

EXPLICAÇÕES

Autoavaliação: Gostei de praticar fazendo esse código, consegui explorar e demonstrar bastante conceitos que aprendi em POO, técnicas de programas, projeto de algoritmo etc. Como pontos fortes no meu desenvolvimento eu destacaria principalmente isso, também que de forma rápida eu consegui idealizar uma ou mais soluções para cada tipo de funcionalidade exigida no dicionário e consegui fazer com que desse certo e de maneira eficiente no código, apesar de extenso (que posso destacar como um ponto negativo ou positivo) o código está bem detalhado; consegui usar bastante bem as Collections e bibliotecas do Java, entendendo a parte em que cada uma deveria entrar e também como pensar e usá-las de maneira eficiente. Como pontos que tive certa dificuldade está ligado a parte de pensar nas exceções que poderiam ser geradas por erro do usuário e em criar uma persistência para cada parte em questão. Também na hora da testagem, a meu ver ficaram testes um pouco óbvios, porém que de fato testa as funcionalidades dos métodos sem necessariamente ter casos de “pleonismo” nos testes, mesmo em número reduzido.

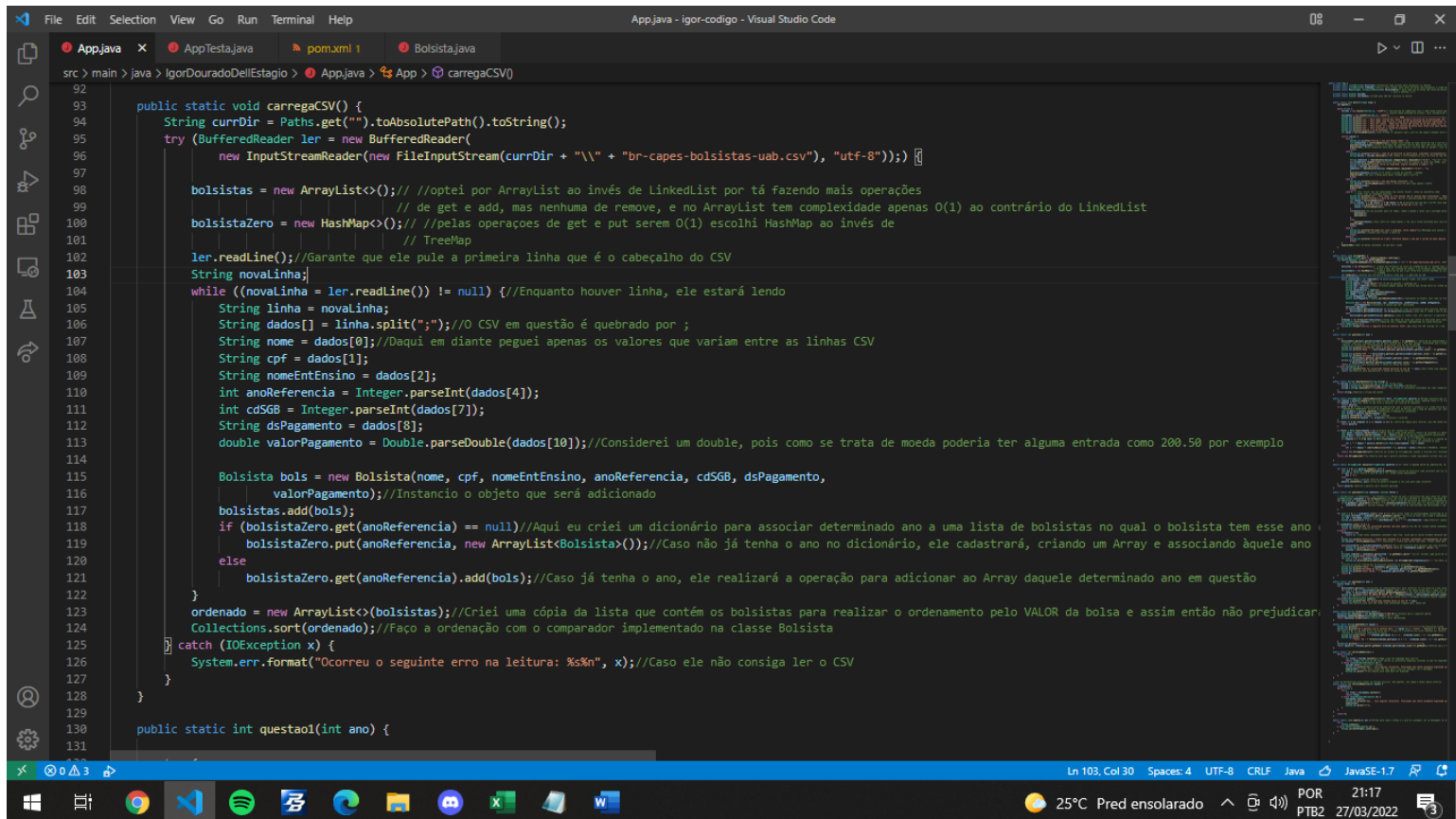
Em primeiro lugar criei meu projeto com o Maven, para criar de forma correta todas as dependências e a área de testes, logo após fiz o escopo do código e minhas ideias em papel e aí então comecei a colocá-las em código, como mostrarei no decorrer desse documento.

Observações:

*O código está em grande parte comentado, então optei por mostrar aqui mais meu pensamento para cada questão e alguns comentários, o código direto da questão, dos métodos que interferem diretamente neles, de um exemplo de execução e dos testes correspondentes a questão. Então no código há comentários mais específicos sobre cada parte do código e por qual motivo usei cada biblioteca ou método em cada parte do código, há métodos gerais que não interferem diretamente na questão e alguns outros testes desses métodos.

*Creio que consegui tratar qualquer exceção que pode vir a acontecer no programa por parte do usuário (com Try_Catch e While) e informei a mensagem adequada de como prosseguir e como não cometer o erro novamente.

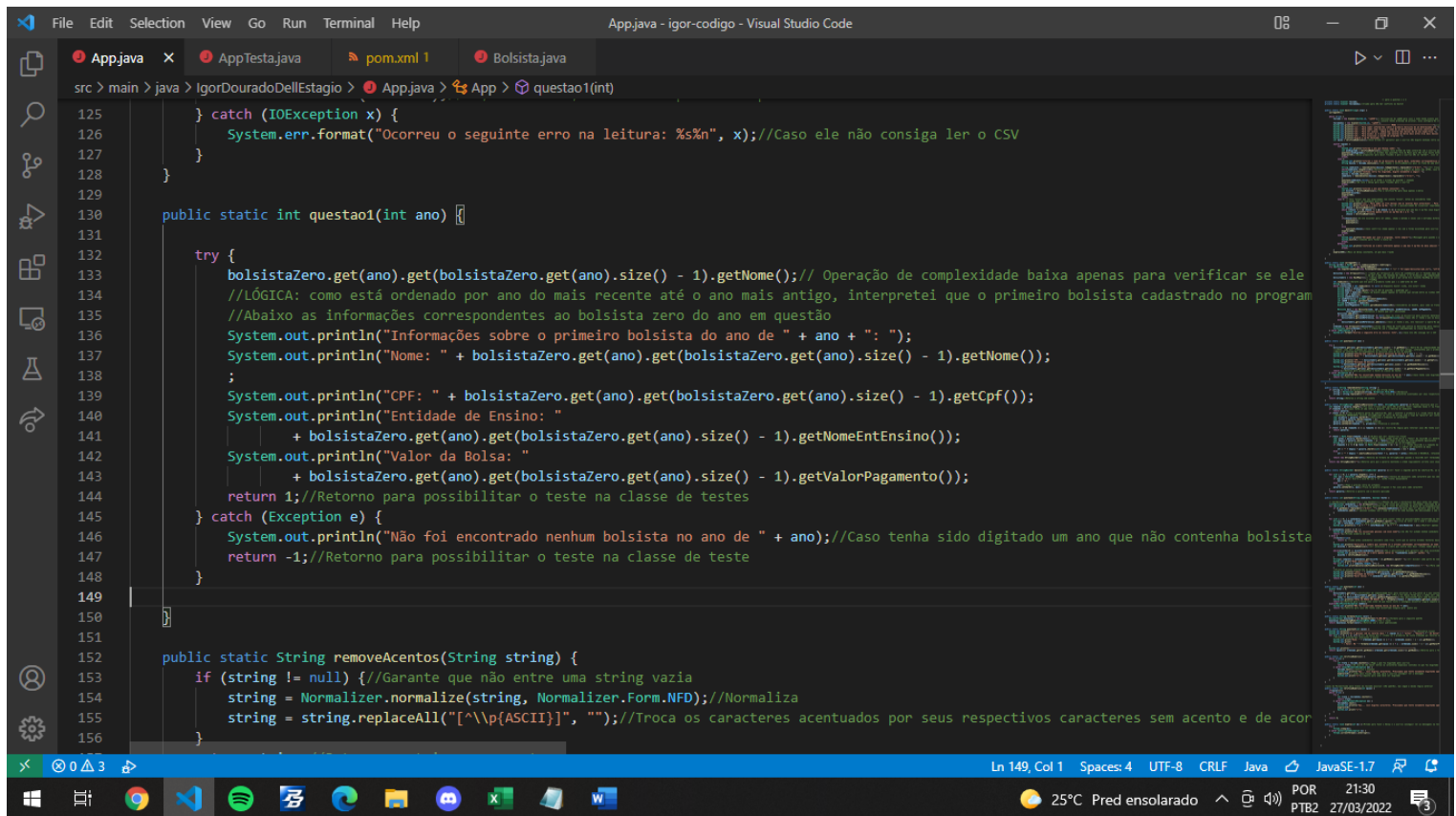
Leitura do CSV:



```
92
93 public static void carregaCSV() {
94     String currDir = Paths.get("").toAbsolutePath().toString();
95     try (BufferedReader ler = new BufferedReader(
96         new InputStreamReader(new FileInputStream(currDir + "\\br-capex-bolsistas-uab.csv"), "utf-8"));) {
97
98         bolsistas = new ArrayList<>(); //optei por ArrayList ao invés de LinkedList por tá fazendo mais operações
99         // de get e add, mas nenhuma de remove, e no ArrayList tem complexidade apenas O(1) ao contrário do LinkedList
100         bolsistaZero = new HashMap<>(); // pelas operações de get e put serem O(1) escolhi HashMap ao invés de
101         // TreeMap
102         ler.readLine(); //Garante que ele pule a primeira linha que é o cabeçalho do CSV
103         String novaLinha;
104         while ((novaLinha = ler.readLine()) != null) { //Enquanto houver linha, ele estará lendo
105             String linha = novaLinha;
106             String dados[] = linha.split(";"); //O CSV em questão é quebrado por ;
107             String nome = dados[0]; //Daqui em diante peguei apenas os valores que variam entre as linhas CSV
108             String cpf = dados[1];
109             String nomeEntEnsin = dados[2];
110             int anoReferencia = Integer.parseInt(dados[4]);
111             int cdSGB = Integer.parseInt(dados[7]);
112             String dsPagamento = dados[8];
113             double valorPagamento = Double.parseDouble(dados[10]); //Considere um double, pois como se trata de moeda poderia ter alguma entrada como 200.50 por exemplo
114
115             Bolsista bols = new Bolsista(nome, cpf, nomeEntEnsin, anoReferencia, cdSGB, dsPagamento,
116                 valorPagamento); //Instancio o objeto que será adicionado
117             bolsistas.add(bols);
118             if (bolsistaZero.get(anoReferencia) == null) //Aqui eu criei um dicionário para associar determinado ano a uma lista de bolsistas no qual o bolsista tem esse ano
119                 bolsistaZero.put(anoReferencia, new ArrayList<Bolsista>()); //Caso não já tenha o ano no dicionário, ele cadastrará, criando um Array e associando àquele ano
120             else
121                 bolsistaZero.get(anoReferencia).add(bols); //Caso já tenha o ano, ele realizará a operação para adicionar ao Array daquele determinado ano em questão
122         }
123         ordenado = new ArrayList<>(bolsistas); //Criei uma cópia da lista que contém os bolsistas para realizar o ordenamento pelo VALOR da bolsa e assim então não prejudicar
124         Collections.sort(ordenado); //Faço a ordenação com o comparador implementado na classe Bolsista
125     } catch (IOException x) {
126         System.err.format("Ocorreu o seguinte erro na leitura: %s\n", x); //Caso ele não consiga ler o CSV
127     }
128 }
129
130 public static int questao1(int ano) {
131 }
```

