

ecm1410_coursework\src\socialmedia\SocialMedia.java

```
1 package socialmedia;
2
3 import java.io.IOException;
4 import java.io.File;
5 import java.io.FileWriter;
6 import java.util.*;
7
8 import java.io.BufferedWriter;
9 import java.io.BufferedReader;
10 import java.io.FileReader;
11
12
13
14
15 public class SocialMedia implements SocialMediaPlatform {
16
17     ArrayList<Account> accounts = new ArrayList<>(); //List of Accounts
18     ArrayList<FeedDenizen> orphanage = new ArrayList<>(); // Comments where the post
    has been deleted
19
20
21     //Returns an Account object of an account from an ID given
22     public Account findAccountFromID(int id) throws AccountIDNotRecognisedException{
23
24         for (Account i : accounts){
25             if(i.getID() == id){
26                 return i;
27             }
28         }
29         throw new AccountIDNotRecognisedException();
30     }
31
32     //Returns an Account object of an account from a given Handle
33     public Account findAccountFromHandle(String handle) throws
    HandleNotRecognisedException{
34
35         for (Account i : accounts){
36             if(handle.equals(i.getHandle())){
37                 return i;
38             }
39         }
40         throw new HandleNotRecognisedException();
41     }
42
43
44     //Returns an Account object of an account from an ID given, only to be used if
    account ID is known to exist
45     public Account findAccountFromKnownID(int id){
46
47         for (Account i : accounts){
48             if(i.getID() == id){
49                 return i;
50             }
51         }
52         return new Account(0, null, null);
53     }
```

```

54
55
56 // Unpacks an ID into Account ID and Post ID
57 public int[] unpackID(int id){
58
59     int accountID = (int)Math.floor(id/10000);
60     int postID = id - (accountID * 10000);
61     int[] result = {accountID, postID};
62     return result;
63 }
64
65
66 // Returns a post from a given ID
67 public FeedDenizen findPostFromID(int id) throws PostIDNotRecognisedException{
68
69     int[] ids = unpackID(id);
70     Account account = findAccountFromKnownID(ids[0]);
71     FeedDenizen post = account.getPost(ids[1]);
72     return post;
73 }
74
75 // Returns a post from a given ID, only to be used if account ID is known to exist
76 public FeedDenizen findPostFromKnownID(int id){
77
78     int[] ids = unpackID(id);
79     Account account = findAccountFromKnownID(ids[0]);
80     FeedDenizen post = account.getKnownPost(ids[1]);
81     return post;
82 }
83
84
85 // Creates an account with a blank description by calling createAccount with a
blank description
86 @Override
87 public int createAccount(String handle) throws IllegalHandleException,
InvalidHandleException {
88
89     return createAccount(handle, "");
90 }
91
92 // Validates if the handle is unique and the description follows the rules, and if
so creates an account with the next ID and adds it to the account list, then returns
the ID
93 @Override
94 public int createAccount(String handle, String description) throws
IllegalHandleException, InvalidHandleException {
95
96     if (!(0 < handle.length() && handle.length() < 31)){
97         throw new InvalidHandleException();
98     }
99     char[] array = handle.toCharArray();
100     for (char i : array ){
101         int ascii = i;
102         if (!(64 < ascii && ascii < 123)){
103             throw new InvalidHandleException();
104         }
105     }
106     for (Account i : accounts){
107         if (handle.equals(i.getHandle())) {
108             throw new IllegalHandleException();

