

Commit

 This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

Refatoração em bo

Browse files

LoginBO - Técnicas: Extract Method, Simplification of Conditions

PessoaBO - Técnicas: Move Method, Extract Method, Encapsulate Field

UfBO - Técnicas: Extract Method, Encapsulate Field

ValidaCamposPessoasBO - Técnicas: Extract Method

 moisesocosta committed on Jun 22

1 parent 90dff36 commit 0643563

Showing 4 changed files with 78 additions and 94 deletions.

Whitespace

Ignore whitespace

Split

Unified

42

src/br/sistema/crud/jdbc/bo/LoginBO.java

```
...      ...      @@ -1,32 +1,30 @@
1      1      package br.sistema.crud.jdbc.bo;
2      2
3      3      import br.sistema.crud.jdbc.dao.LoginDAO;
4      - import br.sistema.crud.jdbc.dao.PessoaDAO;
5      4      import br.sistema.crud.jdbc.dto.LoginDTO;
6      - import br.sistema.crud.jdbc.dto.PessoaDTO;
7      5      import br.sistema.crud.jdbc.exception.NegocioException;
8      6
9      7      public class LoginBO {
10     -
11     -      public boolean login(LoginDTO loginDTO) throws NegocioException{
12     -          boolean resultado = false;
13     -          try{
14     -              if(loginDTO.getNome() == null || "".equals(loginDTO)){
15     -                  throw new NegocioException("Logn Obrigatorio");
16     -              }else if(loginDTO.getSenha() == null ||
17     -                  "".equals(loginDTO.getSenha())){
```

```

17      -                                     throw new NegocioException("Login Obrigatorio");
18      8      -
19      -                                     }else{
20      -                                     LoginDAO loginDAO = new LoginDAO();
21      -                                     resultado= loginDAO.logar(loginDTO);
22      -
23      -                                     }
24      -
25      -                                     }catch(Exception e){
26      -                                     e.printStackTrace();
27      -                                     throw new NegocioException(e.getMessage(),e);
28      -                                     }
29      -                                     return resultado;
30      -                                     }
31      9      + private static final String MENSAGEM_LOGIN_OBRIGATORIO = "Login obrigatório";
32      10     +
33      11     + public boolean logar(LoginDTO loginDTO) throws NegocioException {
34      12     +         validarLogin(loginDTO);
35      13     +
36      14     +         try {
37      15     +             LoginDAO loginDAO = new LoginDAO();
38      16     +             return loginDAO.logar(loginDTO);
39      17     +         } catch (Exception e) {
40      18     +             e.printStackTrace();
41      19     +             throw new NegocioException(e.getMessage(), e);
42      20     +         }
43      21     +     }
44      22     +
45      23     + private void validarLogin(LoginDTO loginDTO) throws NegocioException {
46      24     +         if (loginDTO.getNome() == null || loginDTO.getNome().isEmpty()) {
47      25     +             throw new NegocioException(MENSAGEM_LOGIN_OBRIGATORIO);
48      26     +         } else if (loginDTO.getSenha() == null || loginDTO.getSenha().isEmpty()) {
49      27     +             throw new NegocioException(MENSAGEM_LOGIN_OBRIGATORIO);
50      28     +         }
51      29     +     }
52      30     + }

```

78 src/br/sistema/crud/jdbc/bo/PessoaBO.java

```

14      14     public class PessoaBO {
15      15
16      16         private DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
17      17         private PessoaDAO pessoaDAO;
18      18
19      19         public PessoaBO() {
20      20             this.pessoaDAO = new PessoaDAO();
21      21         }
22
23      23         public void cadastrar(PessoaDTO pessoaDTO) throws NegocioException {
24      24             try {
25      25                 PessoaDAO pessoaDAO = new PessoaDAO();
26      26                 pessoaDAO.inserir(pessoaDTO);

