

**QWERTY ESCOLA DE EDUCAÇÃO PROFISSIONAL**

**Técnico em Informática**

**OESLEY RODRIGUES MACHADO**

**CONSTRUÇÃO DO JOGO ICARUS LABYRINTHUS**

**DOM PEDRITO, RS, BRASIL**

**2016**

**QWERTY ESCOLA DE EDUCAÇÃO PROFISSIONAL**

**Técnico em Informática**

**OESLEY RODRIGUES MACHADO**

**CONSTRUÇÃO DO JOGO ICARUS LABYRINTHUS**

Trabalho de Conclusão de Curso apresentado  
como requisito parcial para obtenção do título de  
Técnico em Informática, da Qwerty Escola de  
Educação Profissional.

**DOM PEDRITO, RS, BRASIL**

**2016**

Monografia apresentada como requisito necessário para obtenção título de Bacharel

em Nome do Curso. Qualquer citação atenderá as normas da ética científica.

---

NOME DO ALUNO

Monografia apresentada em \_\_\_\_/\_\_\_\_/\_\_\_\_

---

Orientador (a) Prof.(a). Titulação. Nome do Orientador

---

1º Examinador (a) Prof.(a). Titulação. Nome do Examinador

---

2ª Examinador (a) Prof.(a). Titulação. Nome do Examinador

---

Coordenador (a) Prof.(a). Titulação. Nome do Coordenador

Dedico este trabalho ao meu pai e minha mãe.

## **AGRADECIMENTOS**

Agradeço a Deus por ter conseguido concluir este trabalho, e a todos aqueles que de forma direta ou indireta colaboraram na elaboração do mesmo.

“A melhor maneira de fugir do seu problema é resolvê-lo”

Robert Anthony

## **RESUMO**

Este trabalho tem como objetivo o de explicar o que a Classe Cenário Jogo realiza no jogo Icarus Labyrinthus. Pois o mesmo ao decorrer de seu conteúdo, demonstra como a POO (Programação Orientada a Objetos) facilita na explicação da Classe Cenário Jogo. Como resultado pode-se concluir que de maneira simples é possível compreender o código fonte do jogo Icarus Labyrinthus.

Palavras-chave: Classe. Jogo. Programação.

## **ABSTRACT**

This work aims to explain what the Class Cenário Jogo performs in the game Icarus Labyrinthus. For the same in the course of its content, it demonstrates how the OOP (Object-Oriented Programming) facilitates in the explanation of the Class Cenário Jogo. As a result it can be concluded that in a simple way it is possible to understand the source code of the game Icarus Labyrinthus.

Keywords: Class. Game. Programming.



## LISTA DE FIGURAS

Figura 1: O Tennis for Two.....	11
Figura 2: O SpaceWar!.....	12
Figura 3: Classe Carro .....	14
Figura 4: Método principal da Classe Carro .....	15
Figura 5: Visualização das instruções passadas ao objeto herbie .....	15
Figura 6: Um dos cenários do jogo.....	16
Figura 7: Um dos cenários do jogo com os elementos ponto e saída .....	17
Figura 8: Atributos da Classe Cenário Jogo .....	18
Figura 9: Método carregar .....	20
Figura 10: Método descarregar .....	22
Figura 11: Método atualizar .....	22
Figura 12: Jogador GANHOU.....	23
Figura 13: Jogador PERDEU .....	23
Figura 14: Método desenhar .....	24
Figura 15: Classe Útil .....	27
Figura 16: Classe Texto .....	28
Figura 17: Classe Menu .....	29
Figura 18: Continuação Classe Menu .....	30
Figura 19: Classe Elemento Parte 01 .....	31
Figura 20: Classe Elemento Parte 02.....	32
Figura 21: Classe Cenário Padrão .....	33
Figura 22: Classe Jogo Parte 01 .....	34
Figura 23: Classe Jogo Parte 02 .....	35
Figura 24: Classe Jogo Parte 03 .....	36
Figura 25: Classe Nível Parte 01.....	37
Figura 26: Classe Nível Parte 02.....	38
Figura 27: Classe Cenário Menu Parte 01 .....	39
Figura 28: Classe Cenário Menu Parte 02 .....	40
Figura 29: Classe Cenário Jogo Parte 01 .....	41
Figura 30: Classe Cenário Jogo Parte 02 .....	42
Figura 31: Classe Cenário Jogo Parte 03 .....	43
Figura 32: Classe Cenário Jogo Parte 04 .....	44

## **LISTA DE SIGLAS**

EUA	Estados Unidos da América
MIT	Massachusetts Institute of Technology
POO	Programação Orientada a Objetos

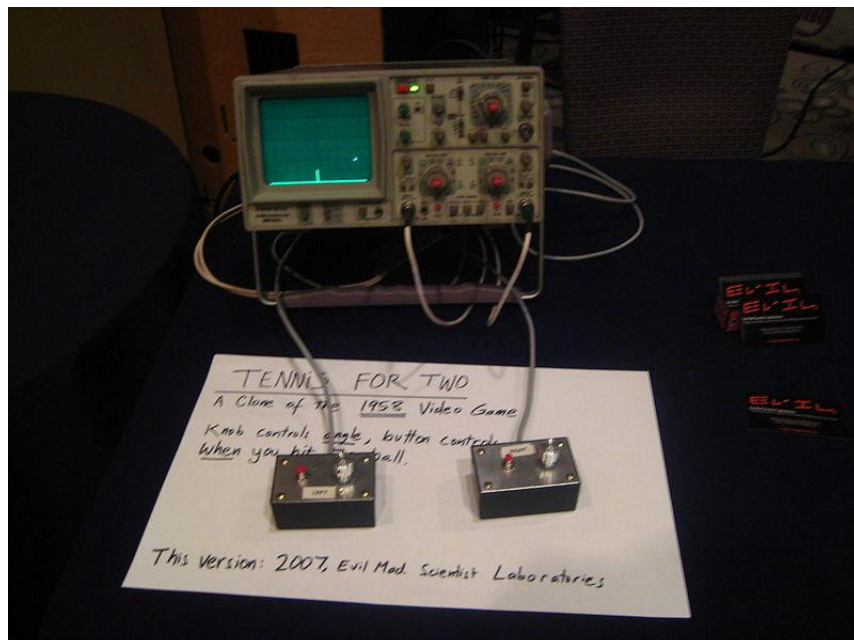
## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>11</b>
1.1 OBJETIVO GERAL .....	12
1.2 OBJETIVOS ESPECIFICOS .....	13
1.3 JUSTIFICATIVA .....	13
1.4 REFERÊNCIAL TEÓRICO .....	13
<b>DESENVOLVIMENTO .....</b>	<b>16</b>
<b>2 DESCRIÇÃO DO JOGO ICARUS LABYRINTHUS.....</b>	<b>16</b>
2.1 DESAFIO DO JOGO .....	16
2.2 REGRAS DO JOGO.....	17
<b>3 CLASSE CENÁRIO JOGO .....</b>	<b>18</b>
3.1 OS ATRIBUTOS.....	18
3.2 O MÉTODO CARREGAR.....	20
3.3 O MÉTODO DESCARREGAR .....	22
3.4 O MÉTODO ATUALIZAR .....	22
3.5 O MÉTODO DESENHAR .....	24
<b>4 CONSIDERAÇÕES FINAIS .....</b>	<b>25</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>26</b>
<b>ANEXO A – CLASSE ÚTIL.....</b>	<b>27</b>
<b>ANEXO B – CLASSE TEXTO.....</b>	<b>28</b>
<b>ANEXO C – CLASSE MENU .....</b>	<b>29</b>
<b>ANEXO D – CLASSE ELEMENTO.....</b>	<b>31</b>
<b>ANEXO E – CLASSE ABSTRATA CENÁRIO PADRÃO .....</b>	<b>33</b>
<b>ANEXO F – CLASSE JOGO COM O MÉTODO <i>MAIN</i>.....</b>	<b>34</b>
<b>APÊNDICE A – CLASSE NIVEL .....</b>	<b>37</b>
<b>APÊNDICE B – CLASSE CENÁRIO MENU.....</b>	<b>39</b>
<b>APÊNDICE C – CLASSE CENÁRIO JOGO .....</b>	<b>41</b>