```

```

109         }
110     }
111     int next_index = accounts.size();
112     accounts.add(new Account(next_index, handle, description));
113     return next_index;
114 }
115
116 // Calls findAccountFromID, if an account is found, calls the delete method on that
117 // account, and stores all the children post in the orphanage list, then removes the
118 // account from the account list
119 @Override
120 public void removeAccount(int id) throws AccountIDNotRecognisedException {
121     Account i = findAccountFromID(id);
122     List<Integer> orphans = i.delete();
123     for (Integer childID : orphans){
124         orphanage.add(findPostFromKnownID(childID));
125     }
126     accounts.remove(i);
127 }
128
129 // Finds an account and adds all the comments of this account's posts to the
130 // orphanage then removes the account and calls the delete account method
131 @Override
132 public void removeAccount(String handle) throws HandleNotRecognisedException {
133     Account i = findAccountFromHandle(handle);
134     List<Integer> orphans = i.delete();
135     for (Integer childID : orphans){
136         orphanage.add(findPostFromKnownID(childID));
137     }
138     accounts.remove(i);
139 }
140
141 // Validates the handle is legal, then iterates through the list of accounts and
142 // checks their handle.
143 // If the handle matches the new handle, the new handle can't be used so
144 // IllegalArgumentException is Thrown
145 // If the handle matches the old handle, this is the account we want to change the
146 // handle of so we store the index for later
147 // Once the loop is done, if an index was found for the account, the handle is
148 // changed, else it throws HandleNotRecognisedException
149 @Override
150 public void changeAccountHandle(String oldHandle, String newHandle)
151     throws HandleNotRecognisedException, IllegalArgumentException,
152     InvalidHandleException {
153     if (!(0 < newHandle.length() && newHandle.length() < 31)){
154         throw new InvalidHandleException();
155     }
156     int acc = -1;
157     for (int i = 0; i < accounts.size(); i++){
158         String currentHandle = accounts.get(i).getHandle();
159         if (newHandle.equals(currentHandle)) {
160             throw new IllegalArgumentException();
161         }
162         if (oldHandle.equals(currentHandle)) {
163             acc = i;
164         }
165     }
166 }

```

```

162         if (acc > -1) {
163             accounts.get(acc).setHandle(newHandle);
164         }
165         else{
166             throw new HandleNotRecognisedException();
167         }
168     }
169
170
171     // Uses the findAccountFromHandle method to find the account, and changes the
description if one is found
172     @Override
173     public void updateAccountDescription(String handle, String description) throws
HandleNotRecognisedException {
174
175         findAccountFromHandle(handle).setDescription(description);
176     }
177
178     // Uses the findAccountFromHandle method to find the account, then returns a
formatted string to display the information about the account
179     @Override
180     public String showAccount(String handle) throws HandleNotRecognisedException {
181
182         Account i = findAccountFromHandle(handle);
183         return "ID: " + i.getID() + System.lineSeparator() +
184             "Handle: " + handle + System.lineSeparator() +
185             "Description: " + i.getDescription() + System.lineSeparator() +
186             "Post count: " + i.getNumberOf("Post") + System.lineSeparator() +
187             "Endorse count: " + i.getNumberOf("Endorsement");
188     }
189
190     // Validates the message length, attempts to find the account, calls the post
method on the account, then returns the posts ID
191     @Override
192     public int createPost(String handle, String message) throws
HandleNotRecognisedException, InvalidPostException {
193
194         if (!(0 < message.length() && message.length() < 100)){
195             throw new InvalidPostException();
196         }
197         Post post = findAccountFromHandle(handle).post(message);
198         return post.getID();
199     }
200
201     // Attempts to find the post and account, then validates that it is a post (not
comment/endorsement) then calls the endorse method on the post and returns the
endorsement ID
202     @Override
203     public int endorsePost(String handle, int id) throws HandleNotRecognisedException,
PostIDNotRecognisedException, NotActionablePostException {
204
205         FeedDenizen post = findPostFromID(id);
206         if (post.getClass().getSimpleName().equals("Post")){
207             Endorsement end = findAccountFromHandle(handle).endorse(id);
208             post.endorse();
209             return end.getID();
210         }
211         throw new NotActionablePostException();
212     }
213