Utilizei a biblioteca de java.io que normalmente uso para ler arquivo e consegui ler o arquivo CSV sem precisar usar nenhuma biblioteca mais complexa, durante o código eu expliquei em comentários o motivo de eu ter escolhido cada biblioteca e Collections específicas em cada parte, por exemplo, na List de “bolsistas” eu escolhi utilizar um ArrayList ao invés de LinkedList pois as operações de Get e Add são de complexidades $O(1)$, ao contrário da LinkedList, ou seja, são mais eficientes. O mesmo ocorreu no map de “bolsistaZero”, em que escolhi HashMap ao invés de TreeMap por razões semelhantes. Por sinal, como comentei no código, eu criei esse dicionário para associar um determinado ano a uma lista de bolsistas no qual o bolsista tem esse ano como ano de referência e uma lista de bolsistas referentes associados a cada ano. Foi minha estratégia pensada para a questão 1 e 3 que usam do atributo de “anos” e fazer uso de poucas operações (e de baixa complexidade de tempo) para deixar o código mais simples, eficiente e rápido nas operações.

Questão 1- Explicação:



```
src > main > java > IgorDouradoDellEstagio > App.java > App > questao1(int)
125     } catch (IOException x) {
126         System.err.format("Ocorreu o seguinte erro na leitura: %s\n", x); //Caso ele não consiga ler o CSV
127     }
128 }
129
130 public static int questao1(int ano) {
131
132     try {
133         bolsistaZero.get(ano).get(bolsistaZero.get(ano).size() - 1).getNome(); // Operação de complexidade baixa apenas para verificar se ele
134         //LÓGICA: como está ordenado por ano do mais recente até o ano mais antigo, interpretei que o primeiro bolsista cadastrado no program
135         //Abaixo as informações correspondentes ao bolsista zero do ano em questão
136         System.out.println("Informações sobre o primeiro bolsista do ano de " + ano + ": ");
137         System.out.println("Nome: " + bolsistaZero.get(ano).get(bolsistaZero.get(ano).size() - 1).getNome());
138         ;
139         System.out.println("CPF: " + bolsistaZero.get(ano).get(bolsistaZero.get(ano).size() - 1).getCpf());
140         System.out.println("Entidade de Ensino: "
141             + bolsistaZero.get(ano).get(bolsistaZero.get(ano).size() - 1).getNomeEntEnsino());
142         System.out.println("Valor da Bolsa: "
143             + bolsistaZero.get(ano).get(bolsistaZero.get(ano).size() - 1).getValorPagamento());
144         return 1; //Retorno para possibilitar o teste na classe de testes
145     } catch (Exception e) {
146         System.out.println("Não foi encontrado nenhum bolsista no ano de " + ano); //Caso tenha sido digitado um ano que não contenha bolsista
147         return -1; //Retorno para possibilitar o teste na classe de teste
148     }
149 }
150
151
152 public static String removeAcentos(String string) {
153     if (string != null) { //Garante que não entre uma string vazia
154         string = Normalizer.normalize(string, Normalizer.Form.NFD); //Normaliza
155         string = string.replaceAll("[^\\p{ASCII}]", ""); //Troca os caracteres acentuados por seus respectivos caracteres sem acento e de acor
156     }
157 }
```

Minha estratégia na questão 1 foi me utilizar do Map que tem como “key” os anos e como “value” um ArrayList com os bolsistas que tem como ano aquele ano digitado pelo usuário e então pegar pela “key” o ano digitado pelo o usuário e verificar se ele consta no dicionário, se constar, peguei o último valor da Lista referente aquele ano por conta da seguinte lógica: como está ordenado por ano do mais recente até o ano mais antigo, interpretei que o primeiro bolsista cadastrado no programa foi o último a aparecer no CSV, logo ele seria o bolsista zero do GERAL e segui essa lógica para cada ano, pegando o último bolsista de cada ano no CSV original e considerando como o bolsista Zero.

Exemplo de execução e resultado:

```
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\igor\d\igor-codigo> cmd /C ""C:\Program Files\AdoptOpenJDK\jdk-11.0.8.10-hotspot\bin\java.exe" @C:\Users\igor\d\AppData\Local\Temp\cp_x03f4k114s5jwbryi4e25bb6.argfile IgorDouradoDellEstagio.App "

----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à média dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
1
Informe o ano que deseja saber:
2015
Informações sobre o primeiro bolsista do ano de 2015:
Nome: ELIANA DA COSTA PEREIRA DE MENEZES
CPF: ***.067.380-**
Entidade de Ensino: UNIVERSIDADE FEDERAL DE SANTA MARIA
Valor da Bolsa: 1300.0

----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à média dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
|
```