## 1 INTRODUÇÃO

Em 1958 um vídeo jogo foi criado pelo físico William Higinbotham. Era um jogo de tênis simples, mostrado em um osciloscópio e processado por um computador analógico. Note que o objetivo do programador, ao criar o jogo, tinha sido simplesmente chamar a atenção do público, que visitava as instalações do "Brookhaven National Laboratories", para verificar o poderio nuclear dos EUA. Mais tarde, o cientista aperfeiçoou o jogo, que recebeu o nome de "Tennis Programming", adaptando-o para ser mostrado em um monitor de 15 polegadas. Mas o projeto é conhecido também como "Tennis for Two", que jamais foi patenteado.

Figura 1: O Tennis for Two



FONTE:

[https://pt.wikipedia.org/wiki/Tennis\\_for\\_Two#/media/File:Tennis\\_for\\_Two\\_Machine\\_at\\_CAX\\_2010.jpg](https://pt.wikipedia.org/wiki/Tennis_for_Two#/media/File:Tennis_for_Two_Machine_at_CAX_2010.jpg)

Em 30 de julho 1961, os estudantes do MIT, Stephen Russell, Peter Samson, Dan Edwards, Martin Graetz, Alan Kotok, Steve Piner e Robert A. Saunders, desenvolveram o "SpaceWar!", na linguagem Assembly, inspirados pelos livros de ficção científica de Edward Elmer "Doc" Smith. No jogo, 2 jogadores devem controlar suas naves em um ambiente escuro, e tentar abater o adversário. Diferentemente do "Tennis for Two", o SpaceWar! foi realmente inventado para ser jogado. Assim, é considerado o primeiro jogo interativo de computador, tendo inspirado os futuros vídeo games.

Figura 2: O SpaceWar!



FONTE: <https://pt.wikipedia.org/wiki/Spacewar!#/media/File:Spacewar!-PDP-1-20070512.jpg>

Os jogos em si podem ser programados em diversas linguagens de programação, que seguem diferentes paradigmas que são as suas metodologias e técnicas utilizadas para uma determinada atividade. E um desses paradigmas é a Orientação a Objetos, que atualmente é o mais difundido entre todos. Isso acontece porque se trata de um padrão que tem evoluído muito, principalmente em questões voltadas para segurança e reaproveitamento de código, o que é muito importante no desenvolvimento de qualquer aplicação moderna, e assim sendo, será apresentada a Classe Cenário Jogo do jogo Icarus Labyrinthus, pois a mesma utiliza POO com a linguagem Java.

Java que além de ser uma linguagem de programação é também uma plataforma desenvolvimento. Com ela é possível criar conteúdo para sites, jogos e aplicativos para dispositivos móveis. Uma de suas vantagens é a portabilidade, pois, uma mesma aplicação Java escrita por um sistema A, pode ser utilizada em um sistema B e C sem muitas alterações. Isso acontece porque a máquina virtual Java cria um emulador do sistema operacional para executar suas aplicações. Por isso, programar em Java cria uma vantagem de distribuição de aplicativos, podendo deixar os jogos multiplataforma.

## 1.1 OBJETIVO GERAL

Explicar o que a Classe Cenário Jogo realiza no jogo Icarus Labyrinthus.

## 1.2 OBJETIVOS ESPECIFICOS

Dar uma visão geral do que é POO.

Dar exemplo de classe em Java.

Apresentar o jogo.

Explicar os atributos e o método carregar da Classe Cenário Jogo.

Dar breve explicação do que os métodos descarregar, atualizar e desenhar da Classe Cenário Jogo realizam no jogo.

## 1.3 JUSTIFICATIVA

O mercado de games está mundialmente em ascensão e deve gerar US\$ 99,6 bilhões até o fim do ano de 2016. O valor é 8,5% maior que o mesmo período no ano passado. A estimativa, realizada pela Newzoo, consultoria referência em pesquisas da indústria de games, também prevê movimento positivo no Brasil. No ano passado, a consultoria classificou o País como o 11.º na lista de países com maior mercado de games do mundo.

Em relação à produção nacional, teve-se um aumento de cerca de 15% nos últimos anos e, até pouco tempo atrás, o mercado brasileiro era menos relevante, não se ouvia falar de produções nacionais. Claro, a grande maioria dos jogos ainda é importada, apenas 3% é desenvolvido aqui, então precisa-se de mão-de-obra qualificada, ou seja, de bons profissionais na área e esse é um dos poucos setores que tem estimativa de crescimento em meio à crise.

## 1.4 REFERÊNCIAL TEÓRICO

A Programação Orientada a Objetos é um dos paradigmas da programação, que tem como objetivo o de aproximar o mundo digital do mundo real. Mas como ela faz isso? A POO tem um Molde que cria Objetos, Molde esse que é chamado de Classe que contém os Atributos, Métodos e Estado do Objeto. Porém o que é um Objeto? É uma coisa material ou abstrata que pode ser percebida pelos sentidos e descrita (classificada) por meio das suas Características (Atributos), Comportamentos (Métodos) e Estado Atual (Estado do Objeto). E para termos uma ideia melhor, consideremos a Classe Carro que pode ser descrita da seguinte forma:

Coisas que eu tenho (Atributos)? Cor branca, quatro portas e teto solar. Coisas que eu faço (Métodos)? Ando para frente, ando para trás, dobro para os lados. Como eu estou agora (Estado do Objeto)? Estou andando para frente, agora estou andando para trás. Então antes de se criar um Objeto tem que se definir a Classe.

Em Java a definição de uma classe ficaria assim:

Figura 3: Classe Carro

```

1  package carro;
2  public class Carro {
3      //ATRIBUTOS
4      private String cor;
5      private int qtdPortas;
6      private boolean tetoSolar;
7      //METODOS
8      public void andarParaFrente() {
9          System.out.println("Indo para frente!");
10     }
11     public void andarParaTras() {
12         System.out.println("Indo para trás!");
13     }
14     public void status() {
15         System.out.println("Minha cor: "+this.getCor());
16         System.out.println("Quantas portas: "+this.getQtdPortas());
17         if (tetoSolar) {
18             System.out.println("Teto Solar: Sim");
19         }else{
20             System.out.println("Teto Solar: Não");
21         }
22     }
23     //METODOS ACESSORES E MODIFICADORES
24     public String getCor() {
25         return cor;
26     }
27     public void setCor(String cor) {
28         this.cor = cor;
29     }
30
31     public int getQtdPortas() {
32         return qtdPortas;
33     }
34     public void setQtdPortas(int qtdPortas) {
35         this.qtdPortas = qtdPortas;
36     }
37
38     public boolean isTetoSolar() {
39         return tetoSolar;
40     }
41     public void setTetoSolar(boolean tetoSolar) {
42         this.tetoSolar = tetoSolar;
43     }
44
45     public Carro() { //CONSTRUTOR
46     }
47

```

FONTE: Elaborada pelo autor

Na Figura acima temos a Classe Carro com os seguintes atributos e métodos: atributo cor, atributo qtdPortas e atributo tetoSolar, método andarParaFrente, método andarParaTras e método status. E ao mandar executar a aplicação, a máquina virtual Java, procura pelo bloco **main** (principal) onde todo o conteúdo que estiver dentro do corpo (das chaves) será executado, assim fazendo a aplicação funcionar.