```

```

214 // Validates the message length, attempts to find the account, then validates that
    it is not a endorsement, then calls the comment method on the account, then returns the
    posts ID
215 @Override
216 public int commentPost(String handle, int id, String message) throws
    HandleNotRecognisedException, PostIDNotRecognisedException, NotActionablePostException,
    InvalidPostException {
217
218     if (!(0 < message.length() && message.length() < 100)){
219         throw new InvalidPostException();
220     }
221     FeedDenizen post = findPostFromID(id);
222     if (!post.getClass().getSimpleName().equals("Endorsement")){
223         Comment comment = findAccountFromHandle(handle).comment(message, id);
224
225         return comment.getID();
226     }
227     throw new NotActionablePostException();
228 }
229
230 // Finds the post, then calls the delete method on it, and adds all the child
    comments to the orphanage
231 @Override
232 public void deletePost(int id) throws PostIDNotRecognisedException {
233     int[] ids = unpackID(id);
234     Account account = findAccountFromKnownID(ids[0]);
235     List<Integer> orphans = account.deletePost(ids[1]);
236     for (Integer childID : orphans){
237         orphanage.add(findPostFromKnownID(childID));
238     }
239 }
240
241 // Formats a string to contain all the information for a single post: ID, account,
    total endorsements, total comments, post text
242 @Override
243 public String showIndividualPost(int id) throws PostIDNotRecognisedException {
244     int[] ids = unpackID(id);
245     Account account = findAccountFromKnownID(ids[0]);
246     FeedDenizen post = findPostFromID(id);
247     return "ID: " + post.getID() + System.lineSeparator() +
248         "Account: " + account.getHandle() + System.lineSeparator() +
249         "No. endorsements: " + post.getTotalEndorsements() + "| No. comments:
    " + post.getTotalComments() + System.lineSeparator() +
250         post.getText();
251 }
252
253 // Finds a post from an ID and adds it to a new List. Then it iterates through the
    list and for each post adds the formatted string from the showIndividualPost method
254 // The for each post it checks if it has children, if so it adds the children to
    the list directly after their parent post and adds one to the indent level
255 // It also adds the int 0 to the list after the children it just added. The 0
    signifies when to go back down an indent level.
256 // When the list is exhausted, the formatted string is returned
257 @Override
258 public StringBuilder showPostChildrenDetails(int id) throws
    PostIDNotRecognisedException, NotActionablePostException {
259
260     List<Integer> posts = new ArrayList<>();
261     posts.add(id);
262
263     StringBuilder str = new StringBuilder();

```

```

264         String indent = "    ";
265
266         int indentLevel = 0;
267         int index = 0;
268         while (index < posts.size()){
269
270             if (posts.get(index) != 0){
271                 String[] comment =
showIndividualPost(posts.get(index)).split(System.lineSeparator());
272                 for (String line : comment){
273                     if ((indentLevel > 0) && (line.substring(0,3).equals("ID:"))){
274                         str.append(indent.repeat(indentLevel -1));
275                         str.append("| > ");
276                         str.append(line);
277                         str.append(System.lineSeparator());
278                     } else {
279                         str.append(indent.repeat(indentLevel));
280                         str.append(line);
281                         str.append(System.lineSeparator());
282                     }
283                 }
284
285
286                 FeedDenizen post = findPostFromID(posts.get(index));
287                 List<Integer> children = post.getChildren();
288                 if (children.size() > 0){
289                     str.append(indent.repeat(indentLevel));
290                     str.append("|");
291                     str.append(System.lineSeparator());
292
293                     children.add(0);
294                     posts.addAll(index+1, children);
295                     indentLevel += 1;
296                 }
297
298                 } else {
299                     indentLevel -=1;
300                 }
301                 index += 1;
302             }
303             return str;
304         }
305
306         // Returns the number of accounts
307         @Override
308         public int getNumberOfAccounts() {
309
310             return accounts.size();
311         }
312
313         // Iterates through the accounts, and sums the number of post type posts
314         @Override
315         public int getTotalOriginalPosts() {
316
317             int count= 0;
318             for (Account i : accounts){
319                 count += i.getNumberOf("Post");
320             }
321             return count;

```

```

322     }
323
324     // Iterates through the accounts, and sums the number of endorsements type posts
325     @Override
326     public int getTotalEndorsementPosts() {
327
328         int count= 0;
329         for (Account i : accounts){
330             count += i.getNumberOf("Endorsement");
331         }
332         return count;
333     }
334
335     // Iterates through the accounts, and sums the number of comment type posts
336     @Override
337     public int getTotalCommentPosts() {
338
339         int count= 0;
340         for (Account i : accounts){
341             count += i.getNumberOf("Comment");
342         }
343         return count;
344     }
345
346     // Iterates through the accounts, and finds the account with the most endorsed post
347     @Override
348     public int getMostEndorsedPost() {
349
350         int previous = 0;
351         int previousID = 0;
352         for (Account account : accounts){
353             int[] current = account.mostEndorsements();
354             if (current[0] - previous > 0){
355                 previous = current[0];
356                 previousID = current[1];
357             }
358         }
359         return previousID;
360     }
361
362     // Iterates through the accounts, and finds the account with the most endorsements
363     @Override
364     public int getMostEndorsedAccount() {
365
366         int previous = 0;
367         int previousID = 0;
368         for (Account account : accounts){
369             int current = account.totalEndorsements();
370             if (current > previous){
371                 previous = current;
372                 previousID = account.getID();
373             }
374         }
375         return previousID;
376     }
377
378     // Removes all references to accounts and therefore posts. Also removes all
379     // reference to orphan posts
380     @Override

