader;
1488 import java.io.BufferedWriter;
1489 import java.io.File;
1490 import java.io.FileNotFoundException;
1491 import java.io.FileReader;
1492 import java.io.FileWriter;
1493 import java.io.IOException;
1494 import java.io.Reader;
1495 public class DeleteNull {
1496     public static void main(String[] args) {
1497         File oldfile = new File("C:\\Users\\Administrator\\Desktop\\实验
1498             try {
1499                 Reader fin1 = new FileReader(oldfile);
1500                 BufferedReader reader = new BufferedReader(fin1);
```