Figura 4: Método principal da Classe Carro

```

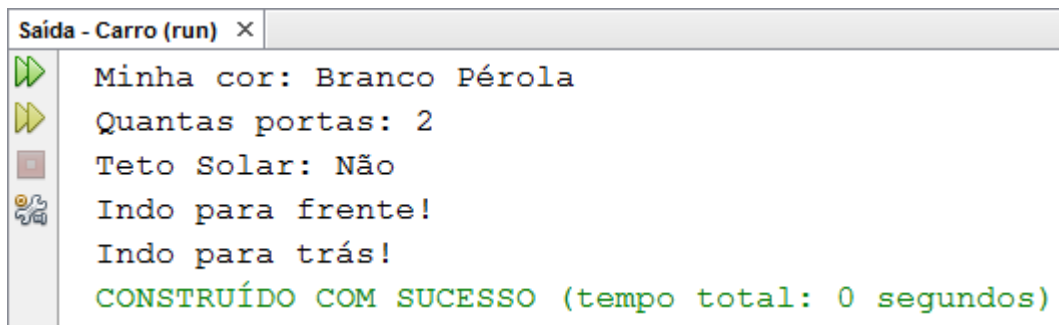
48 public static void main(String[] args) {
49     Carro herbie = new Carro();//AQUI CRIAMOS O OBJETO herbie
50     herbie.setCor("Branco Pérola");//AQUI DEFINIMOS A COR DO herbie QUE SERÁ BRANCO PÉROLA
51     herbie.setQtdPortas(2);//AQUI DEFINIMOS QUE herbie TERÁ 2 PORTAS
52     herbie.setTetoSolar(false);//AQUI DEFINIMOS QUE herbie NÃO TEM TETO SOLAR
53     herbie.status();//AQUI QUEREMOS SABER A SITUAÇÃO DO herbie
54     herbie.andarParaFrente();//AQUI DIZEMOS PARA herbie ANDAR PARA FRENTE
55     herbie.andarParaTras();//AQUI DIZEMOS PARA herbie ANDAR PARA TRÁS
56 }
57
58

```

FONTE: Elaborada pelo autor

E o que acontece ao ser executada a aplicação é o seguinte:

Figura 5: Visualização das instruções passadas ao objeto herbie



```

Saída - Carro (run) X
Minha cor: Branco Pérola
Quantas portas: 2
Teto Solar: Não
Indo para frente!
Indo para trás!
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)

```

FONTE: Elaborada pelo autor



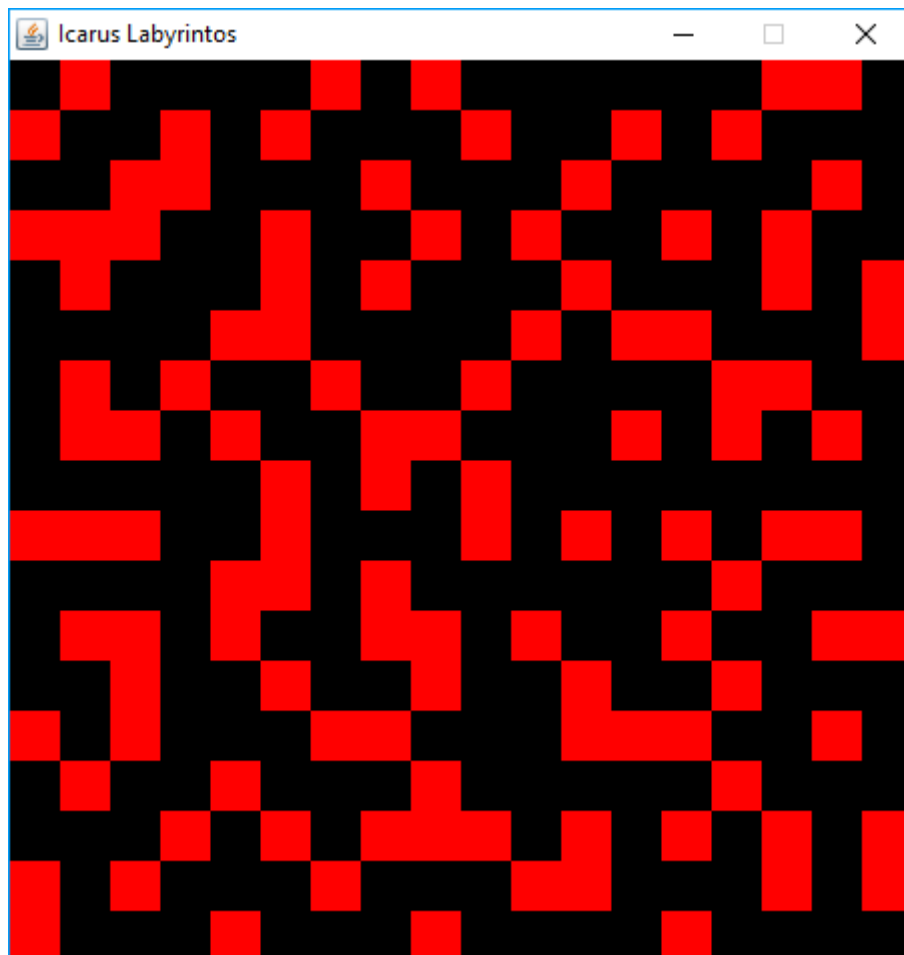
## DESENVOLVIMENTO

### 2 DESCRIÇÃO DO JOGO ICARUS LABYRINTHUS

#### 2.1 DESAFIO DO JOGO

O jogo tem como cenário um labirinto que é constituído por um conjunto de percursos intrincados, criados com a intenção de desorientar o jogador que tenta resolvê-lo. O desafio maior está em levar o ponto azul até o ponto verde que representa a saída do labirinto.

