



FUNDAÇÃO EDSON QUEIROZ

UNIVERSIDADE DE FORTALEZA - UNIFOR

CENTRO DE TECNOLOGIA

CURSO ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

T164-87 PROGRAMAÇÃO ORIENTADA A OBJETOS

Relatório de Desenvolvimento de um jogo Quiz

Equipe:

Felipe Martins

Jardiana Galvão

Samuel Chagas

Saul Santos

PROGRAMAÇÃO ORIENTADA A OBJETOS - POO

PROF. SAMUEL BARROCAS

TRABALHO FINAL AV3

EQUIPE:

NOME	FUNÇÃO
Felipe Martins	Responsável pelos Testes
Jardiana Galvão	Responsável pelos diagramas e relatório
Samuel Chagas	Responsável pela interface e codificação
Saul Santos	Responsável pela interface e codificação

DETALHES:

Projeto: Quiz Game em Java

Tema: O Quiz “CodeJava TheQuiz” abordará questões sobre conceitos relativos à Programação Orientada a Objetos abordados durante o semestre 2 do Curso Análise e Desenvolvimento de Sistemas, Unifor, ministrada pelo Prof. Samuel Barrocas.

APRESENTAÇÃO DO JOGO E REGRAS:

O jogo consiste em uma lista de 10 perguntas sobre linguagem de programação de programação com itens de 'a' até 'd' para o jogador escolher o item correto. À medida que o jogador vai selecionando as respostas o jogo verifica se o item escolhido condiz com a resposta correta e salva a pontuação caso seja a resposta certa. No final das 10 perguntas o jogo apresenta o total da pontuação que o jogador atingiu.

ASPECTOS SOBRE O DESENVOLVIMENTO DO PROJETO

Introdução

Este relatório documenta o desenvolvimento bem-sucedido de um jogo do tipo Quiz em Java, denominado "JavaQuiz". O projeto envolveu diversas etapas, desde o planejamento inicial até a implementação e testes finais. O JavaQuiz visa proporcionar uma experiência interativa e educativa, desafiando os jogadores com perguntas de conhecimento sobre programação Orientada a Objetos.