```

```

380     public void erasePlatform() {
381
382         accounts.removeAll(accounts);
383         orphanage.removeAll(orphanage);
384     }
385
386     // Starts by deleting any existing files with the same filename in the same
387     // directory if they exist
388     // To the txt file, it writes:
389     // For each account it writes the account: ID, handle, description followed by an
390     // opening bracket
391     // For each post from that account's timeline it writes post: type, ID, text,
392     // total comments, total endorsements, parent ID followed by an opening bracket
393     // For each comment for each post it writes the ID followed by a line (|)
394     // Ends the post
395     // Therefore the formatting for one account look like:
396     // 34,Maurice Moss,IT support at Reynholm Industries,(P,340009,Did you see that
397     // ludicrous display last night?,1,3,0,(560023|)/C,340010,thing about Arsenal is they
398     // always try and walk it in,2,2,560023,(560024|470143|)/)
399     @Override
400     public void savePlatform(String filename) throws IOException {
401
402         try{
403             File f = new File(filename);
404             if (f.exists()){
405                 f.delete();
406             }
407             f.createNewFile();
408             FileWriter fWriter = new FileWriter(filename);
409             BufferedWriter bWriter = new BufferedWriter(fWriter);
410             for (Account account : accounts){
411                 StringBuffer line = new StringBuffer();
412                 String[] chars = {String.valueOf(account.getID()),
413                 account.getHandle(), account.getDescription()};
414                 for (String i : chars){
415                     line.append(i + ",");
416                 }
417                 line.append("(");
418                 for (FeedDenizen post : account.getTimeline()){
419                     char type = post.getClass().getSimpleName().charAt(0);
420                     line.append(type + "," + post.getID() + "," + post.getText() + ","
+ post.getTotalComments() + "," + post.getTotalEndorsements() + "," + post.getParent()
+ ",");
421                     line.append("(");
422                     for (Integer child : post.getChildren()){
423                         line.append(child + "|");
424                     }
425                     line.append(")/");
426                 }
427                 line.append(")");
428                 bWriter.write(line.toString());
429                 bWriter.flush();
430             }
431             bWriter.close();
432         } catch (IOException e){
433             throw new IOException();
434         }
435     }

```



```

433 // For each line in the file, it creates an account. For each post on the account
it creates the relevant post and adds it to the timeline.
434 @Override
435 public void loadPlatform(String filename) throws IOException,
ClassNotFoundException {
436
437     BufferedReader reader;
438     String line = null;
439     try{
440         reader = new BufferedReader(new FileReader(filename));
441
442         while ((line = reader.readLine()) != null) {
443             String[] accDesc = line.split(",", 4);
444             String strPosts = accDesc[3].substring(1, accDesc[3].length()-1);
445             String[] posts = strPosts.split("/");
446             int id = Integer.parseInt(accDesc[0]);
447             Account account = new Account(id, accDesc[1], accDesc[2]);
448             for (String p : posts){
449                 String[] splitPost = p.split(",");
450
451                 if (splitPost[0].equals("P")){
452                     Post newPost = account.post(splitPost[2]);
453                     int endorsements = Integer.parseInt(splitPost[4]);
454                     newPost.endorsements = endorsements;
455                     int noComments = Integer.parseInt(splitPost[3]);
456                     if (noComments > 0){
457                         String strComments = splitPost[6].substring(1,
splitPost[6].length() - 1);
458                         String[] comments = strComments.split("|");
459                         List<Integer> children = new ArrayList<>();
460                         for (String i : comments){
461                             int y = Integer.parseInt(i);
462                             children.add(y);
463                         }
464                         newPost.children = children;
465                     }
466                 }
467                 if (splitPost[0].equals("C")){
468                     int parent = Integer.parseInt(splitPost[5]);
469                     Comment newPost = account.comment(splitPost[2], parent);
470                     int endorsements = Integer.parseInt(splitPost[4]);
471                     newPost.endorsements = endorsements;
472                     int noComments = Integer.parseInt(splitPost[3]);
473                     if (noComments > 0){
474                         String strComments = splitPost[6].substring(1,
splitPost[6].length() - 1);
475                         String[] comments = strComments.split("|");
476                         List<Integer> children = new ArrayList<>();
477                         for (String i : comments){
478                             int y = Integer.parseInt(i);
479                             children.add(y);
480                         }
481                         newPost.children = children;
482                     }
483                 }
484                 if (splitPost[0].equals("E")){
485                     int parent = Integer.parseInt(splitPost[5]);
486                     account.endorse(parent);
487                 }
488             }
489         }
490     }
491 }