Testes unitários para a questão 1:

```
src > test > java > IgorDouradoDellEstagio > AppTesta.java > AppTesta

package IgorDouradoDellEstagio;

import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.api.Test;

public class AppTesta {

    @Test
    public void testaQuestao1True() {
        App.carregaCSV(); // Aqui também estou testando se a leitura do CSV correta e é necessário esse
        // método aqui para o funcionamento dos testes que envolvem a leitura e as listas
        assertEquals(1, App.questao1(2013)); // Verifica se acha e mantém esse padrão de devolver 1 caso ach
    }

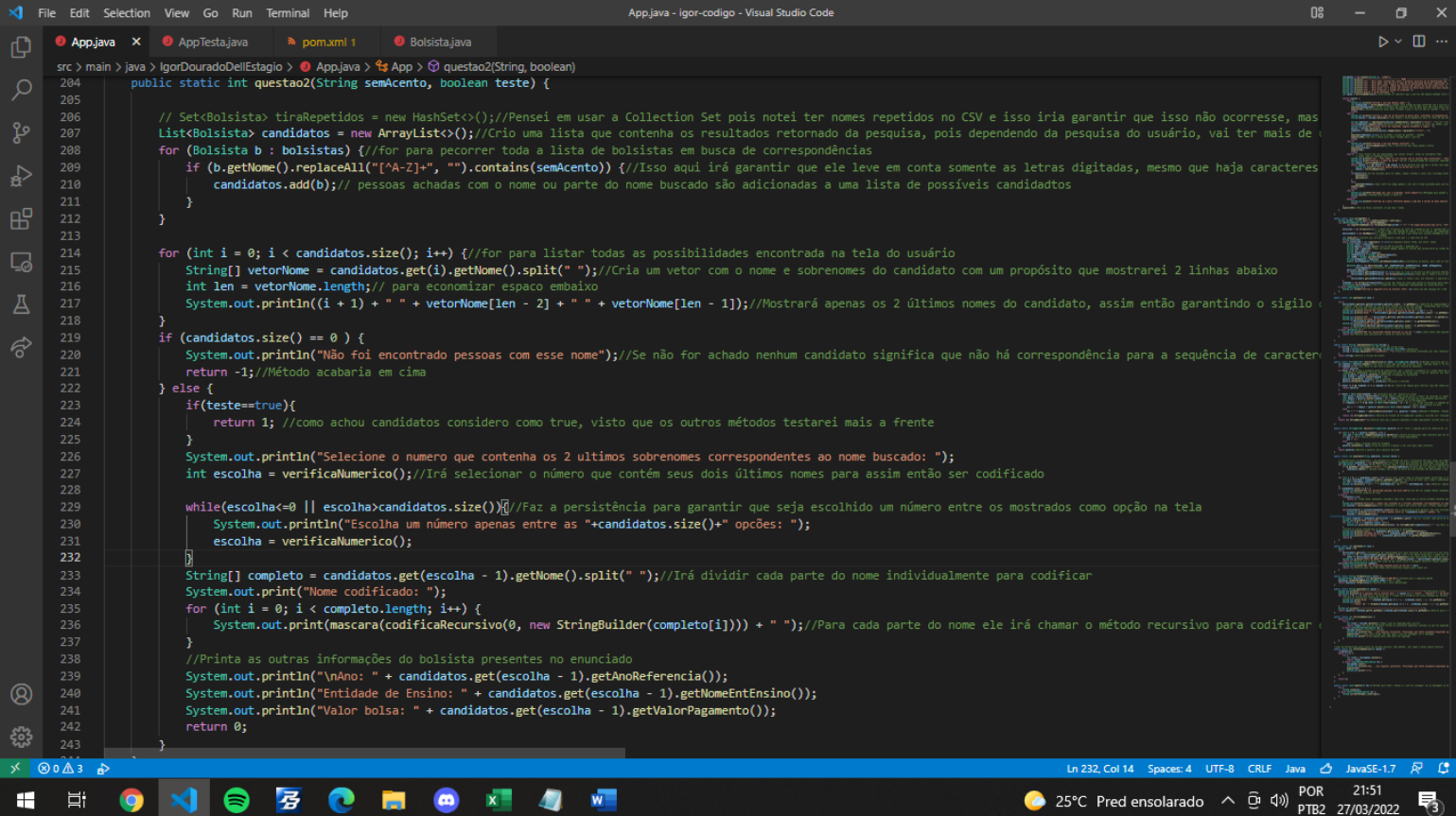
    @Test
    public void testaQuestao1False() {
        App.carregaCSV();
        assertEquals(-1, App.questao1(2009)); // Verifica se não acha e mantém esse padrão de devolver -1 ca
    }

    @Test
    public void testaQuestao2True() {
        App.carregaCSV();
        assertEquals(1, App.questao2("LAISE", true)); // Nomes não completo mas ele consegue achar correspon
    }

    @Test
    public void testaQuestao2False() {
        App.carregaCSV();
        assertEquals(-1, App.questao2("LAISEE", true)); // Não há esse nome na lista (com um 'E' a mais no LA
    }
}
```

Questão 2- Explicação:

O usuário pesquisa por um nome e como pode ter mais correspondências para uma mesma pesquisa de nome, criei uma lista de possíveis candidatos e exibi na tela contendo apenas os 2 últimos sobrenomes de cada bolsista, visto que sigilo dos bolsistas é uma questão importante na questão. Então, se houver, o programa listará as opções e o usuário deve escolher o que contenha os 2 últimos sobrenomes do nome que procura, sem saber o nome completo de outros, e só então ele entrará na parte da codificação.



```
src > main > java > IgorDouradoDellEstagio > App.java > App > questao2(String semAcento, boolean teste) {
284 public static int questao2(String semAcento, boolean teste) {
285
286 // Set<Bolsista> tiraRepetidos = new HashSet<>(); //Pensei em usar a Collection Set pois notei ter nomes repetidos no CSV e isso iria garantir que isso não ocorresse, mas
287 List<Bolsista> candidatos = new ArrayList<>(); //Crio uma lista que contenha os resultados retornado da pesquisa, pois dependendo da pesquisa do usuário, vai ter mais de
288 for (Bolsista b : bolsistas) { //for para percorrer toda a lista de bolsistas em busca de correspondências
289     if (b.getNome().replaceAll("[^A-Z]+", "").contains(semAcento)) { //Isso aqui irá garantir que ele leve em conta somente as letras digitadas, mesmo que haja caracteres
290         candidatos.add(b); // pessoas achadas com o nome ou parte do nome buscado são adicionadas a uma lista de possíveis candidatos
291     }
292 }
293
294 for (int i = 0; i < candidatos.size(); i++) { //for para listar todas as possibilidades encontrada na tela do usuário
295     String[] vetorNome = candidatos.get(i).getNome().split(" "); //Cria um vetor com o nome e sobrenomes do candidato com um propósito que mostrarei 2 linhas abaixo
296     int len = vetorNome.length; // para economizar espaço embaixo
297     System.out.println((i + 1) + " " + vetorNome[len - 2] + " " + vetorNome[len - 1]); //Mostrará apenas os 2 últimos nomes do candidato, assim então garantindo o sigilo
298 }
299
300 if (candidatos.size() == 0) {
301     System.out.println("Não foi encontrado pessoas com esse nome"); //Se não for achado nenhum candidato significa que não há correspondência para a sequência de caracteres
302     return -1; //Método acabaria em cima
303 } else {
304     if (teste == true) {
305         return 1; //como achou candidatos considero como true, visto que os outros métodos testarei mais a frente
306     }
307     System.out.println("Selecione o numero que contenha os 2 ultimos sobrenomes correspondentes ao nome buscado: ");
308     int escolha = verificaNumerico(); //Irá selecionar o número que contém seus dois últimos nomes para assim então ser codificado
309
310 while (escolha < 0 || escolha > candidatos.size()) { //Faz a persistência para garantir que seja escolhido um número entre os mostrados como opção na tela
311     System.out.println("Escolha um número apenas entre as "+candidatos.size()+" opções: ");
312     escolha = verificaNumerico();
313 }
314
315 String[] completo = candidatos.get(escolha - 1).getNome().split(" "); //Irá dividir cada parte do nome individualmente para codificar
316 System.out.print("Nome codificado: ");
317 for (int i = 0; i < completo.length; i++) {
318     System.out.print(mascara(codificaRecursivo(0, new StringBuilder(completo[i]))) + " "); //Para cada parte do nome ele irá chamar o método recursivo para codificar
319 }
320
321 //Printa as outras informações do bolsista presentes no enunciado
322 System.out.println("\nAno: " + candidatos.get(escolha - 1).getAnoReferencia());
323 System.out.println("Entidade de Ensino: " + candidatos.get(escolha - 1).getNomeEntEnsino());
324 System.out.println("Valor bolsa: " + candidatos.get(escolha - 1).getValorPagamento());
325 return 0;
326 }
327 }
```

Pra primeira parte da codificação eu pensei de duas formas: uma utilizando o método reverse, visto que a palavra será revertida em um segundo ponto ou utilizar recursão, visto que ocorre seguidas troca, mas achei interessante explorar a recursão nesse ponto e pensei da seguinte forma:

PAZ → ZAP

PALAVRAS
* Em Palavras de tamanho ímpar a letra do meio se mantém

FUGA → AUGF → FGUA

* Palavras de tamanho Par todas as letras alteram em algum momento

PERIGO → ÖERIGP → PGIREO
* 0 0 *

* Em Palavras maiores as letras do meio da Palavra são trocadas de uma vez

→ ~~Após~~ Após isso, pega o caractere subsequente no alfabeto ao seu correspondente na string

DOURADO → DOURADO → DDARUOO
(EEBSVPP)