1. Objetivo do Projeto

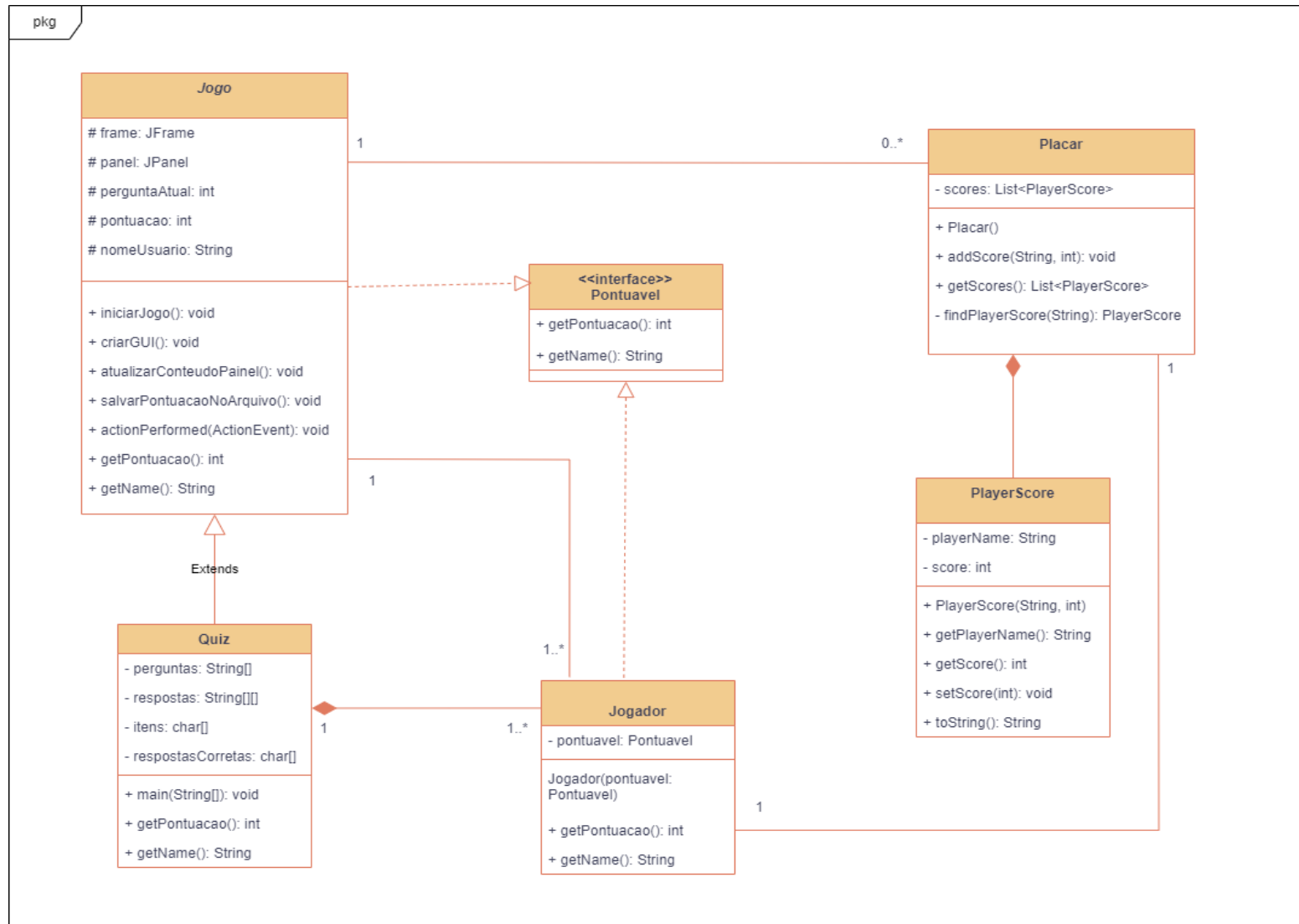
O objetivo principal do projeto foi criar um jogo de Quiz interativo, utilizando a linguagem de programação Java. O JavaQuiz foi concebido para entreter e educar de forma lúdica, abordando perguntas relacionadas à Programação Orientada a Objetos.

2. Metodologia de Desenvolvimento

As etapas de desenvolvimento foram realizadas por meio de sprints semanais com o acompanhamento do professor, e de forma resumida se dividiram em:

- **Planejamento:** Definição das funcionalidades, elaboração das categorias de perguntas e criação de um esboço inicial do design e diagrama de classes.
- **Codificação:** Desenvolvimento do código-fonte em Java, utilizando boas práticas de programação e orientação a objetos.
- **Testes:** Realização de testes unitários e integração para garantir a estabilidade e correção de bugs.
- **Aprimoramento:** Incorporação de feedbacks e otimizações para melhorar a experiência do usuário, bem como os requisitos solicitados para o projeto e sugestões obtidas durante as sprints.

3. Diagrama de Classes



4. Requisitos do Projeto

A construção do jogo incorporou os conceitos estudados na disciplina de programação Orientada a Objetos, conforme detalhado a seguir:

4.1. Classe Abstrata e Herança:

- Classe Abstrata: A classe abstrata é representada por *abstract class Jogo*. Ela é abstrata e contém métodos abstratos.

```
31 abstract class Jogo implements ActionListener, Pontuavel {
32     protected JFrame frame;
33     protected JPanel panel;
34     protected int perguntaAtual = 0;
35     protected int pontuacao = 0;
36     protected String nomeUsuario;
37
38     public void iniciarJogo() {
39         SwingUtilities.invokeLater(() -> {
40             criarGUI();
41         });
42     }
43
44     protected abstract void criarGUI();
45     protected abstract void atualizarConteudoPainel();
46     protected abstract void exibirPlacarLideres();
47     protected abstract void salvarPontuacaoNoArquivo();
48
49     @Override
50     public abstract void actionPerformed(ActionEvent e);
51 }
52
```