Figura 6: Um dos cenários do jogo

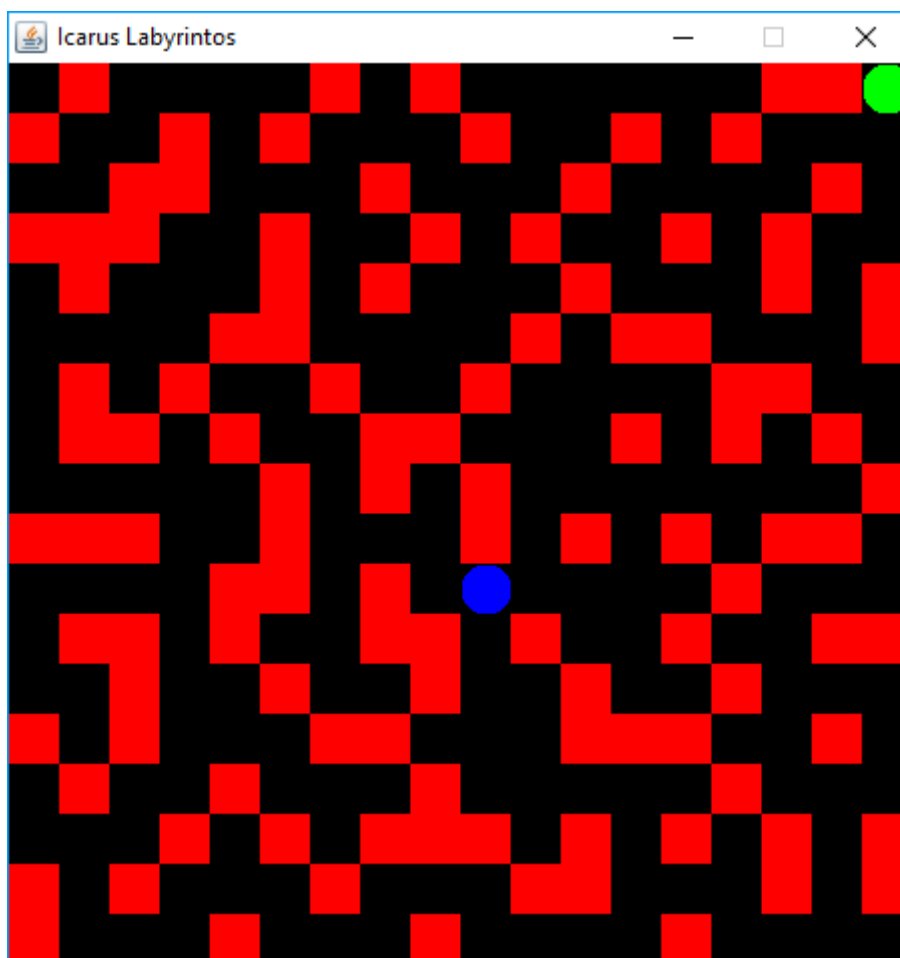


FONTE: Elaborada pelo autor

## 2.2 REGRAS DO JOGO

Como o ponto azul se movimenta constantemente em uma velocidade pré definida, o jogador deve direcioná-lo com as setas do teclado até o ponto verde, entretanto não o podendo deixar encostar nas paredes vermelhas do cenário e nem sair for do mesmo, pois isso acarretará ao game over.

Figura 7: Um dos cenários do jogo com os elementos ponto e saída



FONTE: Elaborada pelo autor

### 3 CLASSE CENÁRIO JOGO

A Classe Cenário Jogo é responsável pela criação e animação dos elementos que o cenário do jogo terá.

#### 3.1 OS ATRIBUTOS

Figura 8: Atributos da Classe Cenário Jogo

```

15 public class CenarioJogo extends CenarioPadrao{
16
17     enum Estado {
18         JOGANDO, GANHOU, PERDEU, INICIO_ANIMACAO, FIM_ANIMACAO
19     }
20
21     private static final int LARG = 25;
22     private int dx, dy;
23     private int temporizador = 0;
24     private Elemento saida;
25     private Elemento ponto;
26     private Elemento[] nivel;
27     private int contadorNivel;
28     private Estado estado = Estado.JOGANDO;
29     private Texto texto = new Texto(new Font("Arial", Font.PLAIN, 25));
30     //ANIMACAO
31     private int duracaoAnimacao = 2;
32     private int iniAnimacao = 0;
33     private int fimAnimacao = Nivel.fases[0].length + duracaoAnimacao;
34
35     public CenarioJogo(int largura, int altura) {
36         super(largura, altura);
37     }

```

FONTE: Elaborada pelo autor

Na Figura acima temos a declaração dos atributos e o método construtor da Classe Cenário Jogo, que se definem da seguinte forma: temos a constante Estado que define o estado do jogo, podendo-o ser JOGANDO, GANHOU, PERDEU, INICIO\_ANIMACAO ou FIM\_ANIMACAO. O atributo LARG é referente à dimensão padrão dos elementos do jogo que será de 25px. Os atributos dx e dy serão responsáveis pela direção do elemento ponto azul, sendo dx direção horizontal e dy direção vertical. O atributo temporizador fica encarregado de fazer a animação do elemento ponto. Os elementos do cenário estão representados pelos nomes de saída (ponto verde), ponto (ponto azul) e nível (paredes vermelhas). O atributo contadorNivel será responsável por receber o valor da quantidade de elementos necessários para construir as paredes do cenário. O atributo que recebe as constantes do jogo é o estado, que de início recebe o valor Estado JOGANDO.

O atributo `texto` define que os textos escritos no cenário do jogo terão sua fonte Arial, normal, tamanho 25. O atributo `iniAnimacao` recebe o valor zero, pois a ele será adicionado o tempo da animação da troca das fases. O atributo `fimAnimacao` recebe o valor 20, pois `Nível.fases[0].length` tem o valor 18, e adicionando o atributo `duracaoAnimacao` que tem o valor 2, chegamos ao valor 20. O método construtor fica responsável em receber os valores da dimensão do cenário, sendo elas largura e altura iguais a 450px.

### 3.2 O MÉTODO CARREGAR

Figura 9: Método carregar

```

39      @Override
40      public void carregar() {
41
42          //define direcao inicial
43          dx = -1;
44          contadorNivel = 0;
45
46          saida = new Elemento(0, 0, LARG, LARG);
47          saida.setAtivo(true);
48          saida.setCor(Color.GREEN);
49
50          ponto = new Elemento(0, 0, LARG, LARG);
51          ponto.setAtivo(true);
52          ponto.setCor(Color.BLUE);
53          ponto.setVel(4);
54
55          char[][] nivelSelecionado = Nivel.fases[Jogo.nivel];
56
57          int total = 0;
58
59          for (int linha = 0; linha < nivelSelecionado.length; linha++) {
60              for (int coluna = 0; coluna < nivelSelecionado[0].length; coluna++) {
61                  if (nivelSelecionado[linha][coluna] == '0') {
62                      total++;
63                  } else if (nivelSelecionado[linha][coluna] == '1') {
64                      ponto.setPx(LARG * coluna);
65                      ponto.setPy(LARG * linha);
66                  } else if (nivelSelecionado[linha][coluna] == '2') {
67                      saida.setPx(LARG * coluna);
68                      saida.setPy(LARG * linha);
69                  }
70              }
71          }
72
73          nivel = new Elemento[total];
74
75          for (int linha = 0; linha < nivelSelecionado.length; linha++) {
76              for (int coluna = 0; coluna < nivelSelecionado[0].length; coluna++) {
77                  if (nivelSelecionado[linha][coluna] == '0') {
78                      Elemento e = new Elemento();
79                      e.setAtivo(true);
80                      e.setCor(Color.RED);
81
82                      e.setPx(LARG * coluna);
83                      e.setPy(LARG * linha);
84
85                      e.setAltura(LARG);
86                      e.setLargura(LARG);
87
88                      nivel[contadorNivel++] = e;
89                  }
90              }
91          }
92      }

```

FONTE: Elaborada pelo autor

Na Figura acima temos o método carregar que é responsável por configurar os elementos do cenário. Assim sendo começamos definindo a direção inicial do ponto azul, que vai ser na direção horizontal (dx) movendo-se para a esquerda (-1).