```

```

22      26          } catch(Exception exception) {
23      27              exception.printStackTrace();
26      30      }
27      31
28      32      public String[][] listagem(List<Integer> idsPessoas) throws
NegocioException {
29      -          int numCols = 10;
30      -          String[][] listaRetorno = null;
31      33      +          final int numCols = 10;
32      34      try {
33      -          PessoaDAO pessoaDAO = new PessoaDAO();
34      -
35      35          List<PessoaDTO> lista = pessoaDAO.listarTodos();
36      36      +          listaRetorno = new String[lista.size()][numCols];
37      37      +          String[][] listaRetorno = new String[lista.size()]
[numCols];
38      38
39      39          for (int i = 0; i < lista.size(); i++) {
40      40              PessoaDTO pessoa = lista.get(i);
41      41              listaRetorno[i][8] = "ALTER";
42      42              listaRetorno[i][9] = "DEL";
43      43          }
44      44
45      45      +          return listaRetorno;
46      46      +
47      47      } catch(Exception exception) {
48      48          throw new NegocioException(exception.getMessage());
49      49      }
50      50      return listaRetorno;
51      51      }
52      52
53      53      public boolean validaNome(String nome) throws ValidacaoException {
54      54          boolean ehValido = true;
55      55          if (nome == null || nome.equals("")) {
56      56      +          if (nome == null || nome.isEmpty()) {
57      57              ehValido = false;
58      58              throw new ValidacaoException("Campo nome é obrigatório!");
59      59      +          throw new ValidacaoException("Campo nome é obrigatório!");
60      60          } else if (nome.length() > 30) {
61      61              ehValido = false;
62      62              throw new ValidacaoException("Campo nome comporta no máximo
30 chars!");
63      63      +          throw new ValidacaoException("Campo nome comporta no máximo
30 chars!");
64      64          }
65      65          return ehValido;
66      66      }
67      67
68      68      public boolean validaCpf(String cpf) throws ValidacaoException {
69      69          boolean ehValido = true;
70      70          if (cpf == null || cpf.equals("")) {
71      71      +          if (cpf == null || cpf.isEmpty()) {
72      72              ehValido = false;

```

```

74      -          throw new ValidacaoException("Campo CPF é obrigatório!");
75      76      +          throw new ValidacaoException("Campo CPF é obrigatório!");
76      77      +      } else if (cpf.length() != 11) {
77      78      +          ehValido = false;
78      79      -          throw new ValidacaoException("Campo CPF deve ter 11
          dígitos!");
79      80      +          throw new ValidacaoException("Campo CPF deve ter 11
          dígitos!");
80      81      +      } else {
81      82      +          char[] digitos = cpf.toCharArray();
82      83      +          for (char digito : digitos) {
83      84      +              if (!Character.isDigit(digito)) {
84      85      +                  ehValido = false;
85      86      +                  throw new ValidacaoException("Campo CPF é
          somente numérico!");
86      87      +                  throw new ValidacaoException("Campo CPF é
          somente numérico!");
87      88      +              }
88      89      +          }
89      90      +      }
90      91      +
91      92      +      public boolean validaEndereco(EnderecoDTO enderecoDTO) throws
          ValidacaoException {
92      93      +          boolean ehValido = true;
93      94      -          if (enderecoDTO.getLogadouro() == null ||
          enderecoDTO.getLogadouro().equals("")) {
94      95      +          if (enderecoDTO.getLogadouro() == null ||
          enderecoDTO.getLogadouro().isEmpty()) {
95      96      +              ehValido = false;
96      97      -          throw new ValidacaoException("Campo Logradouro é
          obrigatório!");
97      98      -          } else if (enderecoDTO.getBairro() == null ||
          enderecoDTO.getBairro().equals("")) {
98      99      +          throw new ValidacaoException("Campo Logradouro é
          obrigatório!");
99      100      +          } else if (enderecoDTO.getBairro() == null ||
          enderecoDTO.getBairro().isEmpty()) {
100      101      +              ehValido = false;
101      102      -          throw new ValidacaoException("Bairro Obrigatorio");
102      103      -          } else if (enderecoDTO.getNumero() == null ||
          enderecoDTO.getNumero().equals(0)) {
103      104      +          throw new ValidacaoException("Bairro Obrigatório");
104      105      +          } else if (enderecoDTO.getNumero() == null ||
          enderecoDTO.getNumero() == 0) {
105      106      +              ehValido = false;
106      107      -          throw new ValidacaoException("Numero Obrigatorio");
107      108      -          } else if (enderecoDTO.getCep() == null ||
          enderecoDTO.getCep().equals(0)) {
108      109      +          throw new ValidacaoException("Número Obrigatório");
109      110      +          } else if (enderecoDTO.getCep() == null || enderecoDTO.getCep() ==
          0) {
110      111      +              ehValido = false;