- Herança: A classe `Quiz` herda da classe abstrata `Jogo` com *public class Quiz extends Jogo*.

```

57 public class Quiz extends Jogo {
58
59     private JFrame frame;
60     private JPanel panel;
61     private int perguntaAtual = 0;
62     private int pontuacao = 0;
63     private String nomeUsuario;
64
65     String[] perguntas = {"
77
78     String[][] respostas = {"
90
91     char[] itens = {'A', 'B', 'C', 'D'};
92
93     char[] respostasCorretas = {'D', 'C', 'B', 'A', 'D', 'B', 'A', 'C', 'B', 'B'};
94
95     public static void main(String[] args) {
96         Quiz quiz = new Quiz();
97         quiz.iniciarJogo();
98         Jogador jogador = new Jogador(quiz);
99         System.out.println(jogador.getName());
100        System.out.println("O quiz vale de 0 a " + jogador.getPontuacao());
101    }
102
103    @Override
104    public int getPontuacao() {
105        return pontuacao;
106    }
107
108    @Override
109    public String getName() {
110        return nomeUsuario;
111    }

```

4.2. Interface e Implementação:

- Interface: A interface é representada por interface *Pontuavel*. Ela define métodos *getPontuacao* e *getName*.

```

9 interface Pontuavel {
10     int getPontuacao();
11     String getName();
12 }
13
14 class Jogador implements Pontuavel {
15     public Jogador(Pontuavel pontuavel) {
16     }
17
18     @Override
19     public int getPontuacao() {
20         return 10;
21     }
22
23     @Override
24     public String getName() {
25         return "Bem vindo Aluno de POO";
26     }
27 }
28

```

- Implementação: A classe *Jogador* implementa a interface *Pontuavel* com *class Jogador implements Pontuavel*.

```

14 class Jogador implements Pontuavel {
15     public Jogador(Pontuavel pontuavel) {
16     }
17
18     @Override
19     public int getPontuacao() {
20         return 10;
21     }
22
23     @Override
24     public String getName() {
25         return "Bem vindo Aluno de POO";
26     }
27 }
28
29 abstract class Jogo implements ActionListener, Pontuavel {
30     protected JFrame frame;
31     protected JPanel panel;
32     protected int perguntaAtual = 0;
33     protected int pontuacao = 0;
34     protected String nomeUsuario;
35
36     public void iniciarJogo() {
37         SwingUtilities.invokeLater(() -> {
38             criarGUI();
39         });
40     }

```

4.3. Sobrescrita de Método:

- Há sobrescrita de métodos nas classes *Jogador* (métodos *getPontuacao* e *getName*) e na classe *Quiz* (métodos *getPontuacao*, *getName*, *criarGUI*, *atualizarConteudoPainel*, *exibirPlacarLideres*, *salvarPontuacaoNoArquivo* e *actionPerformed*).

```

14 class Jogador implements Pontuavel {
15     public Jogador(Pontuavel pontuavel) {
16     }
17
18     @Override
19     public int getPontuacao() {
20         return 10;
21     }
22
23     @Override
24     public String getName() {
25         return "Bem vindo Aluno de POO";
26     }
27 }
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55 public class Quiz extends Jogo {
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

```

4.4. Atributos Privados com Métodos Get e Set:

- Todos os atributos nas classes *Jogo*, *Quiz*, *Jogador*, e *PlayerScore* são privados. Métodos *get* e *set* estão implementados para esses atributos.

```

41     public static class PlayerScore {
42         private String playerName;
43         private int score;
44
45         public PlayerScore(String playerName, int score) {
46             this.playerName = playerName;
47             this.score = score;
48         }
49
50         public String getPlayerName() {
51             return playerName;
52         }
53
54         public int getScore() {
55             return score;
56         }
57
58         public void setScore(int score) {
59             this.score = score;
60         }
61
62         @Override
63         public String toString() {
64             return playerName + ": " + score;
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