RECURSÃO

1º Verifica tamanho e retorna se == 1

2º ^{retorna} insere só uma vez caso tamanho == 2 ou 3

3º faz a inversão da primeira e última letra

4º caso seja ímpar, faz verificação especial

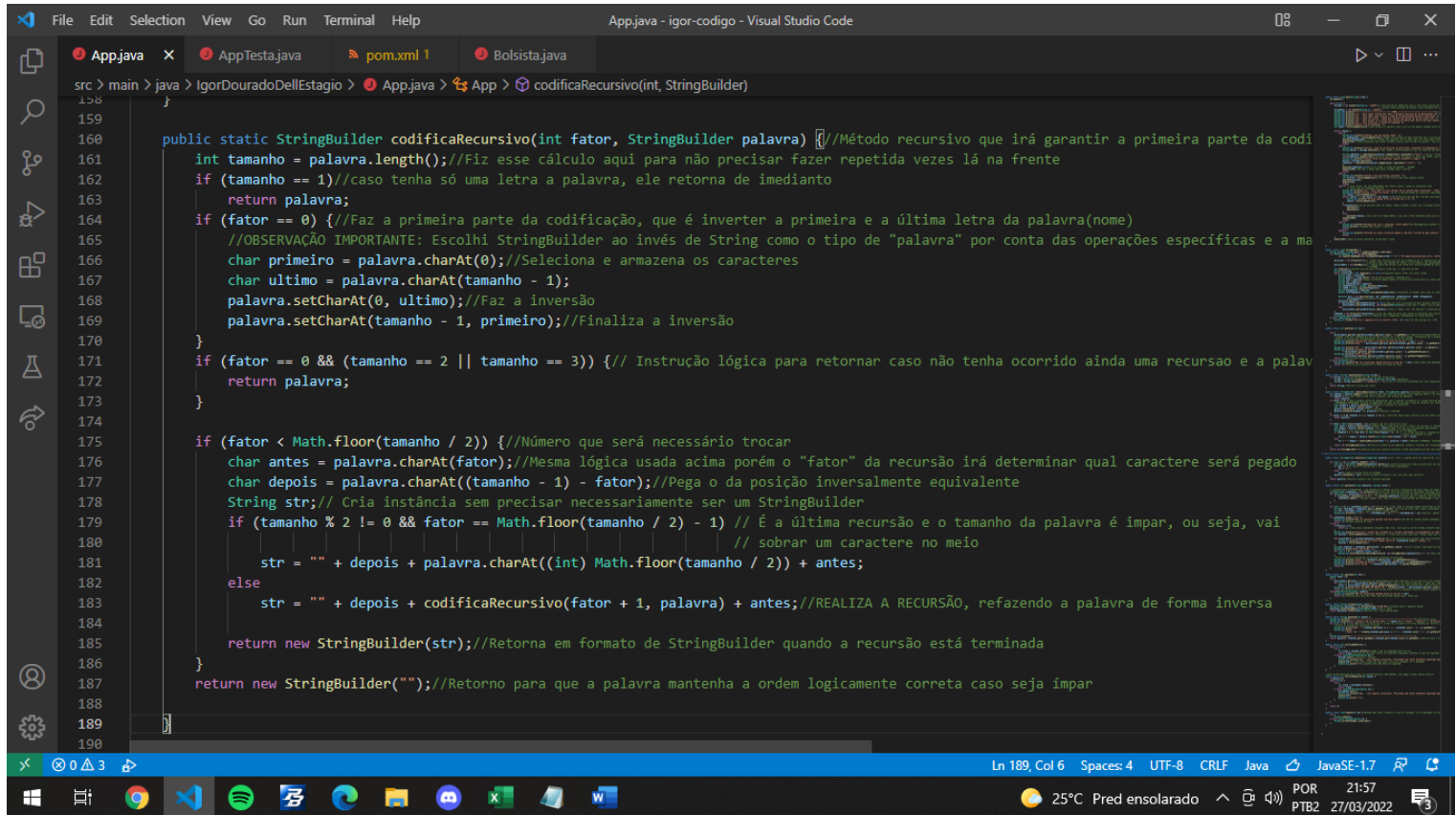
5º → Troca e chama recursivamente nesse sentido

→ Troca + chama recursão + Troca

retorna

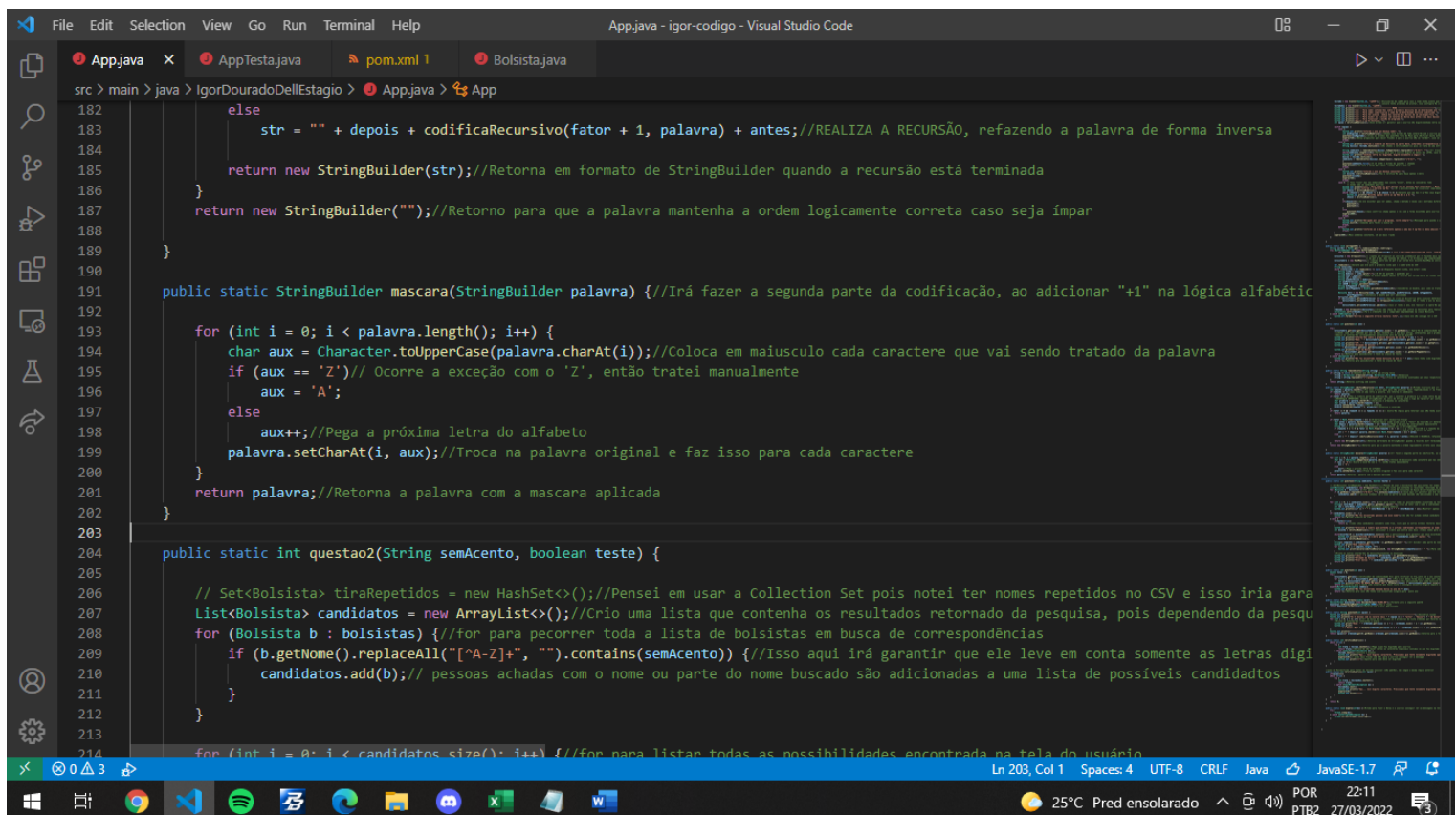
1º → 2ª Palavra
2º → troca primeira e última letra
3º → troca todos da Palavra > 2 ou 3 caracteres e caso seja ímpar o do meio permanece

letra recursão



```
src > main > java > IgorDouradoDellEstagio > App.java > App > codificaRecursivo(int, StringBuilder)
158
159
160 public static StringBuilder codificaRecursivo(int fator, StringBuilder palavra) { //Método recursivo que irá garantir a primeira parte da codi
161     int tamanho = palavra.length(); //Fiz esse cálculo aqui para não precisar fazer repetida vezes lá na frente
162     if (tamanho == 1) //caso tenha só uma letra a palavra, ele retorna de imediato
163         return palavra;
164     if (fator == 0) { //Faz a primeira parte da codificação, que é inverter a primeira e a última letra da palavra(nome)
165         //OBSERVAÇÃO IMPORTANTE: Escolhi StringBuilder ao invés de String como o tipo de "palavra" por conta das operações específicas e a ma
166         char primeiro = palavra.charAt(0); //Seleciona e armazena os caracteres
167         char ultimo = palavra.charAt(tamanho - 1);
168         palavra.setCharAt(0, ultimo); //Faz a inversão
169         palavra.setCharAt(tamanho - 1, primeiro); //Finaliza a inversão
170     }
171     if (fator == 0 && (tamanho == 2 || tamanho == 3)) { // Instrução lógica para retornar caso não tenha ocorrido ainda uma recursao e a palav
172         return palavra;
173     }
174
175     if (fator < Math.floor(tamanho / 2)) { //Número que será necessário trocar
176         char antes = palavra.charAt(fator); //Mesma lógica usada acima porém o "fator" da recursão irá determinar qual caractere será pegado
177         char depois = palavra.charAt((tamanho - 1) - fator); //Pega o da posição inversalmente equivalente
178         String str; // Cria instância sem precisar necessariamente ser um StringBuilder
179         if (tamanho % 2 != 0 && fator == Math.floor(tamanho / 2) - 1) // É a última recursão e o tamanho da palavra é ímpar, ou seja, vai
180             // sobrar um caractere no meio
181             str = "" + depois + palavra.charAt((int) Math.floor(tamanho / 2)) + antes;
182         else
183             str = "" + depois + codificaRecursivo(fator + 1, palavra) + antes; //REALIZA A RECURSÃO, refazendo a palavra de forma inversa
184         return new StringBuilder(str); //Retorna em formato de StringBuilder quando a recursão está terminada
185     }
186     return new StringBuilder(""); //Retorno para que a palavra mantenha a ordem logicamente correta caso seja ímpar
187
188
189
190
```