```

```

103 | - | throw new ValidacaoException("CEP Obrigatorio");
    | 105 | + | throw new ValidacaoException("CEP Obrigatório");
104 | 106 | }
105 | - |
106 | - |
    | 107 | + |
107 | 108 | return ehValido;
108 | 109 | }
109 | 110 |
110 | 111 | public boolean validaDtNasc(String dtNasc) throws ValidacaoException {
111 | 112 |     boolean ehValido = true;
112 | - | if (dtNasc == null || dtNasc.equals("")) {
    | 113 | + | if (dtNasc == null || dtNasc.isEmpty()) {
113 | 114 |         ehValido = false;
114 | - |         throw new ValidacaoException("Campo Dt. Nasc. é
    |     | obrigatório!");
    | 115 | + |         throw new ValidacaoException("Campo Dt. Nasc. é
    |     | obrigatório!");
115 | 116 |     } else {
116 | - |         ehValido = false;
117 | 117 |         try {
118 | 118 |             dateFormat.parse(dtNasc);
119 | 119 |         } catch (ParseException e) {
120 | - |             throw new ValidacaoException("Formato inválido de
    |     | data!");
    | 120 | + |             ehValido = false;
    | 121 | + |             throw new ValidacaoException("Formato inválido de
    |     | data!");
121 | 122 |         }
122 | 123 |     }
123 | 124 |     return ehValido;
124 | 125 | }
125 | 126 |
126 | 127 | public String[][] listaConsulta(String nome, Long cpf, char sexo, String
    |     | orderBy) throws NegocioException {
127 | - |     int numCols = 6;
128 | - |     String[][] listaRetorno = null;
    | 128 | + |     final int numCols = 6;
129 | 129 |     try {
130 | - |         PessoaDAO pessoaDAO = new PessoaDAO();
131 | - |
132 | 130 |         List<PessoaDTO> lista = pessoaDAO.filtrarPessoa(nome, cpf,
    |     | String.valueOf(sexo), orderBy);
133 | - |         listaRetorno = new String[lista.size()][numCols];
    | 131 | + |         String[][] listaRetorno = new String[lista.size()]
    |     | [numCols];
134 | 132 |
135 | 133 |         for (int i = 0; i < lista.size(); i++) {
136 | 134 |             PessoaDTO pessoa = lista.get(i);
141 | 139 |             listaRetorno[i][4] = pessoa.getSexo() == 'M' ?
    |     | "Masculino" : "Feminino";

```

```

142      140      listaRetorno[i][5] =
dateFormat.format(pessoa.getDtNascimento());
143      141      }
      142      +
      143      +
      return listaRetorno;
144      144      } catch(Exception exception) {
145      145      throw new NegocioException(exception.getMessage());
146      146      }
147      -
      return listaRetorno;
148      147      }
149      148
150      149      public void removePessoa(Integer idPessoa, Integer idEndereco) throws
NegocioException {
151      150      try {
152      -
      PessoaDAO pessoaDAO = new PessoaDAO();
153      151      pessoaDAO.deletar(idPessoa);
154      152      pessoaDAO.deletarEndereco(idEndereco);
155      153      } catch(Exception exception) {
159      157
160      158      public void removeTudo() throws NegocioException {
161      159      try {
162      -
      PessoaDAO pessoaDAO = new PessoaDAO();
163      160      pessoaDAO.deletarTudo();
164      -
165      161      } catch(Exception e) {
166      162      throw new NegocioException(e.getMessage());
167      163      }
168      164      }
169      165
170      166      public PessoaDTO buscaPorId(Integer idPessoa) throws NegocioException{
171      -
      PessoaDTO pessoaDTO = null;
172      -
173      167      try {
174      -
      PessoaDAO pessoaDAO = new PessoaDAO();
175      -
      pessoaDTO = pessoaDAO.buscarPorId(idPessoa);
      168      +
      return pessoaDAO.buscarPorId(idPessoa);
176      169      } catch (Exception e) {
177      170      throw new NegocioException(e.getMessage(), e);
178      -
179      171      }
180      -
      return pessoaDTO;
181      172      }
182      -
183      173      }