```


4.5. Tratamento de Exceção:

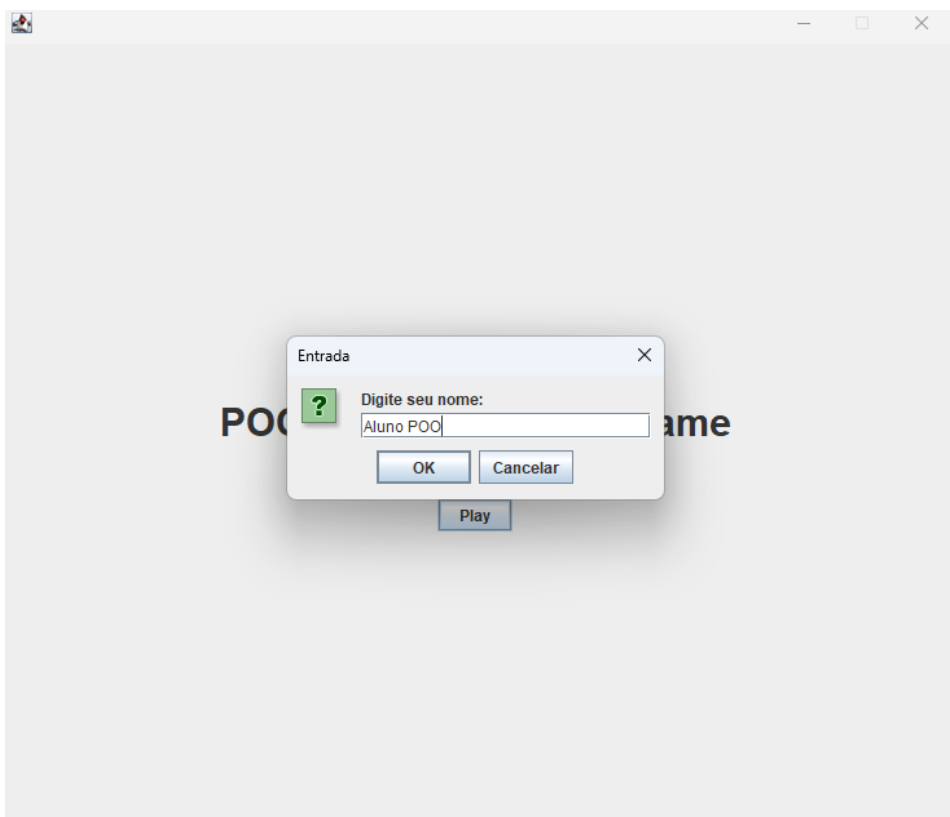
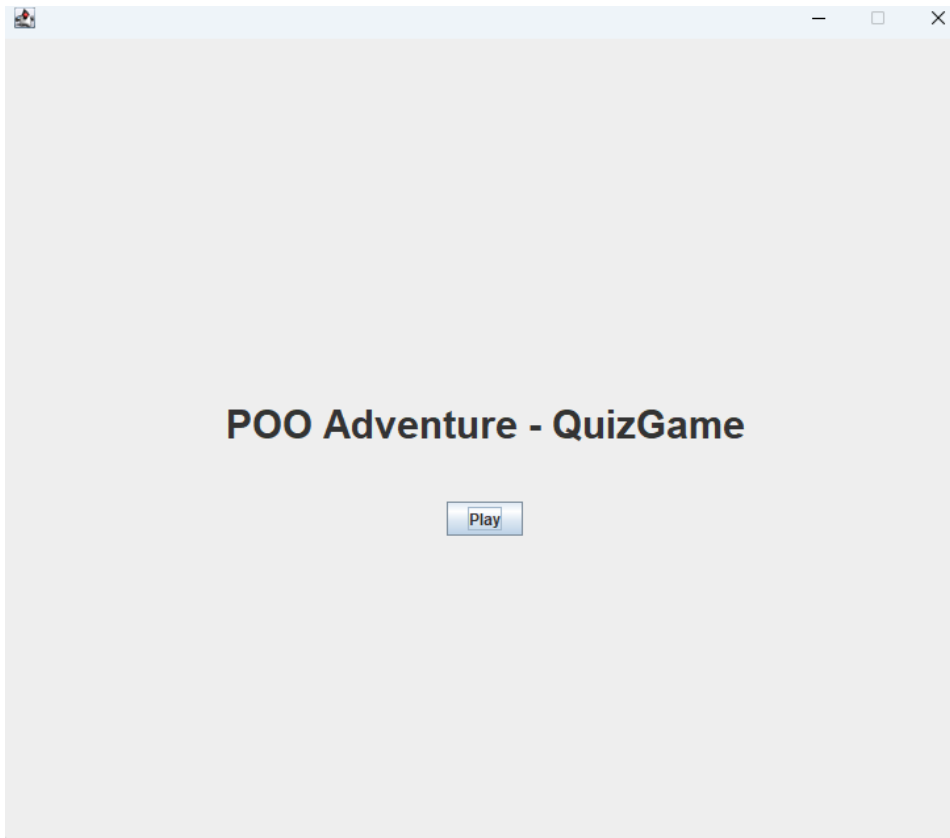
- Há um tratamento de exceção na classe *Quiz* ao salvar a pontuação no arquivo (*salvarPontuacaoNoArquivo*). O *IOException* é tratado usando *try* e *catch*.