Logo após apliquei a máscara para “adicionar 1” a cada caractere da palavra primeiramente codificada e ficou seguinte forma:



```
src > main > java > IgorDouradoDellEstagio > App.java > App
182     else
183         str = "" + depois + codificaRecursivo(fator + 1, palavra) + antes; //REALIZA A RECURSÃO, refazendo a palavra de forma inversa
184     return new StringBuilder(str); //Retorna em formato de StringBuilder quando a recursão está terminada
185 }
186 return new StringBuilder(""); //Retorno para que a palavra mantenha a ordem logicamente correta caso seja ímpar
187
188
189 }
190
191 public static StringBuilder mascara(StringBuilder palavra) { //Irá fazer a segunda parte da codificação, ao adicionar "+1" na lógica alfabétic
192
193     for (int i = 0; i < palavra.length(); i++) {
194         char aux = Character.toUpperCase(palavra.charAt(i)); //Coloca em maiusculo cada caractere que vai sendo tratado da palavra
195         if (aux == 'Z') // Ocorre a exceção com o 'Z', então tratei manualmente
196             aux = 'A';
197         else
198             aux++; //Pega a próxima letra do alfabeto
199         palavra.setCharAt(i, aux); //Troca na palavra original e faz isso para cada caractere
200     }
201     return palavra; //Retorna a palavra com a mascara aplicada
202 }
203
204 public static int questao2(String semAcento, boolean teste) {
205
206     // Set<Bolsista> tiraRepetidos = new HashSet<>(); //Pensei em usar a Collection Set pois notei ter nomes repetidos no CSV e isso iria gara
207     List<Bolsista> candidatos = new ArrayList<>(); //Crio uma lista que contenha os resultados retornado da pesquisa, pois dependendo da pesqu
208     for (Bolsista b : bolsistas) { //for para percorrer toda a lista de bolsistas em busca de correspondências
209         if (b.getNome().replaceAll("[^A-Z]+", "").contains(semAcento)) { //Isso aqui irá garantir que ele leve em conta somente as letras digi
210             candidatos.add(b); // pessoas achadas com o nome ou parte do nome buscado são adicionadas a uma lista de possíveis candidatos
211         }
212     }
213
214     for (int i = 0; i < candidatos.size(); i++) { //for para listar todas as possibilidades encontrada na tela do usuário
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
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2
```


File Edit Selection View Go Run Terminal Help App.java - igor-codigo - Visual Studio Code

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 3

Microsoft Windows [versão 10.0.19044.1586]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\igord\igor-codigo> cmd /C ""C:\Program Files\AdoptOpenJDK\jdk-11.0.8.10-hotspot\bin\java.exe" @C:\Users\igord\AppData\Local\Temp\cp_x03f4k114s5jwbryi4e25bb6.argfile IgorDouradoDellEstagio.App ""

```
----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à media dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
2
Informe o nome de um bolsista ou parte dele, acharemos correspondência se houver:
Mariá SILVA m
1 MAGALHAES RIBEIRO
2 MAGALHAES RIBEIRO
3 SILVA MARTINS
Selecione o numero que contenha os 2 últimos sobrenomes correspondentes ao nome buscado:
3
Nome codificado: TOBAVB NJSBB TWJB NOJUSBT
Ano: 2013
Entidade de Ensino: UNIVERSIDADE DO ESTADO DA BAHIA
Valor bolsa: 765.0

----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à media dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:

```

Ln 203, Col 1 Spaces: 4 UTF-8 CRLF Java JavaSE-1.7

25°C Pred ensolarado 22:18 27/03/2022

File Edit Selection View Go Run Terminal Help AppTesta.java - igor-codigo - Visual Studio Code

TESTING

Filter (e.g. text, !exclude, @tag)

1/1 tests passed (100%)

- ✓ igor-codigo 8.0s
 - ✓ IgorDouradoDellEstagio 8.0s
 - ✓ AppTesta 8.0s
 - ✓ testaQuestao1True() 236ms
 - ✓ testaQuestao1False() 1.9s
 - ✓ testaQuestao2True() 1.8s
 - ✓ testaQuestao2False() 2.5s
 - ✓ testaQuestao3True() 297ms
 - ✓ testaQuestao3False() 403ms
 - ✓ testaQuestao4Opcao1() 303ms
 - ✓ testaQuestao4Opcao2() 317ms
 - ✓ testaRemoverDeAcento() 17ms
 - ✓ testaCodificadorRecursivo1() 2.0ms
 - ✓ testaCodificadorRecursivo2() 1.0ms
 - ✓ testaCodificadorRecursivo3() 1.0ms
 - ✓ testaCodificadorRecursivo4() 3.0ms
 - ✓ testaMascara1() 4.0ms
 - ✓ testaMascara2() 4.0ms
 - ✓ testaMascara3() 2.0ms
 - ✓ testaMascara4() 1.0ms
 - ✓ testaFormata() 136ms

src > test > java > IgorDouradoDellEstagio > AppTesta.java > AppTesta

```
9 public void testaQuestao1True() {
10     App.carregaCSV();// Aqui também estou testando se a leitura do CSV correta e é necessário esse
11     // método aqui para o funcionamento dos testes que envolvem a leitura e as listas
12     assertEquals(1, App.questao1(2013));// Verifica se acha e mantém esse padrão de devolver 1 caso ach
13 }
14
15
16 @Test
17 public void testaQuestao1False() {
18     App.carregaCSV();
19     assertEquals(-1, App.questao1(2009));// Verifica se não acha e mantém esse padrão de devolver -1 ca
20 }
21
22 @Test
23 public void testaQuestao2True() {
24     App.carregaCSV();
25     assertEquals(1, App.questao2("LAISE", true));// Nomes não completo mas ele consegue achar correspon
26 }
27
28 @Test
29 public void testaQuestao2False() {
30     App.carregaCSV();
31     assertEquals(-1, App.questao2("LAISEE", true));// Não há esse nome na lista (com um 'E' a mais no LA
32 }
33
34 @Test
35 public void testaQuestao3True() {
36     App.carregaCSV();
37     assertEquals(1, App.questao3(2015));// Ano que consta
38 }
39
40 @Test
41 public void testaQuestao3False() {
42     App.carregaCSV();

```

Ln 21, Col 4 (507 selected) Spaces: 4 UTF-8 CRLF Java JavaSE-1.7

25°C Pred ensolarado 22:19 27/03/2022


```
src > test > java > IgorDouradoDellEstagio > AppTest.java > AppTest > testaCodificadorRecursivo2()

56
57
58 @Test
59 public void testaRemoverDeAcento() {
60     assertEquals("acentuada", App.removeAcentos("âçêñtúádã")); // Troca as letras acentuadas por suas letras
61     // correspondentes sem acentuação, elas serão
62     // colocadas em maiúsculas no metodo de mascara
63 }
64
65 // Fiz 4 testes a seguir, com as 3 palavras de exemplo do teste e mais o meu
66 // sobrenome como exemplo
67 // Para o metodo de codificadorRecursivo e o de mascara
68 // as palavras serão tratadas NECESSARIAMENTE em maiúsculas no metodo mascara,
69 // metodo interligado com o anterior e esse próximo
70 @Test
71 public void testaCodificadorRecursivo1() {
72     assertEquals(new StringBuilder("PGIREO").toString(),
73                 App.codificaRecursivo(0, new StringBuilder("PERIGO")).toString());
74 }
75
76 @Test
77 public void testaCodificadorRecursivo2() {
78     assertEquals(new StringBuilder("FUGA").toString(),
79                 App.codificaRecursivo(0, new StringBuilder("FUGA")).toString());
80 }
81
82 @Test
83 public void testaCodificadorRecursivo3() {
84     assertEquals(new StringBuilder("ZAP").toString(),
85                 App.codificaRecursivo(0, new StringBuilder("PAZ")).toString());
86 }
87
88 @Test
89 public void testaCodificadorRecursivo4() {
90     assertEquals(new StringBuilder("DOARUO").toString(),
91                 App.codificaRecursivo(0, new StringBuilder("DOURADO")).toString());
92 }
93
94 @Test
95 public void testaMascara1() {
96     assertEquals(new StringBuilder("QHJSFP").toString(),
97                 App.mascara(App.codificaRecursivo(0, new StringBuilder("perigo")).toString());
98 }
99
```