```

```
489         }
490
491         reader.close();
492
493     } catch (IOException e){
494         throw new IOException();
495     }
496
497 }
498
499 }
500 }
501
```

ecm1410_coursework\src\socialmedia\Account.java

```
1 package socialmedia;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 public class Account {
7     int ID;
8     String handle;
9     String description;
10    List<FeedDenizen> timeline;
11
12    public Account(int id, String y, String z){
13        this.ID = id;
14        this.handle = y;
15        this.description = z;
16        this.timeline = new ArrayList<>();
17    }
18
19    public int getID() {
20        return this.ID;
21    }
22
23    public String getHandle() {
24        return this.handle;
25    }
26
27    public void setHandle(String str) {
28        this.handle = str;
29    }
30
31    public String getDescription() {
32        return this.description;
33    }
34
35    public List<FeedDenizen> getTimeline(){
36        return this.timeline;
37    }
38
39    // Iterates through the timeline and finds the most endorsed post
40    public int[] mostEndorsements(){
41        int previous = 0;
42        int previousID = 0;
43        for (FeedDenizen i : this.timeline){
44            if (i.getTotalEndorsements() > previous){
45                previous = i.getTotalEndorsements();
46                previousID = i.getID();
47            }
48        }
49        return new int[] {previous, previousID};
50    }
51
52    // Sums the total amount of endorsements the account has received
53    public int totalEndorsements(){
54        int sum = 0;
55    }
```

```

56         for (FeedDenizen i : timeline){
57             sum += i.getTotalEndorsements();
58         }
59         return sum;
60     }
61
62     public FeedDenizen getPost(int id) throws PostIDNotRecognisedException{
63         for (FeedDenizen i : this.timeline){
64             if (id == i.getPostID()){
65                 return i;
66             }
67         }
68         throw new PostIDNotRecognisedException();
69     }
70
71     public FeedDenizen getKnownPost(int id){
72         for (FeedDenizen i : this.timeline){
73             if (id == i.getPostID()){
74                 return i;
75             }
76         }
77         return null;
78     }
79
80     public void setDescription(String str) {
81         this.description = str;
82     }
83
84     // Finds the total number of elements in the timeline with the same type as the
argument
85     public int getNumberOf(String arg){
86         int count = 0;
87         for (FeedDenizen i : this.timeline){
88             String type = i.getClass().getSimpleName();
89             if (type.equals(arg)){
90                 count += 1;
91             }
92         }
93         return count;
94     }
95
96     // Deletes all of the posts and collects their children, then clears all the user
data - handle, description and timeline and returns the children
97     public List<Integer> delete(){
98         List<Integer> orphans = new ArrayList<>();
99         for (FeedDenizen i : this.timeline){
100             orphans.addAll(1, i.getChildren());
101             i.delete();
102         }
103         this.handle = "[DELETED USER]";
104         this.description = "";
105         this.timeline = new ArrayList<>();
106         return orphans;
107     }
108
109     // Creates a post on the timeline
110     public Post post(String message){
111         Post post = new Post(this.timeline.size(), this.ID, message);
112         this.timeline.add(post);