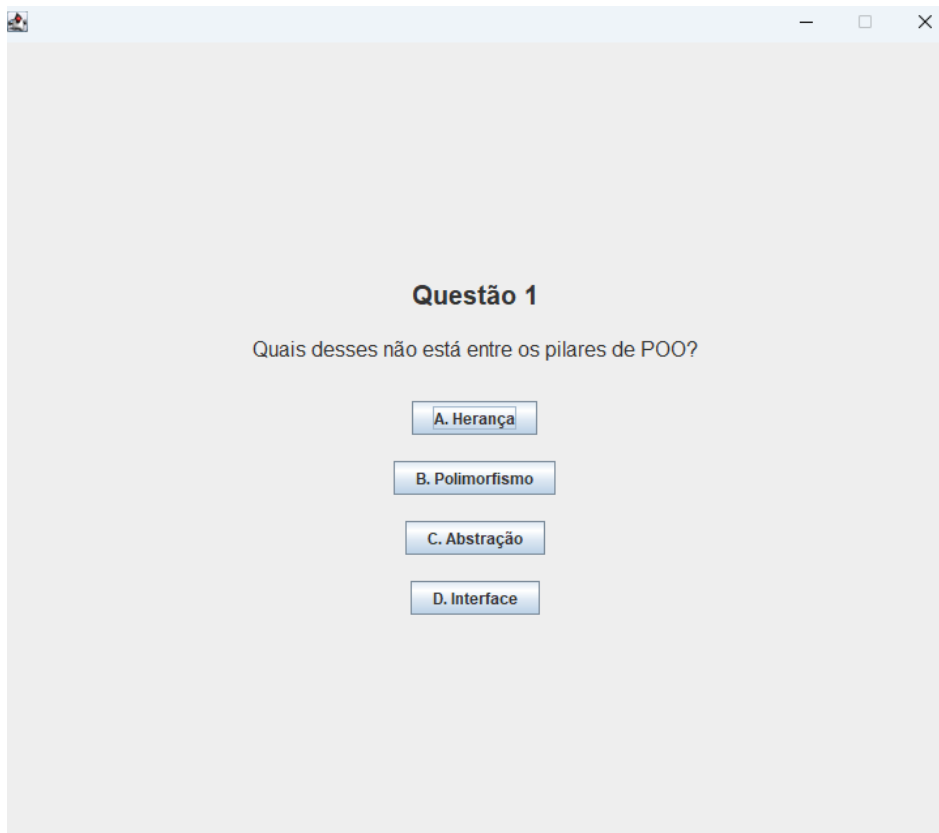
```
239
240 protected void salvarPontuacaoNoArquivo() {
241     try (PrintWriter writer = new PrintWriter(new FileWriter("D:\\Temp\\ws-eclipse\\Quiz\\src\\pontuacao.txt", true))) {
242         writer.println("Nome: " + nomeUsuario + ", Pontuação: " + pontuacao);
243     } catch (IOException e) {
244         e.printStackTrace();
245     }
246 }
247
```