É adicionado o valor de zero ao contadorNivel para que ele posteriormente receba a adição de outros valores.

A saída recebe um novo elemento de posição horizontal 0 e posição vertical 0, tendo dimensões de 25px de largura e 25px de altura, sendo a saída configurada como um elemento ativo (setAtivo(true)) e também recebe a cor verde (setCor(Color.GREEN)).

O ponto recebe um novo elemento de posição horizontal 0 e posição vertical 0, tendo dimensões de 25px de largura e 25px de altura, sendo o ponto configurado como um elemento ativo (setAtivo(true)), também recebe a cor azul (setCor(Color.BLUE)) e é determinada sua velocidade com o valor 4 (setVel(4)).

Cria-se o atributo bidimensional nivelSelecionado para receber o layout de como deve ser o cenário, isto é, a posição de cada elemento no cenário.

Cria-se o atributo total que ficara responsável por guardar o valor de quantos elementos serão necessários para a construção das paredes vermelha. E onde o nivelSelecionado tiver no layout o valor igual a '0', adiciona-se valores para o atributo total.

É posicionado o elemento ponto (setPx(LARG\*coluna), setPy (LARG\*linha)), a onde o atributo nivelSelecionado tiver no layout o valor igual a '1'.

É posicionado o elemento saída (setPx(LARG\*coluna), setPy (LARG\*linha)), a onde o atributo nivelSelecionado tiver no layout o valor igual a '2'.

Cria-se o atributo nivel que recebe o valor do atributo total, fazendo com nivel receba o valor de quantos elementos paredes serão desenhados no cenário.

Cria-se o elemento e que ficara responsável por ativar cada elemento da parede, atribuir a eles a cor vermelha (setCor(Color.RED)), posicioná-los na posição horizontal e na posição vertical dos mesmos e atribuir a eles as dimensões de 25px de largura e 25px de altura.

### 3.3 O MÉTODO DESCARREGAR

O método descarregar serve para liberarmos espaço de memória, por este motivo dentro do bloco não há nenhuma instrução.

Figura 10: Método descarregar

```

94  @Override
95  public void descarregar() {
96  }
97

```

FONTE: Elaborada pelo autor

### 3.4 O MÉTODO ATUALIZAR

Figura 11: Método atualizar

```

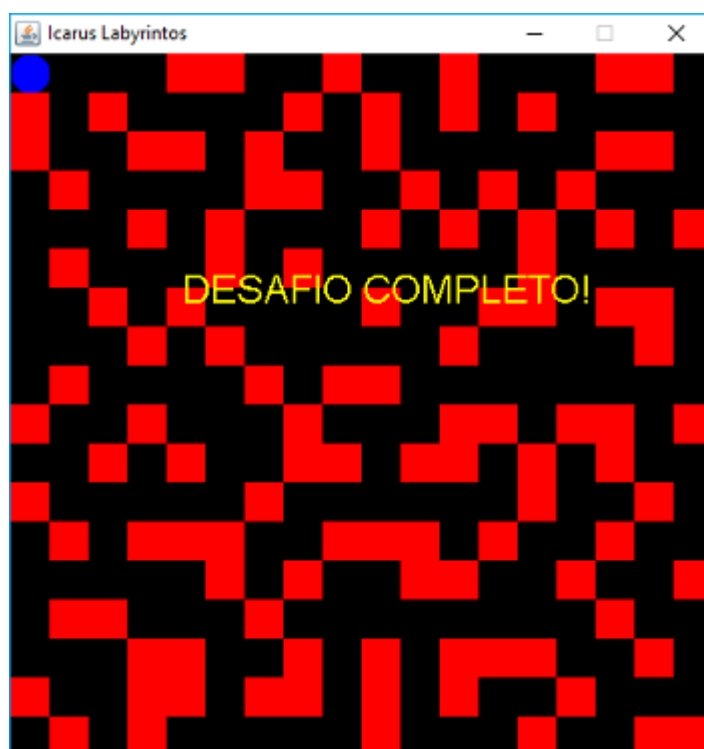
98  @Override
99  public void atualizar() {
100      if (estado == Estado.GANHOU || estado == Estado.PERDEU) {
101          return;
102      }
103      if (estado == Estado.INICIO_ANIMACAO || estado == Estado.FIM_ANIMACAO) {
104          if (estado == Estado.INICIO_ANIMACAO) {
105              iniAnimacao++;
106              if (iniAnimacao == fimAnimacao) {
107                  descarregar();
108                  carregar();
109                  estado = Estado.FIM_ANIMACAO;
110              }
111          } else {
112              iniAnimacao--;
113              if (iniAnimacao == 0) estado = Estado.JOGANDO;
114          }
115          return;
116      }

```

FONTE: Elaborada pelo autor

Na figura acima temos parte do código do método atualizar que é responsável por fazer a verificação do estado do jogo, isto é, qual a animação deve ser feita no jogo. Se for a de quando o jogador está jogando, ou a de se jogador ganhou no caso completou o desafio, ou a de se jogador perdeu, ou se é hora de fazer a animação de troca de nível, isto é, se o jogador passou de fase.

Figura 12: Jogador GANHOU



FONTE: Elaborada pelo autor

Figura 13: Jogador PERDEU



FONTE: Elaborada pelo autor



### 3.5 O MÉTODO DESENHAR

Figura 14: Método desenhar

```

171
172 @Override
173 public void desenhar(Graphics2D g) {
174     for (Elemento e : nivel) {
175         if (e == null) break;
176         e.desenhaRect(g);
177     }
178     saida.desenhaOval(g);
179     ponto.desenhaOval(g);
180
181     if (estado == Estado.INICIO_ANIMACAO || estado == Estado.FIM_ANIMACAO) {
182         char[][] nivelSelecioneado = Nivel.fases[Jogo.nivel];
183         int limite = 0;
184         g.setColor(Color.LIGHT_GRAY);
185         for (int linha = 0; linha < nivelSelecioneado.length; linha++) {
186             limite++;
187             if (limite >= iniAnimacao) break;
188             for (int coluna = 0; coluna < nivelSelecioneado[0].length; coluna++) {
189                 g.fillRect(LARG * coluna, LARG * linha, LARG, LARG);
190             }
191         }
192     } else if (estado == Estado.GANHOU || estado == Estado.PERDEU) {
193         if (estado == Estado.GANHOU) {
194             g.setColor(Color.YELLOW);
195             texto.desenha(g, "DESAFIO COMPLETO!", 110, 160);
196         } else {
197             g.setColor(Color.BLACK);
198             g.fillRect(ponto.getPx() + LARG * 2, ponto.getPy() + LARG * 2, 450, 450);
199             g.fillRect(ponto.getPx() - 450 - LARG, ponto.getPy() - 450 - LARG, 450, 450);
200             g.fillRect(ponto.getPx() + LARG * 2, ponto.getPy() - 450 - LARG, 450, 450);
201             g.fillRect(ponto.getPx() - 450 - LARG, ponto.getPy() + LARG * 2, 450, 450);
202             g.setColor(Color.YELLOW);
203             texto.desenha(g, "GAME OVER!", 160, 160);
204             texto.desenha(g, "PARA REINICIAR APERTE ENTER!", 20, 420);
205         }
206     }
207 }
208 }
209

```

FONTE: Elaborada pelo autor

Na Figura acima temos o método desenhar que fica responsável por desenhar os elementos definidos no método carregar, e de fazer a animação que é definida no método atualizar.