```

```
113         return post;
114     }
115
116     // Creates an endorsement on the timeline
117     public Endorsement endorse(int id){
118         Endorsement endorsement = new Endorsement(this.timeline.size(), this.ID, id);
119         this.timeline.add(endorsement);
120         return endorsement;
121     }
122
123     // Creates a comment on the timeline
124     public Comment comment(String message, int postID){
125         Comment comment = new Comment(this.timeline.size(), this.ID, postID, message);
126         this.timeline.add(comment);
127         return comment;
128     }
129
130     // Deletes a post given specific ID
131     public List<Integer> deletePost(int id) throws PostIDNotRecognisedException{
132         FeedDenizen post = getPost(id);
133         List<Integer> orphans = post.getChildren();
134         post.delete();
135         return orphans;
136     }
137 }
```

ecm1410_coursework\src\socialmedia\FeedDenizen.java

```
1  package socialmedia;
2
3  import java.util.List;
4
5  abstract class FeedDenizen {
6      int ID;
7      int accountID;
8
9      public void init(int id, int accID){
10         this.ID = id;
11         this.accountID = accID;
12     }
13
14     public int getID(){
15         return this.accountID * 10000 + this.ID;
16     }
17
18     public int getPostID(){
19         return this.ID;
20     }
21
22     public int getAccount(){
23         return this.accountID;
24     }
25
26     abstract List<Integer> getChildren();
27
28     abstract void delete();
29
30     abstract void endorse();
31
32     abstract int getTotalComments();
33
34     public int getParent(){
35         return 0;
36     }
37
38     public int getTotalEndorsements(){
39         return 0;
40     }
41
42     public String getText(){
43         return null;
44     }
45 }
46
```

ecm1410_coursework\src\socialmedia\Post.java

```
1  package socialmedia;
2
3  import java.util.ArrayList;
4  import java.util.List;
5
6  public class Post extends FeedDenizen {
7      int endorsements;
8      String text;
9      List<Integer> children;
10
11     public Post(int id, int accID, String message){
12         init(id, accID);
13         this.endorsements = 0;
14         this.text = message;
15         this.children = new ArrayList<>();
16     }
17
18     public String getText(){
19         return this.text;
20     }
21
22     public void endorse(){
23         this.endorsements += 1;
24     }
25
26     public void addComment(int commentID){
27         this.children.add(commentID);
28     }
29
30     public void removeComment(int commentID){
31         boolean found = false;
32         for (int i = 0; (i < children.size()) & !found; i++){
33             if (children.get(i) == commentID){
34                 children.remove(i);
35                 found = true;
36             }
37         }
38     }
39
40     public int getTotalEndorsements(){
41         return this.endorsements;
42     }
43
44     public int getTotalComments(){
45         return this.children.size();
46     }
47
48     @Override
49     public List<Integer> getChildren(){
50         return this.children;
51     }
52
53     @Override
54     public void delete(){
55         this.accountID = 0;
```

```
56 |         this.text = "[POST DELETED]";
57 |     }
58 | }
59 |
```


ecm1410_coursework\src\socialmedia\Comment.java

```
1  package socialmedia;
2  import java.util.ArrayList;
3  import java.util.List;
4
5  public class Comment extends FeedDenizen{
6      int parentID;
7      int endorsements;
8      String text;
9      List<Integer> children;
10
11     public Comment(int id, int accID, int postID, String message){
12         init(id, accID);
13         this.parentID = postID;
14         this.text = message;
15         this.children = new ArrayList<>();
16         this.endorsements = 0;
17     }
18
19     public int getParent(){
20         return this.parentID;
21     }
22
23     public int getTotalComments(){
24         return this.children.size();
25     }
26
27     public void endorse(){
28         this.endorsements += 1;
29     }
30
31     public String getText(){
32         return this.text;
33     }
34
35     @Override
36     public List<Integer> getChildren(){
37         return this.children;
38     }
39
40     @Override
41     public void delete(){
42         this.accountID = 0;
43         this.text = "[POST DELETED]";
44     }
45 }
46
```

ecm1410_coursework\src\socialmedia\Endorsement.java

```
1 package socialmedia;
2 import java.util.ArrayList;
3 import java.util.List;
4
5
6 public class Endorsement extends FeedDenizen{
7     int postID;
8
9     public Endorsement(int id, int accID, int targetID){
10         init(id, accID);
11         this.postID = targetID;
12     }
13
14     public void endorse(){
15     }
16
17     public int getParent() {
18         return this.postID;
19     }
20
21     public int getTotalComments(){
22         return 0;
23     }
24
25
26     @Override
27     void delete() {
28         this.accountID = 0;
29     }
30
31     @Override
32     List<Integer> getChildren(){
33         return new ArrayList<>();
34     }
35 }
36 }
```