4.6. Interface Gráfica com o Usuário (GUI):

- A interface gráfica está implementada na classe *Quiz*, especialmente no método *criarGUI*. Ela utiliza a biblioteca *Swing* para criar uma interface interativa.

```
112 protected void criarGUI() {
113     frame = new JFrame("");
114     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
115     frame.setSize(750, 650);
116     frame.setResizable(false);
117
118     panel = new JPanel(new GridBagLayout());
119
120     JLabel titleLabel = new JLabel("POO Adventure - QuizGame");
121     titleLabel.setFont(new Font("Roboto", Font.BOLD, 30));
122
123     GridBagConstraints titleConstraints = new GridBagConstraints();
124     titleConstraints.gridx = 0;
125     titleConstraints.gridy = 0;
126     titleConstraints.insets = new Insets(50, 0, 30, 0);
127     panel.add(titleLabel, titleConstraints);
128
129     JButton playButton = new JButton("Play");
130
131     GridBagConstraints buttonConstraints = new GridBagConstraints();
132     buttonConstraints.gridx = 0;
133     buttonConstraints.gridy = 1;
134     buttonConstraints.insets = new Insets(10, 0, 10, 0);
135     panel.add(playButton, buttonConstraints);
136
137     buttonConstraints.gridy = 2;
138
139     frame.add(panel, BorderLayout.CENTER);
140
141     frame.setVisible(true);
142 }
```







Esses critérios estão distribuídos ao longo do código em diferentes partes, conforme mencionado acima. O código demonstra uma estrutura orientada a objetos, encapsulamento, tratamento de exceções e interação com o usuário por meio de uma interface gráfica.

5. Arquitetura do Jogo

A arquitetura do jogo segue o padrão Model-View-Controller (MVC). Abaixo destacamos uma breve descrição de cada parte:

Model (Jogo, Jogador, PlayerScore):

- `Jogo` é a classe abstrata que define a estrutura básica do jogo, contendo métodos abstratos que são implementados em suas subclasses.
- `Quiz` é a implementação concreta do jogo, estendendo a classe abstrata `Jogo` e fornecendo implementações específicas para os métodos abstratos.
- `Jogador` representa um jogador, implementando a interface `Pontuavel` para obter pontuação e nome.
- `PlayerScore` é uma classe interna usada para representar a pontuação do jogador, mantida pelo sistema de placar.

View (Interface Gráfica):

- A interface gráfica do usuário (GUI) é construída usando a biblioteca Swing do Java.
- A janela principal (`JFrame`), painéis (`JPanel`), rótulos (`JLabel`), botões (`JBUTTON`), etc., são utilizados para criar uma interface interativa.

Controller (Event Listeners):

- A lógica de controle é gerenciada pelos event listeners, especialmente no método `actionPerformed`.
- Botões, como "Play" e "Próxima Pergunta", têm listeners que respondem aos eventos do usuário, desencadeando ações correspondentes.

6. Principais Funcionalidades

I. Iniciar Jogo:

- O jogo começa quando o usuário clica no botão "Play".
- O jogador é solicitado a inserir seu nome.

II. Apresentação de Perguntas:

- O jogo apresenta uma série de perguntas sobre programação orientada a objetos (POO).
- Cada pergunta é exibida na interface gráfica.

III. Escolher Respostas:

- O jogador escolhe uma das opções de resposta para cada pergunta.

IV. Pontuação:

- A pontuação é calculada com base nas respostas corretas.
- Cada resposta correta contribui para a pontuação total.

V. Exibição de Resultados:

- Ao final do jogo, a pontuação do jogador é exibida na interface gráfica.
- Uma mensagem informa ao jogador sobre o término do quiz.

VI. Placar de Líderes:

- As pontuações dos jogadores são salvas em um arquivo.
- Uma função de placar de líderes exibe os melhores resultados.

VII. Tratamento de Exceções:

- Exceções relacionadas à entrada do usuário e manipulação de arquivos (IOException) são tratadas.

VIII. Interface Gráfica Interativa:

- A GUI permite uma interação amigável com o usuário, incluindo botões, caixas de diálogo e atualizações dinâmicas.

Essas funcionalidades fornecem uma experiência completa de jogo de Quiz educativo, incentivando os jogadores a testarem seus conhecimentos em programação orientada a objetos.

Conclusão

O desenvolvimento do "POO Adventure - QuizGame" foi concluído com êxito, proporcionando uma experiência educativa para os usuários e desafiadora para a equipe que desenvolveu o jogo.. O projeto demonstra a capacidade de criar jogos interativos em Java, explorando conceitos de POO.