## **4 CONSIDERAÇÕES FINAIS**

Com conhecimentos obtidos durante o curso, conclui que é possível desenvolver aplicações para várias áreas, não somente as apresentadas durante o curso, como entre elas a de jogos. Possibilitando assim uma oportunidade maior no mercado de trabalho, como por exemplo, o desenvolvimento de jogos Indie (independentes). No presente momento este trabalho não teve o intuito de apresentar o desenvolvimento completo da aplicação, pois ele trata de dar uma demonstração de como programar Orientado a Objetos facilita na elaboração da aplicação.

## REFERÊNCIAS BIBLIOGRÁFICAS

Becker, Marcus. A lógica do jogo: recriando clássicos da história dos videogames. São Paulo: Casa do Código, 2016.

Cantelli, Geraldo Cesar. Java: uma abordagem sobre programação Java. Santa Cruz do Rio Pardo: EDITORA VIENA, 2014.

Gasparotto, Henrique Machado. Os 4 pilares da Programação Orientada a Objetos. 2014. Disponível em: <<http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>>. Acesso em 30 de Novembro de 2016.

Guanabara, Gustavo. Curso de POO Java (Programação Orientada a Objetos). 2016. Disponível em: <[https://www.youtube.com/playlist?list=PLHz\\_AreHm4dkqe2aR0tQK74m8SFe-aGsY](https://www.youtube.com/playlist?list=PLHz_AreHm4dkqe2aR0tQK74m8SFe-aGsY)>. Acesso em 01 de Agosto de 2016.

História: Primeiros jogos digitais. 2015. <<http://www.ufpa.br/dicas/net1/int-h-jo.htm>> Acesso em 30 de Novembro de 2016.

Lima, Mariana. Mercado de games avança no Brasil. 2016. <<http://www.inova.jor.br/2016/06/01/mercado-games-brasil/>>. Acesso em 30 de Novembro de 2016.

Peres, Helder Henrique do Nascimento. Java para games: o tutorial para sua grande aventura. Santa Cruz do Rio Pardo: EDITORA VIENA, 2015.

## ANEXO A – CLASSE ÚTIL

Figura 15: Classe Útil

```

1  package br.com.orm.icaruslabyrinthus.ferramentas;
2
3  /**
4   *
5   * @author orm
6   */
7  public class Util {
8      public static boolean colide(Elemento a, Elemento b){
9          if(!a.isAtivo() || !b.isAtivo()) return false;
10         //posição no eixo X + largura do elemento A e B
11         final int plA = a.getPx() + a.getLargura();
12         final int plB = b.getPx() + b.getLargura();
13         //posição no eixo Y + altura do elemento A e B
14         final int paA = a.getPy() + a.getAltura();
15         final int paB = b.getPy() + b.getAltura();
16
17         if(plA > b.getPx() && a.getPx() < plB &&
18            paA > b.getPy() && a.getPy() < paB){
19             return true;
20         }
21         return false;
22     }
23
24     public static boolean colideX(Elemento a, Elemento b){
25         if (!a.isAtivo() || !b.isAtivo()) return false;
26
27         if (a.getPx() + a.getLargura() >= b.getPx() &&
28            a.getPx() <= b.getPx() + b.getLargura()) {
29             return true;
30         }
31
32         return false;
33     }
34
35     public static void centraliza(Elemento el, int larg, int alt){
36         if (alt > 0) {
37             el.setPy(alt / 2 - el.getAltura() / 2);
38         }
39         if (larg > 0) {
40             el.setPx(larg / 2 - el.getLargura() / 2);
41         }
42     }
43
44     public static boolean saiu(Elemento e, int largura, int altura){
45         if (e.getPx() < 0 || e.getPx() + e.getLargura() > largura) {
46             return true;
47         }
48         if (e.getPy() < 0 || e.getPy() + e.getAltura() > altura) {
49             return true;
50         }
51         return false;
52     }
53
54 }
55

```

FONTE: Elaborada pelo autor

## ANEXO B – CLASSE TEXTO

Figura 16: Classe Texto

```

1  package br.com.orm.icaruslabyrinthus.ferramentas;
2
3  import java.awt.Font;
4  import java.awt.Graphics2D;
5
6  /**
7   *
8   * @author orm
9   */
10 public class Texto extends Elemento {
11     private Font fonte;
12
13     public Texto() {
14         fonte = new Font("Tahoma", Font.PLAIN, 16);
15     }
16
17     public Texto(Font fonte) {
18         this.fonte = fonte;
19     }
20
21     public void desenha(Graphics2D g, String texto) {
22         desenha(g, texto, getPx(), getPy());
23     }
24
25     public void desenha(Graphics2D g, String texto, int px, int py) {
26         if (getCor() != null) {
27             g.setColor(getCor());
28         }
29
30         g.setFont(fonte);
31         g.drawString(texto, px, py);
32     }
33
34     public Font getFonte() {
35         return fonte;
36     }
37
38     public void setFonte(Font fonte) {
39         this.fonte = fonte;
40     }
41
42 }
43

```

FONTE: Elaborada pelo autor

## ANEXO C – CLASSE MENU

Figura 17: Classe Menu

```

1  package br.com.orm.icaruslabyrinthus.ferramentas;
2
3  import java.awt.Color;
4  import java.awt.Graphics2D;
5
6  /**
7   *
8   * @author orm
9   */
10 public class Menu extends Texto {
11     private short idx;
12     private String rotulo;
13     private String[] opcoes;
14     private boolean selecionado;
15
16     public Menu(String rotulo) {
17         super();
18         this.rotulo = rotulo;
19         setLargura(120);
20         setAltura(20);
21         setCor(Color.WHITE);
22     }
23
24     public void addOpcoes(String... opcao) {
25         opcoes = opcao;
26     }
27
28     public void desenha(Graphics2D g) {
29         if (opcoes == null) return;
30
31         g.setColor(getCor());
32         super.desenha(g, getRotulo()+"<"+opcoes[idx]+">",
33             getPx(), getPy() + getAltura());
34
35         if (selecionado) {
36             g.drawLine(getPx(), getPy() + getAltura() + 5,
37                 getPx() + getLargura(), getPy() + getAltura() + 5);
38         }
39     }
40
41     public String getRotulo() {
42         return rotulo;
43     }
44
45     public void setRotulo(String rotulo) {
46         this.rotulo = rotulo;
47     }
48
49     public boolean isSelecionado() {
50         return selecionado;
51     }
52
53     public void setSelecionado(boolean selecionado) {
54         this.selecionado = selecionado;
55     }
56
57     public int getOpcaoId() {
58         return idx;
59     }
60

```

FONTE: Elaborada pelo autor

Figura 18: Continuação Classe Menu

```
61 public String getOpcaoTexto() {  
62     return opcoes[idx];  
63 }  
64  
65 public void setTrocaOpcao(boolean esquerda) {  
66     if(!isSelecionado() || !isAtivo()) return;  
67  
68     idx += esquerda? -1: 1;  
69  
70     if (idx < 0) {  
71         idx = (short) (opcoes.length - 1);  
72     }else if(idx == opcoes.length){  
73         idx = 0;  
74     }  
75 }  
76  
77 }  
78
```