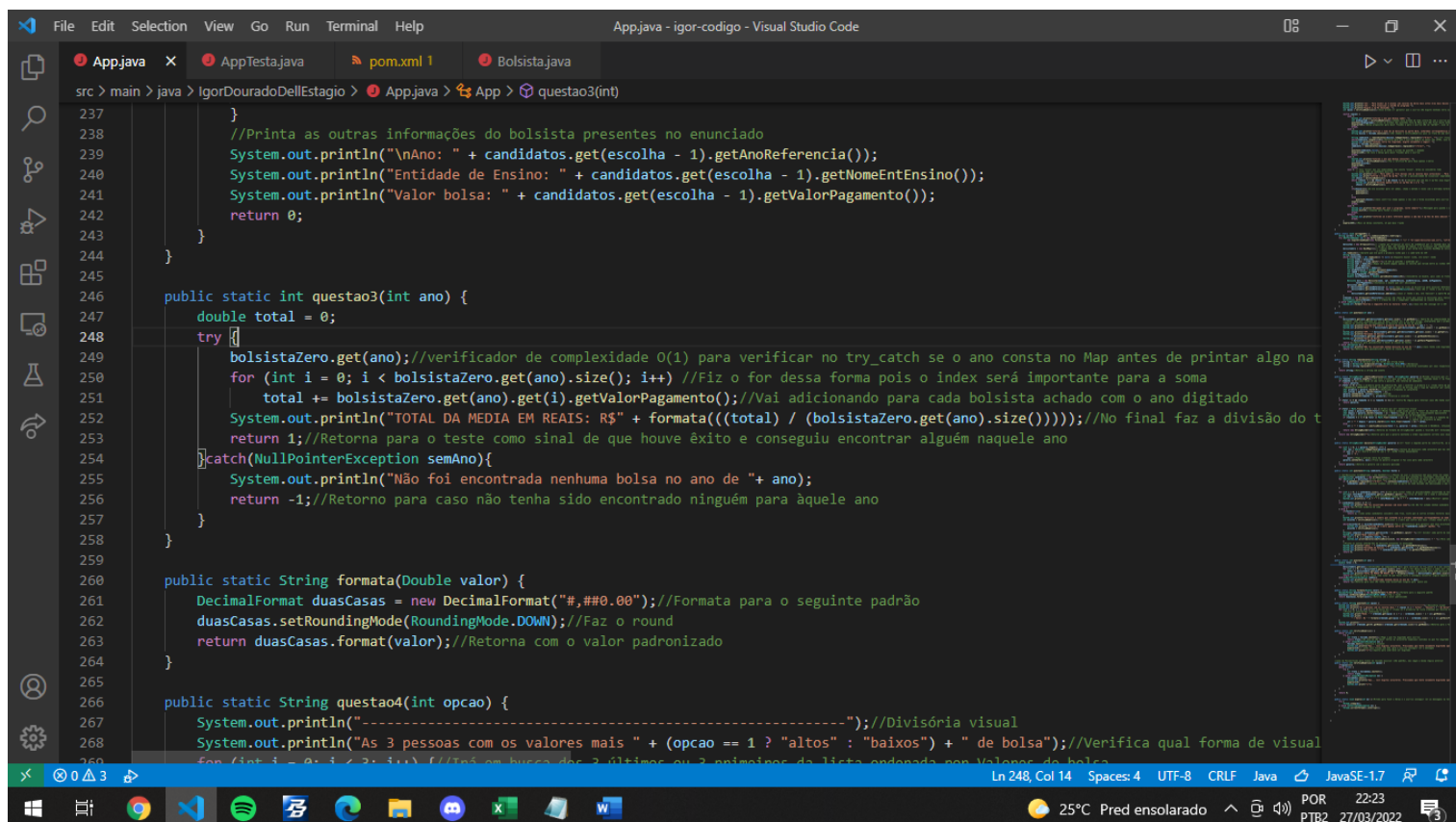
```
src > test > java > IgorDouradoDellEstagio > AppTest.java > AppTest > testaMascara2()

79
80
81 @Test
82 public void testaCodificadorRecursivo3() {
83     assertEquals(new StringBuilder("ZAP").toString(),
84                 App.codificaRecursivo(0, new StringBuilder("PAZ")).toString());
85 }
86
87 @Test
88 public void testaCodificadorRecursivo4() {
89     assertEquals(new StringBuilder("DOARUO").toString(),
90                 App.codificaRecursivo(0, new StringBuilder("DOURADO")).toString());
91 }
92
93 @Test
94 public void testaMascara1() {
95     assertEquals(new StringBuilder("QHJSFP").toString(),
96                 App.mascara(App.codificaRecursivo(0, new StringBuilder("perigo")).toString());
97 }
98
99 @Test
100 public void testaMascara2() {
101     assertEquals(new StringBuilder("GHVB").toString(),
102                 App.mascara(App.codificaRecursivo(0, new StringBuilder("fuGA")).toString()); // metade minuscula e
103     // metade maiuscula para
104     // testar
105 }
106
107 @Test
108 public void testaMascara3() {
109     assertEquals(new StringBuilder("ABQ").toString(),
110                 App.mascara(App.codificaRecursivo(0, new StringBuilder("paz")).toString());
111 }
112
113 @Test
114 public void testaMascara4() {
115     assertEquals(new StringBuilder("EEBSVPP").toString(),
116                 App.mascara(App.codificaRecursivo(0, new StringBuilder("dourado")).toString());
117 }
118
```

Questão 3- Explicação:

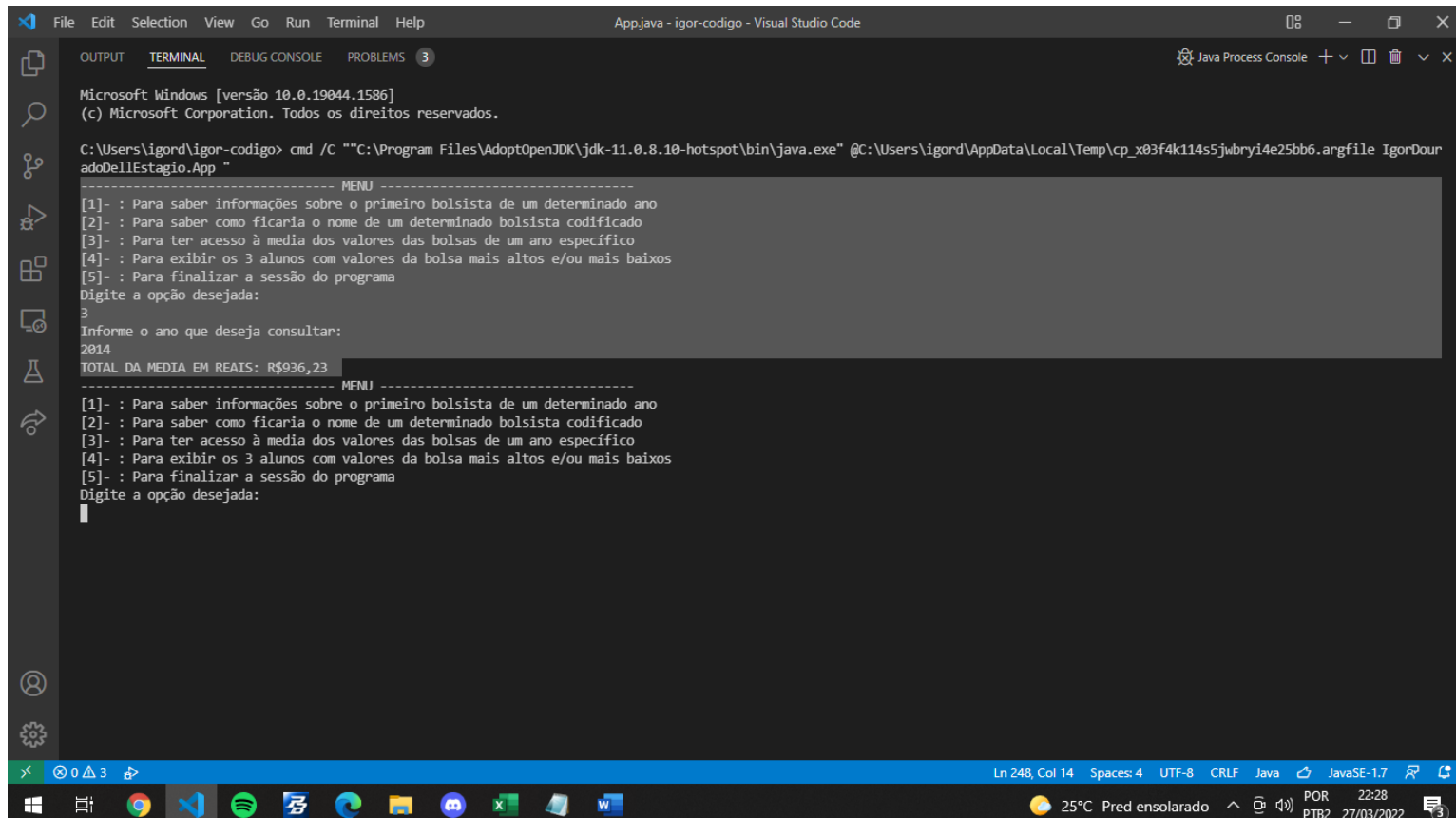
Aqui novamente me utilizei do Map (dicionário) criado lá em cima, pesquiso o ano como key do Map e, se constar, é retornado uma lista com todos os bolsistas daquele ano. Então aí eu somo todos eles daquele ano e armazeno em uma variável e no final eu divido pelo total de bolsistas que constam naquele ano, resultando na média de valores de bolsa do ano pesquisado, que depois eu formatei para ficar no padrão de

moeda brasileira ou mostra a mensagem que não consta bolsistas naquele ano pesquisado.



```
src > main > java > IgorDouradoDellEstagio > App.java > App > questao3(int)
237     }
238     //Printa as outras informações do bolsista presentes no enunciado
239     System.out.println("\nAno: " + candidatos.get(escolha - 1).getAnoReferencia());
240     System.out.println("Entidade de Ensino: " + candidatos.get(escolha - 1).getNomeEntEnsino());
241     System.out.println("Valor bolsa: " + candidatos.get(escolha - 1).getValorPagamento());
242     return 0;
243 }
244
245
246 public static int questao3(int ano) {
247     double total = 0;
248     try {
249         bolsistaZero.get(ano); //verificador de complexidade O(1) para verificar no try_catch se o ano consta no Map antes de printar algo na
250         for (int i = 0; i < bolsistaZero.get(ano).size(); i++) //fiz o for dessa forma pois o index será importante para a soma
251             total += bolsistaZero.get(ano).get(i).getValorPagamento(); //Vai adicionando para cada bolsista achado com o ano digitado
252         System.out.println("TOTAL DA MEDIA EM REAIS: R$" + formata(((total) / (bolsistaZero.get(ano).size())))); //No final faz a divisão do t
253         return 1; //Retorna para o teste como sinal de que houve êxito e conseguiu encontrar alguém naquele ano
254     } catch (NullPointerException semAno) {
255         System.out.println("Não foi encontrada nenhuma bolsa no ano de " + ano);
256         return -1; //Retorno para caso não tenha sido encontrado ninguém para aquele ano
257     }
258 }
259
260 public static String formata(Double valor) {
261     DecimalFormat duasCasas = new DecimalFormat("#,##0.00"); //Formata para o seguinte padrão
262     duasCasas.setRoundingMode(RoundingMode.DOWN); //Faz o round
263     return duasCasas.format(valor); //Retorna com o valor padronizado
264 }
265
266 public static String questao4(int opcao) {
267     System.out.println("-----"); //Divisória visual
268     System.out.println("As 3 pessoas com os valores mais " + (opcao == 1 ? "altos" : "baixos") + " de bolsa"); //Verifica qual forma de visual
269     for (int i = 0; i < 3; i++) { //Três em busca dos 3 últimos ou 3 primeiros da lista ordenada por valores de bolsa
```