```

▼ 28 ■■■ src/br/sistema/crud/jdbc/bo/UfBO.java

```

...      ...      @@ -1,28 +1,24 @@
1      1      package br.sistema.crud.jdbc.bo;
2      2
3      - import java.util.ArrayList;

```

```

4      3      import java.util.List;
5      4
6      5      import br.sistema.crud.jdbc.dao.UfDAO;
7      - import br.sistema.crud.jdbc.dto.UfDTO;
8      6      import br.sistema.crud.jdbc.exception.NegocioException;
9      7      + import br.sistema.crud.jdbc.model.Uf;
10     8
11     9      public class UfBO {
12     10
13     -      public List<UfDTO> listaUfs() throws NegocioException{
14     -          List<UfDTO> lista = null;
15     -
16     -      try{
17     -          UfDAO ufDAO = new UfDAO();
18     -          lista = ufDAO.listaEstado();
19     -
20     -      }catch (Exception e){
21     -          throw new NegocioException(e.getMessage(),e);
22     +      private UfDAO ufDAO;
23     +
24     +      public UfBO() {
25     +          this.ufDAO = new UfDAO();
26     +      }
27     +
28     +      public List<Uf> listarUfs() throws NegocioException {
29     +          try {
30     +              return ufDAO.listarUfs();
31     +          } catch (Exception e) {
32     +              throw new NegocioException(e.getMessage(), e);
33     +          }
34     +          return lista;
35     -
36     -
37     -
38     -
39     -
40     -      }
41     -
42     -
43     -
44     -
45     -
46     -
47     -
48     -
49     -
50     -
51     -
52     -
53     -
54     -
55     -
56     -
57     -
58     -
59     -
60     -
61     -
62     -
63     -
64     -
65     -
66     -
67     -
68     -
69     -
70     -
71     -
72     -
73     -
74     -
75     -
76     -
77     -
78     -
79     -
80     -
81     -
82     -
83     -
84     -
85     -
86     -
87     -
88     -
89     -
90     -
91     -
92     -
93     -
94     -
95     -
96     -
97     -
98     -
99     -
100    -
101    -
102    -
103    -
104    -
105    -
106    -
107    -
108    -
109    -
110    -
111    -
112    -
113    -
114    -
115    -
116    -
117    -
118    -
119    -
120    -
121    -
122    -
123    -
124    -
125    -
126    -
127    -
128    -
129    -
130    -
131    -
132    -
133    -
134    -
135    -
136    -
137    -
138    -
139    -
140    -
141    -
142    -
143    -
144    -
145    -
146    -
147    -
148    -
149    -
150    -
151    -
152    -
153    -
154    -
155    -
156    -
157    -
158    -
159    -
160    -
161    -
162    -
163    -
164    -
165    -
166    -
167    -
168    -
169    -
170    -
171    -
172    -
173    -
174    -
175    -
176    -
177    -
178    -
179    -
180    -
181    -
182    -
183    -
184    -
185    -
186    -
187    -
188    -
189    -
190    -
191    -
192    -
193    -
194    -
195    -
196    -
197    -
198    -
199    -
200    -
201    -
202    -
203    -
204    -
205    -
206    -
207    -
208    -
209    -
210    -
211    -
212    -
213    -
214    -
215    -
216    -
217    -
218    -
219    -
220    -
221    -
222    -
223    -
224    -
225    -
226    -
227    -
228    -
229    -
230    -
231    -
232    -
233    -
234    -
235    -
236    -
237    -
238    -
239    -
240    -
241    -
242    -
243    -
244    -
245    -
246    -
247    -
248    -
249    -
250    -
251    -
252    -
253    -
254    -
255    -
256    -
257    -
258    -
259    -
260    -
261    -
262    -
263    -
264    -
265    -
266    -
267    -
268    -
269    -
270    -
271    -
272    -
273    -
274    -
275    -
276    -
277    -
278    -
279    -
280    -
281    -
282    -
283    -
284    -
285    -
286    -
287    -
288    -
289    -
290    -
291    -
292    -
293    -
294    -
295    -
296    -
297    -
298    -
299    -
300    -
301    -
302    -
303    -
304    -
305    -
306    -
307    -
308    -
309    -
310    -
311    -
312    -
313    -
314    -
315    -
316    -
317    -
318    -
319    -
320    -
321    -
322    -
323    -
324    -
325    -
326    -
327    -
328    -
329    -
330    -
331    -
332    -
333    -
334    -
335    -
336    -
337    -
338    -
339    -
340    -
341    -
342    -
343    -
344    -
345    -
346    -
347    -
348    -
349    -
350    -
351    -
352    -
353    -
354    -
355    -
356    -
357    -
358    -
359    -
360    -
361    -
362    -
363    -
364    -
365    -
366    -
367    -
368    -
369    -
370    -
371    -
372    -
373    -
374    -
375    -
376    -
377    -
378    -
379    -
380    -
381    -
382    -
383    -
384    -
385    -
386    -
387    -
388    -
389    -
390    -
391    -
392    -
393    -
394    -
395    -
396    -
397    -
398    -
399    -
400    -
401    -
402    -
403    -
404    -
405    -
406    -
407    -
408    -
409    -
410    -
411    -
412    -
413    -
414    -
415    -
416    -
417    -
418    -
419    -
420    -
421    -
422    -
423    -
424    -
425    -
426    -
427    -
428    -
429    -
430    -
431    -
432    -
433    -
434    -
435    -
436    -
437    -
438    -
439    -
440    -
441    -
442    -
443    -
444    -