FONTE: Elaborada pelo autor

## ANEXO D – CLASSE ELEMENTO

Figura 19: Classe Elemento Parte 01

```

1  package br.com.orm.icaruslabyrinthus.ferramentas;
2
3  import java.awt.Color;
4  import java.awt.Graphics2D;
5
6  /**
7   *
8   * @author orm
9   */
10 public class Elemento {
11     private int px, py, largura, altura, vel;
12     private boolean ativo;
13     private Color cor;
14
15     public Elemento() {}
16
17     public Elemento(int px, int py, int largura, int altura) {
18         this.px = px;
19         this.py = py;
20         this.largura = largura;
21         this.altura = altura;
22     }
23
24     public void atualiza() {}
25
26     public void desenhaRect(Graphics2D g) {
27         g.setColor(cor);
28         //g.drawRect(px, py, largura, altura);
29         g.fillRect(px, py, largura, altura);
30     }
31
32     public void desenhaOval(Graphics2D g) {
33         g.setColor(cor);
34         //g.drawOval(px, py, largura, altura);
35         g.fillOval(px, py, largura, altura);
36     }
37
38     public int getLargura() {
39         return largura;
40     }
41     public void setLargura(int largura) {
42         this.largura = largura;
43     }
44
45     public int getAltura() {
46         return altura;
47     }
48     public void setAltura(int altura) {
49         this.altura = altura;
50     }
51
52     public int getPx() {
53         return px;
54     }
55     public void setPx(int px) {
56         this.px = px;
57     }
58
59     public int getPy() {
60         return py;
61     }

```

FONTE: Elaborada pelo autor



Figura 20: Classe Elemento Parte 02

```

62  public void setPy(int py) {
63      this.py = py;
64  }
65
66  public int getVel() {
67      return vel;
68  }
69  public void setVel(int vel) {
70      this.vel = vel;
71  }
72
73  public boolean isAtivo() {
74      return ativo;
75  }
76  public void setAtivo(boolean ativo) {
77      this.ativo = ativo;
78  }
79
80  public Color getCor() {
81      return cor;
82  }
83  public void setCor(Color cor) {
84      this.cor = cor;
85  }
86
87  public void incPx(int x) {
88      px = px + x;
89  }
90  public void incPy(int y) {
91      py = py + y;
92  }
93
94  @Override
95  public String toString(){
96      return "Elemento [px=" + px + ", py=" + py + "]";
97  }
98
99  }

```

FONTE: Elaborada pelo autor

## ANEXO E – CLASSE ABSTRATA CENÁRIO PADRÃO

Figura 21: Classe Cenário Padrão

```
1 package br.com.orm.icaruslabyrinthus.ferramentas;
2
3 import java.awt.Graphics2D;
4
5 /**
6  *
7  * @author orm
8  */
9 public abstract class CenarioPadrao {
10     protected int altura;
11     protected int largura;
12
13     public CenarioPadrao(int largura, int altura) {
14         this.altura = altura;
15         this.largura = largura;
16     }
17
18     public abstract void carregar();
19
20     public abstract void descarregar();
21
22     public abstract void atualizar();
23
24     public abstract void desenhar(Graphics2D g);
25 }
26
```

FONTE: Elaborada pelo autor

## ANEXO F – CLASSE JOGO COM O MÉTODO *main*

Figura 22: Classe Jogo Parte 01

```

1  package br.com.orm.icaruslabyrinthus;
2
3  import br.com.orm.icaruslabyrinthus.ferramentas.CenarioPadrao;
4  import java.awt.Color;
5  import java.awt.Dimension;
6  import java.awt.Graphics;
7  import java.awt.Graphics2D;
8  import java.awt.event.KeyEvent;
9  import java.awt.event.KeyListener;
10 import java.awt.image.BufferedImage;
11 import javax.swing.JFrame;
12 import javax.swing.JPanel;
13
14 /**
15  *
16  * @author orm
17  */
18 public class Jogo extends JFrame{
19     private static final long serialVersionUID = 1L; // IDENTIFICAÇÃO EX.: VERSÃO 1.0
20
21     private static final int FPS = 1000 / 20; // 50ms == 20 QUADROS POR SEGUNDO
22
23     private static final int JANELA_LARGURA = 450;
24     private static final int JANELA_ALTURA = 450;
25
26     private JPanel tela;
27     private Graphics2D g2d;
28     private BufferedImage buffer;
29
30     private CenarioPadrao cenario;
31
32     public enum Tecla{
33         CIMA, BAIXO, ESQUERDA, DIREITA, ESC, ENTER
34     }
35
36     public static boolean[] controleTecla = new boolean[Tecla.values().length];
37     public static void liberarTeclas(){
38         for (int i = 0; i < controleTecla.length; i++) {
39             controleTecla[i] = false;
40         }
41     }
42     private void setaTecla(int tecla, boolean pressionada){
43         switch(tecla){
44             case KeyEvent.VK_UP: // case KeyEvent.VK_W:
45                 controleTecla[Tecla.CIMA.ordinal()] = pressionada;
46                 break;
47             case KeyEvent.VK_DOWN: // case KeyEvent.VK_S:
48                 controleTecla[Tecla.BAIXO.ordinal()] = pressionada;
49                 break;
50             case KeyEvent.VK_LEFT: // case KeyEvent.VK_A:
51                 controleTecla[Tecla.ESQUERDA.ordinal()] = pressionada;
52                 break;
53             case KeyEvent.VK_RIGHT: // case KeyEvent.VK_D:
54                 controleTecla[Tecla.DIREITA.ordinal()] = pressionada;
55                 break;
56             case KeyEvent.VK_ESCAPE:
57                 controleTecla[Tecla.ESC.ordinal()] = pressionada;
58                 break;
59             case KeyEvent.VK_ENTER: // case KeyEvent.VK_SPACE:
60                 controleTecla[Tecla.ENTER.ordinal()] = pressionada;
61                 break;
62             } // FIM switch
63     } // FIM setaTecla

```

FONTE: Elaborada pelo autor

Figura 23: Classe Jogo Parte 02

```

64
65 public static int nivel;
66 public static int velocidade;
67
68 public Jogo() { //METODO CONSTRUTOR
69     this.addKeyListener(new KeyListener() {
70         @Override //EVENTO PARA TECLA APERTADA
71         public void keyTyped(KeyEvent ke) {
72         }
73
74         @Override //EVENTO PARA TECLA PRESSIONADA
75         public void keyPressed(KeyEvent ke) {
76             setaTecla(ke.getKeyCode(), true);
77         }
78
79         @Override //EVENTO PARA TECLA LIBERADA
80         public void keyReleased(KeyEvent ke) {
81             setaTecla(ke.getKeyCode(), false);
82         }
83     });
84
85     buffer = new BufferedImage(JANELA_LARGURA, JANELA_ALTURA,
86         BufferedImage.TYPE_INT_RGB);
87
88     g2d = buffer.createGraphics(); //OBTENÇÃO DO OBJETO Graphics2D
89
90     tela = new JPanel() {
91         private static final long serialVersionUID = 1L;
92
93         //O METODO paintComponent(Graphics g) FAZ A PINTURA NO JPanel
94         @Override
95         public void paintComponent(Graphics g) {
96             g.drawImage(buffer, 0, 0, null);
97         }
98
99         @Override //NAO PODE PERDE ESPACO PARA AS BORDAS
100        public Dimension getPreferredSize() {
101            return new Dimension(JANELA_LARGURA, JANELA_ALTURA);
102        }
103
104        @Override //AQUI O TAMANHO MINIMO tela
105        public Dimension getMinimumSize() {
106            return getPreferredSize();
107        }
108    };
109
110    getContentPane().add(tela); //ADICIONANDO A TELA NA MOLDURA
111    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //FECHAR JANELA
112    setTitle("Icarus Labirintos"); //TITULO DA JANELA
113    setVisible(true); //MOSTRAR JANELA
114    setResizable(false); //NAO DEIXA A JANELA SER REDIMENSIONADA
115    setLocation(458, 159); //LOCALIZAÇÃO DA JANELA A SER GERADA
116    pack(); //ESTE METODO DIZ A JANELA QUE O ESPACO QUE SE PRECISA É DE ACORDO
117    AOS TAMANHOS DOS ELEMENTOS QUE ESTAO NELA*/
118    tela.repaint(); //FAZ OS COMPONENTES DE tela SE REPINTAREM EM ms
119 }
120
121 private void carregarJogo() {
122     cenario = new CenarioMenu(tela.getWidth(), tela.getHeight());
123     cenario.carregar();
124 }
125
126 public void iniciarJogo() {
127     long prxAtualizacao = 0;
128
129     while(true) {