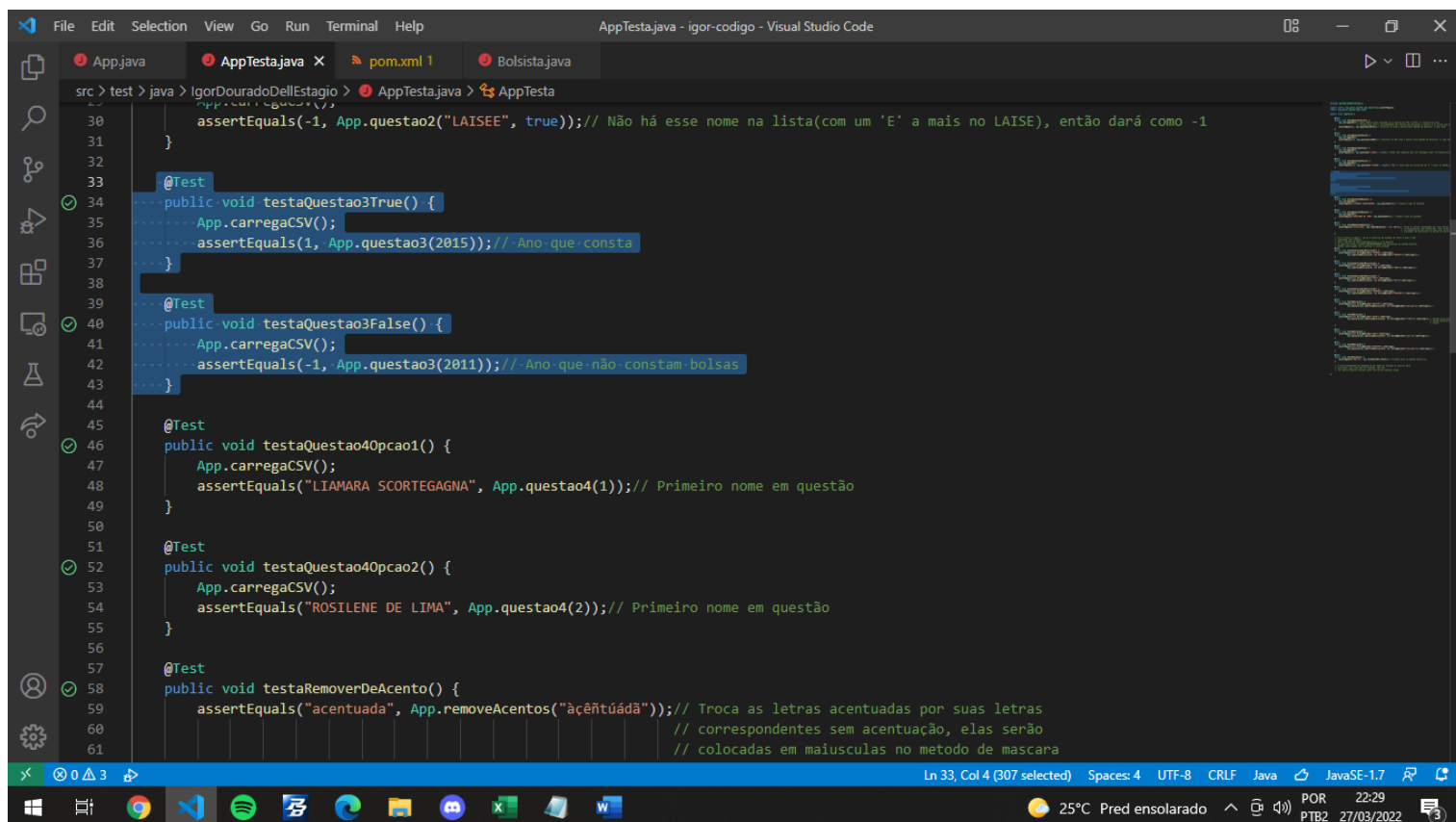
Execução e testes unitários da questão 3:



```
Microsoft Windows [versão 10.0.19044.1586]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\igord\Igor-codigo> cmd /C ""C:\Program Files\AdoptOpenJDK\jdk-11.0.8-hotspot\bin\java.exe" @C:\Users\igord\AppData\Local\Temp\cp_x03f4k114s5jwbryi4e25bb6.argfile IgorDouradoDellEstagio.App ""

----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à media dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
3
Informe o ano que deseja consultar:
2014
TOTAL DA MEDIA EM REAIS: R$936,23
----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à media dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
|
```



```
src > test > java > IgorDouradoDellEstagio > AppTest.java > AppTest.java
30 assertEquals(-1, App.questao2("LAISEE", true)); // Não há esse nome na lista (com um 'E' a mais no LAISE), então dará como -1
31 }
32
33 @Test
34 public void testaQuestao3True() {
35     App.carregaCSV();
36     assertEquals(1, App.questao3(2015)); // Ano que consta
37 }
38
39 @Test
40 public void testaQuestao3False() {
41     App.carregaCSV();
42     assertEquals(-1, App.questao3(2011)); // Ano que não constam bolsas
43 }
44
45 @Test
46 public void testaQuestao40pcao1() {
47     App.carregaCSV();
48     assertEquals("LIAMARA SCORTEGAGNA", App.questao4(1)); // Primeiro nome em questão
49 }
50
51 @Test
52 public void testaQuestao40pcao2() {
53     App.carregaCSV();
54     assertEquals("ROSILENE DE LIMA", App.questao4(2)); // Primeiro nome em questão
55 }
56
57 @Test
58 public void testaRemoverDeAcento() {
59     assertEquals("acentuada", App.removeAcentos("âêñtúádã")); // Troca as letras acentuadas por suas letras
60     // correspondentes sem acentuação, elas serão
61     // colocadas em maiúsculas no método de máscara
```

Questão 4 – Explicação:

Para essa questão eu pensei desde a leitura do CSV (como aconteceu com as questões anteriores também) e criei a cópia do ArrayList de bolsistas para que a ordem original não seja alterada, e então eu ordenei por valor de pagamento da bolsa esse ArrayList copiado em ordem decrescente com um comparador que implementei na classe de Bolsista e já ordenei na leitura mesmo. Após essa List ser ordenada, no método da questão, dependendo da opção escolhida pelo o usuário para a exibição eu peguei os 3 últimos (mais baixos) ou os 3 primeiros (mais altos) e printei suas informações, dependendo da escolha do usuário, é possível printar ambos.

```
src > main > java > IgorDouradoDellEstagio > App.java > App > questao4(int)
252     System.out.println("TOTAL DA MEDIA EM REAIS: R$" + formata(((total) / (bolsistaZero.get(ano).size())))); //No final faz a divisao do t
253     return 1; //Retorna para o teste como sinal de que houve sucesso e conseguiu encontrar alguem naquele ano
254 } catch (NullPointerException semAno) {
255     System.out.println("Não foi encontrada nenhuma bolsa no ano de " + ano);
256     return -1; //Retorna para caso não tenha sido encontrado ninguém para aquele ano
257 }
258 }
259
260 public static String formata(Double valor) {
261     DecimalFormat duasCasas = new DecimalFormat("#,##0.00"); //Formata para o seguinte padrão
262     duasCasas.setRoundingMode(RoundingMode.DOWN); //Faz o round
263     return duasCasas.format(valor); //Retorna com o valor padronizado
264 }
265
266 public static String questao4(int opcao) {
267     System.out.println("-----"); //Divisória visual
268     System.out.println("As 3 pessoas com os valores mais " + (opcao == 1 ? "altos" : "baixos") + " de bolsa"); //Verifica qual forma de visual
269     for (int i = 0; i < 3; i++) { //Irá em busca dos 3 últimos ou 3 primeiros da lista ordenada por Valores de bolsa
270         //Printa as informações referentes a ele(a)
271         System.out.print("Nome: " + ordenado.get(opcao == 1 ? i : (ordenado.size() - 1 - i)).getNome());
272         System.out.print(
273             " / Valor: R$" + formata(ordenado.get(opcao == 1 ? i : (ordenado.size() - 1 - i)).getValorPagamento()) + "\n");
274     }
275     System.out.println("-----");
276     return opcao == 1 ? ordenado.get(0).getNome() : ordenado.get(ordenado.size() - 1).getNome(); //Retorna para o Teste
277 }
278
279 public static int verificaNumerico() {
280     while (true) {
281         try {
282             int trata = teclado.nextInt(); //Pega o que foi digitado pelo usuário
283             return trata; //Em caso de não haver letras ou caracteres especiais contidos no que foi digitado e apenas números, ele retorna
284         } catch (InputMismatchException e) {
```

```
src > main > java > IgorDouradoDellEstagio > Bolsista.java > Bolsista
11 private String dsPagamento;
12 private double valorPagamento; //Por se tratar de uma moeda que pode ter representação decimal importante para a análise, coloquei como tipo
13 public Bolsista(String nome, String cpf, String nomeEntEnsino, int anoReferencia, int cdSGB,
14     String dsPagamento, double valorPagamento) { //Instancio o construtor
15     this.nome = nome;
16     this.cpf = cpf;
17     this.nomeEntEnsino = nomeEntEnsino;
18     this.anoReferencia = anoReferencia;
19     this.cdSGB = cdSGB;
20     this.dsPagamento = dsPagamento;
21     this.valorPagamento = valorPagamento;
22 }
23
24 @Override
25 public int compareTo(Bolsista ordenador) { //Para exercer o comparador entre os bolsistas baseado no valor de pagamento da bolsa, fiz para fic
26     if (this.valorPagamento > ordenador.getValorPagamento()) {
27         return -1;
28     }
29     if (this.valorPagamento < ordenador.getValorPagamento()) {
30         return 1;
31     }
32     return 0;
33 }
34
35 //Abaixo os Gets e os Sets para cada atributo do objeto
36 public String getNome() {
37     return nome;
38 }
39 public void setNome(String nome) {
40     this.nome = nome;
41 }
42 public String getCpf() {
```