445    -
446    -
447    -
448    -
449    -
450    -
451    -
452    -
453    -
454    -
455    -
456    -
457    -
458    -
459    -
460    -
461    -
462    -
463    -
464    -
465    -
466    -
467    -
468    -
469    -
470    -
471    -
472    -
473    -
474    -
475    -
476    -
477    -
478    -
479    -
480    -
481    -
482    -
483    -
484    -
485    -
486    -
487    -
488    -
489    -
490    -
491    -
492    -
493    -
494    -
495    -
496    -
497    -
498    -
499    -
500    -
501    -
502    -
503    -
504    -
505    -
506    -
507    -
508    -
509    -
510    -
511    -
512    -
513    -
514    -
515    -
516    -
517    -
518    -
519    -
520    -
521    -
522    -
523    -
524    -
525    -
526    -
527    -
528    -
529    -
530    -
531    -
532    -
533    -
534    -
535    -
536    -
537    -
538    -
539    -
540    -
541    -
542    -
543    -
544    -
545    -
546    -
547    -
548    -
549    -
550    -
551    -
552    -
553    -
554    -
555    -
556    -
557    -
558    -
559    -
560    -
561    -
562    -
563    -
564    -
565    -
566    -
567    -
568    -
569    -
570    -
571    -
572    -
573    -
574    -
575    -
576    -
577    -
578    -
579    -
580    -
581    -
582    -
583    -
584    -
585    -
586    -
587    -
588    -
589    -
590    -
591    -
592    -
593    -
594    -
595    -
596    -
597    -
598    -
599    -
600    -
601    -
602    -
603    -
604    -
605    -
606    -
607    -
608    -
609    -
610    -
611    -
612    -
613    -
614    -
615    -
616    -
617    -
618    -
619    -
620    -
621    -
622    -
623    -
624    -
625    -
626    -
627    -
628    -
629    -
630    -
631    -
632    -
633    -
634    -
635    -
636    -
637    -
638    -
639    -
640    -
641    -
642    -
643    -
644    -
645    -
646    -
647    -
648    -
649    -
650    -
651    -
652    -
653    -
654    -
655    -
656    -
657    -
658    -
659    -
660    -
661    -
662    -
663    -
664    -
665    -
666    -
667    -
668    -
669    -
670    -
671    -
672    -
673    -
674    -
675    -
676    -
677    -
678    -
679    -
680    -
681    -
682    -
683    -
684    -
685    -
686    -
687    -
688    -
689    -
690    -
691    -
692    -
693    -
694    -
695    -
696    -
697    -
698    -
699    -
700    -
701    -
702    -
703    -
704    -
705    -
706    -
707    -
708    -
709    -
710    -
711    -
712    -
713    -
714    -
715    -
716    -
717    -
718    -
719    -
720    -
721    -
722    -
723    -
724    -
725    -
726    -
727    -
728    -
729    -
730    -
731    -
732    -
733    -
734    -
735    -
736    -
737    -
738    -
739    -
740    -
741    -
742    -
743    -
744    -
745    -
746    -
747    -
748    -
749    -
750    -
751    -
752    -
753    -
754    -
755    -
756    -
757    -
758    -
759    -
760    -
761    -
762    -
763    -
764    -
765    -
766    -
767    -
768    -
769    -
770    -
771    -
772    -
773    -
774    -
775    -
776    -
777    -
778    -
779    -
780    -
781    -
782    -
783    -
784    -
785    -
786    -
787    -
788    -
789    -
790    -
791    -
792    -
793    -
794    -
795    -
796    -
797    -
798    -
799    -
800    -
801    -
802    -
803    -
804    -
805    -
806    -
807    -
808    -
809    -
810    -
811    -
812    -
813    -
814    -
815    -
816    -
817    -
818    -
819    -
820    -
821    -
822    -
823    -
824    -
825    -
826    -
827    -
828    -
829    -
830    -
831    -
832    -
833    -
834    -
835    -
836    -
837    -
838    -
839    -
840    -
841    -
842    -
843    -
844    -
845    -
846    -
847    -
848    -
849    -
850    -
851    -
852    -
853    -
854    -
855    -
856    -
857    -
858    -
859    -
860    -
861    -
862    -
863    -
864    -
865    -
866    -
867    -
868    -
869    -
870    -
871    -
872    -
873    -
874    -
875    -
876    -
877    -
878    -
879    -
880    -
881    -
882    -
883    -
884    -
885    -
886    -
887    -
888    -
889    -
890    -
891    -
892    -
893    -
894    -
895    -
896    -
897    -
898    -
899    -
900    -
901    -
902    -
903    -
904    -
905    -
906    -
907    -
908    -
909    -
910    -
911    -
912    -
913    -
914    -
915    -
916    -
917    -
918    -
919    -
920    -
921    -
922    -
923    -
924    -
925    -
926    -
927    -
928    -
929    -
930    -
931    -
932    -
933    -
934    -
935    -
936    -
937    -
938    -
939    -
940    -
941    -
942    -
943    -
944    -
945    -
946    -
947    -
948    -
949    -
950    -
951    -
952    -
953    -
954    -
955    -
956    -
957    -
958    -
959    -
960    -
961    -
962    -
963    -
964    -
965    -
966    -
967    -
968    -
969    -
970    -
971    -
972    -
973    -
974    -
975    -
976    -
977    -
978    -
979    -
980    -
981    -
982    -
983    -
984    -
985    -
986    -
987    -
988    -
989    -
990    -
991    -
992    -
993    -
994    -
995    -
996    -
997    -
998    -
999    -
1000   -