Execução e testes unitários da questão 4:

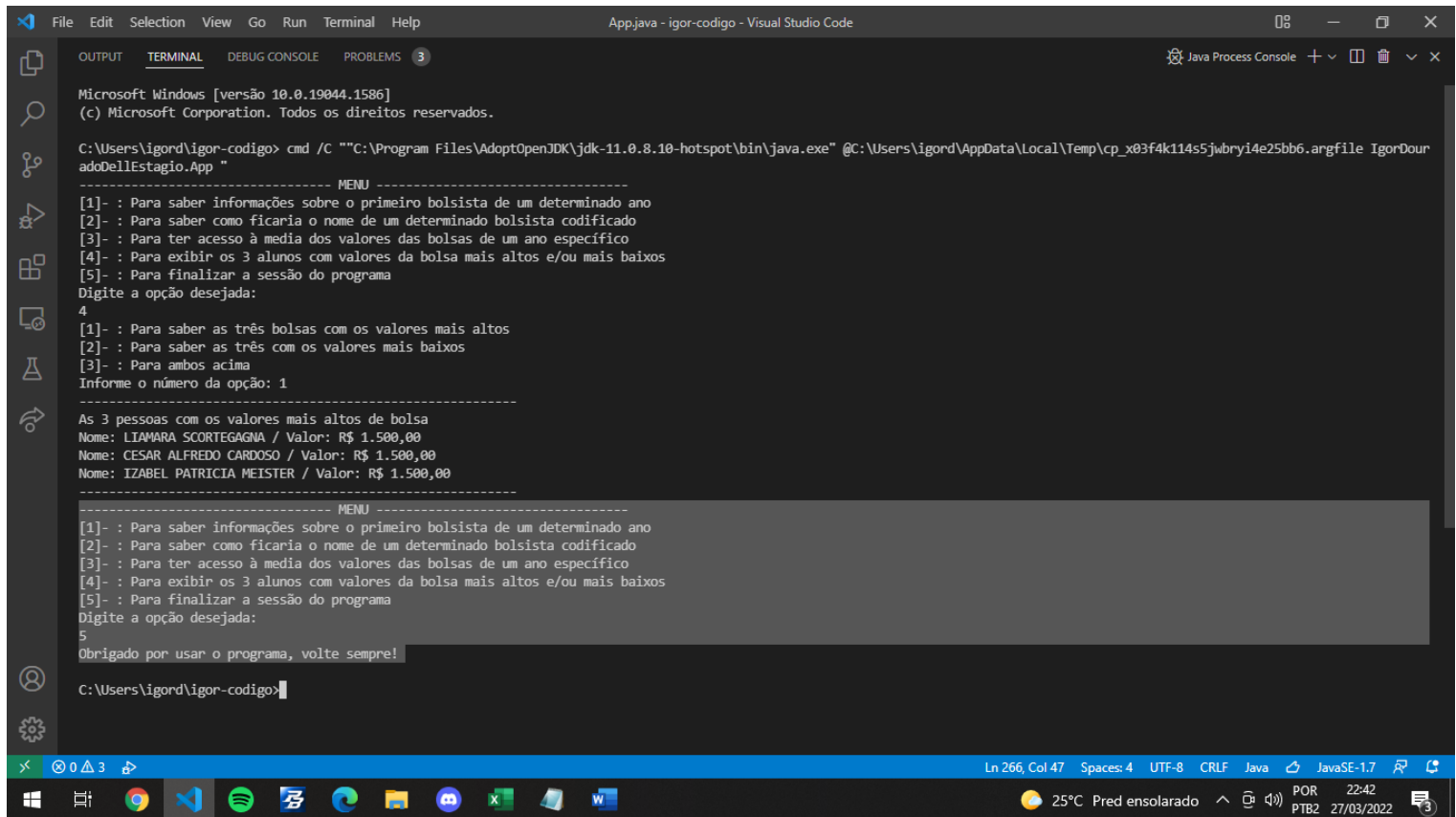

```
C:\Users\igord\igor-codigo> cmd /c ""C:\Program Files\AdoptOpenJDK\jdk-11.0.8.10-hotspot\bin\java.exe" @C:\Users\igord\AppData\Local\Temp\cp_x03f4k114s5jwbryi4e255bb6.argfile IgorDouradoDellEstagio.App ""

----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à media dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
4
[1]- : Para saber as três bolsas com os valores mais altos
[2]- : Para saber as três com os valores mais baixos
[3]- : Para ambos acima
Informe o número da opção: 3
-----
As 3 pessoas com os valores mais altos de bolsa
Nome: LTAMARA SCORTEGAGNA / Valor: R$ 1.500,00
Nome: CESAR ALFREDO CARDOSO / Valor: R$ 1.500,00
Nome: IZABEL PATRICIA MEISTER / Valor: R$ 1.500,00
-----
As 3 pessoas com os valores mais baixos de bolsa
Nome: ROSILENE DE LIMA / Valor: R$ 765,00
Nome: GLAUCIA ANK GUARINO / Valor: R$ 765,00
Nome: NATALIA NOGUEIRA SARAIVA / Valor: R$ 765,00
-----
----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à media dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
|
```

```
src > test > java > IgorDouradoDellEstagio > AppTesta.java > AppTesta

38
39
40 @Test
41 public void testaQuestao3False() {
42     App.carregaCSV();
43     assertEquals(-1, App.questao3(2011)); // Ano que não constam bolsas
44 }
45
46 @Test
47 public void testaQuestao4Opcao1() {
48     App.carregaCSV();
49     assertEquals("LIAMARA SCORTEGAGNA", App.questao4(1)); // Primeiro nome em questão
50 }
51
52 @Test
53 public void testaQuestao4Opcao2() {
54     App.carregaCSV();
55     assertEquals("ROSILENE DE LIMA", App.questao4(2)); // Primeiro nome em questão
56 }
57
58 @Test
59 public void testaRemoverDeAcento() {
60     assertEquals("acentuada", App.removeAcentos("ãçêñtúáãä")); // Troca as letras acentuadas por suas le
61     // correspondentes sem acentuação, elas s
62     // colocadas em maiúsculas no metodo de m
63 }
64
65 // Fiz 4 testes a seguir, com as 3 palavras de exemplo do teste e mais o meu
66 // sobrenome como exemplo
67 // Para o metodo de codificadorRecurso e o de mascara
68 // as palavras serão tratadas NECESSARIAMENTE em maiúsculas no metodo mascara,
69 // metodo interligado com o anterior e esse próximo
70 @Test
```

Questão 5- Demonstração saindo do programa:



```
Microsoft Windows [versão 10.0.19044.1586]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\igord\igor-codigo> cmd /C ""C:\Program Files\AdoptOpenJDK\jdk-11.0.8.10-hotspot\bin\java.exe" @C:\Users\igord\AppData\Local\Temp\cp_x03f4k114s5jwbry14e25bb6.argfile IgorDouradoBellEstagio.App ""

----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à media dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
4
[1]- : Para saber as três bolsas com os valores mais altos
[2]- : Para saber as três com os valores mais baixos
[3]- : Para ambos acima
Informe o número da opção: 1
-----
As 3 pessoas com os valores mais altos de bolsa
Nome: LIAMARA SCORTEGAGNA / Valor: R$ 1.500,00
Nome: CESAR ALFREDO CARDOSO / Valor: R$ 1.500,00
Nome: IZABEL PATRICIA MEISTER / Valor: R$ 1.500,00
-----
----- MENU -----
[1]- : Para saber informações sobre o primeiro bolsista de um determinado ano
[2]- : Para saber como ficaria o nome de um determinado bolsista codificado
[3]- : Para ter acesso à media dos valores das bolsas de um ano específico
[4]- : Para exibir os 3 alunos com valores da bolsa mais altos e/ou mais baixos
[5]- : Para finalizar a sessão do programa
Digite a opção desejada:
5
Obrigado por usar o programa, volte sempre!

C:\Users\igord\igor-codigo>
```

Referências que usei em partes de dúvida durante o código:

how to know if a string is contained in another java:

<https://stackoverflow.com/questions/2275004/in-java-how-do-i-check-if-a-string-contains-a-substring-ignoring-case>

imprimir acento na tela:

<https://itqna.net/questions/11564/allow-special-characters-java-keyboard-input>

como fazer para ignorar caracteres especiais no java:

<https://stackoverflow.com/questions/31716511/how-to-ignore-special-characters-and-spaces-in-string>

How to swap String characters in Java?

<https://stackoverflow.com/questions/956199/how-to-swap-string-characters-in-java>

How do I increment a variable to the next or previous letter in the alphabet?

<https://stackoverflow.com/questions/2899301/how-do-i-increment-a-variable-to-the-next-or-previous-letter-in-the-alphabet>

Comparator

<https://beginnersbook.com/2013/12/java-arraylist-of-object-sort-example-comparable-and-comparator/>

Testes unitários:

https://www.youtube.com/watch?v=60yrTfVdFwo&ab_channel=software_technology