```

✓ 24  src/br/sistema/crud/jdbc/bo/ValidaCamposPessoasBO.java 

```

...      ...      @@ -1,23 +1,23 @@
1      1      package br.sistema.crud.jdbc.bo;
2      2
3      -
4      3      import javax.swing.text.AttributeSet;
5      4      import javax.swing.text.BadLocationException;
6      5      import javax.swing.text.PlainDocument;
7      6
8      -
9      7      public class ValidaCamposPessoasBO extends PlainDocument {
10     -

```

```
11      8
12      -         private static final long serialVersionUID = 5649489410014769260L;
13      -
14      -         public void insertString(int offSet, String str, AttributeSet attr)throws
BadLocationException{
15      -             super.insertString(offSet, str.replaceAll("[^a-z|^A-Z |^ ]", ""),
attr);
16      -         }
17      -
18      -         public void nome(int offSet, String str, javax.swing.text.AttributeSet
attr)throws BadLocationException{
19      -             super.insertString(offSet, str.replaceAll("[^a-z|^A-Z]", ""),
attr);
20      9 +         private static final long serialVersionUID = 5649489410014769260L;
21      10 +
22      11 +         @Override
23      12 +         public void insertString(int offset, String str, AttributeSet attr) throws
BadLocationException {
24      13 +             super.insertString(offset, removeNonAlphabeticCharacters(str), attr);
25      14 +         }
26      15
27      16 +         public void nome(int offset, String str, AttributeSet attr) throws
BadLocationException {
28      17 +             super.insertString(offset, removeNonAlphabeticCharacters(str), attr);
29      18 +         }
30      19
31      20 +         private String removeNonAlphabeticCharacters(String str) {
32      21 +             return str.replaceAll("[^a-zA-Z ]", "");
33      22 +         }
34      23 }
```

0 comments on commit 0643563

Please [sign in](#) to comment.