```

FONTE: Elaborada pelo autor

Figura 24: Classe Jogo Parte 03

```

130         if (System.currentTimeMillis() >= prxAtualizacao) {
131             g2d.setColor(Color.BLACK);
132             g2d.fillRect(0, 0, JANELA_LARGURA, JANELA_ALTURA);
133             if (controleTecla[Tecla.ENTER.ordinal()]) {
134                 //PRESSIONOU ENTER
135                 if (cenario instanceof CenarioMenu) {
136                     cenario Descarregar();
137                     cenario = null;
138
139                     if (Jogo.nivel < Nivel.fases.length) { //PASSANDO DE FASE
140                         cenario = new CenarioJogo(tela.getWidth(), tela.getHeight());
141                     }
142
143                     cenario.carregar();
144
145                 }else{
146                     //Jogo.pausado = !Jogo.pausado;
147                     cenario Descarregar();
148                     //cenario = null;
149                     cenario = new CenarioJogo(tela.getWidth(), tela.getHeight());
150                     cenario.carregar();
151                 }
152
153                 liberarTeclas();
154
155             }else if (controleTecla[Tecla.ESC.ordinal()]) {
156                 //PRESSIONOU ESC
157                 if (!(cenario instanceof CenarioMenu)) {
158                     cenario Descarregar();
159                     cenario = null;
160                     cenario = new CenarioMenu(tela.getWidth(), tela.getHeight());
161                     cenario.carregar();
162                 }
163
164                 liberarTeclas();
165
166             }
167
168             if (cenario == null) {
169                 g2d.setColor(Color.WHITE);
170                 g2d.drawString("Carregando...", 20, 20);
171             }else{
172                 //if (!Jogo.pausado) {
173                     cenario.atualizar();
174                 //}
175                 cenario.desenhar(g2d);
176             }
177
178             tela.repaint();
179             prxAtualizacao = System.currentTimeMillis() + FPS;
180         } //FIM if(System.currentTimeMillis() >= prxAtualizacao)
181     } //FIM while
182 } //FIM METODO iniciarJogo
183
184 public static void main(String[] args){
185     Jogo jogoNovo = new Jogo();
186     jogoNovo.carregarJogo();
187     jogoNovo.iniciarJogo();
188 }
189
190 }
191

```

FONTE: Elaborada pelo autor

## APÊNDICE A – CLASSE NIVEL

Figura 25: Classe Nível Parte 01

[illegible]

FONTE: Elaborada pelo autor

Figura 26: Classe Nível Parte 02

```

52      { //NÍVEL 2
53      //COLUNA{'1','02','03','04','05','06','07','08','09','10','11','12','13','14','15','16','17','18'},
54      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 01
55      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 02
56      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 03
57      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 04
58      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 05
59      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 06
60      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 07
61      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 08
62      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 09
63      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 10
64      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 11
65      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 12
66      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 13
67      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 14
68      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 15
69      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 16
70      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 17
71      {'2','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 18
72      },
73      { //NÍVEL 3
74      //COLUNA{'1','02','03','04','05','06','07','08','09','10','11','12','13','14','15','16','17','18'},
75      {'2','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 1
76      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 2
77      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 3
78      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 4
79      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 5
80      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 6
81      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 7
82      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 8
83      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 9
84      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 10
85      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 11
86      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 12
87      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 13
88      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 14
89      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 15
90      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 16
91      {'0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 17
92      {'1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'},//LINHA 18
93      }
94      };
95  }
96

```

FONTE: Elaborada pelo autor

## APÊNDICE B – CLASSE CENÁRIO MENU

Figura 27: Classe Cenário Menu Parte 01

```

1  package br.com.orm.icaruslabyrinthus;
2
3  import br.com.orm.icaruslabyrinthus.ferramentas.CenarioPadrao;
4  import br.com.orm.icaruslabyrinthus.ferramentas.Menu;
5  import br.com.orm.icaruslabyrinthus.ferramentas.Util;
6  import java.awt.Color;
7  import java.awt.Graphics2D;
8
9  /**
10   *
11   * @author orm
12   */
13  public class CenarioMenu extends CenarioPadrao{
14
15      public CenarioMenu(int largura, int altura) {
16          super(largura, altura);
17      }
18
19      private Menu jogar;
20      private Menu instrucao;
21
22      @Override
23      public void carregar() {
24          jogar = new Menu("");
25          instrucao = new Menu("");
26
27          jogar.addOpcoes("      JOGAR      ");
28          instrucao.addOpcoes("INSTRUÇÕES");
29
30          Util.centraliza(jogar, largura, altura);
31          Util.centraliza(instrucao, largura, altura);
32
33          instrucao.setPy(jogar.getPy() + jogar.getAltura());
34
35          jogar.setAtivo(true);
36          jogar.setSelecionado(true);
37          instrucao.setAtivo(true);
38      }
39
40
41      @Override
42      public void descarregar() {
43      }
44
45      @Override
46      public void atualizar() {
47          if (Jogo.controleTecla[Jogo.Tecla.CIMA.ordinal()] ||
48              Jogo.controleTecla[Jogo.Tecla.BAIXO.ordinal()]) {
49              if (jogar.isSelecionado()) {
50                  jogar.setSelecionado(false);
51                  instrucao.setSelecionado(true);
52              } else {
53                  jogar.setSelecionado(true);
54                  instrucao.setSelecionado(false);
55              }
56          } else if (Jogo.controleTecla[Jogo.Tecla.ESQUERDA.ordinal()] ||
57                      Jogo.controleTecla[Jogo.Tecla.DIREITA.ordinal()]) {
58              jogar.setTrocaOpcao(Jogo.controleTecla[Jogo.Tecla.ESQUERDA.ordinal()]);
59          }
60          Jogo.liberarTeclas();
61      }

```

FONTE: Elaborada pelo autor



Figura 28: Classe Cenário Menu Parte 02

```
62
63
64 ①
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79

@Override
public void desenhar(Graphics2D g) {
    jogar.desenha(g);
    instrucao.desenha(g);
    if (instrucao.isSelecioneado()) {
        g.setColor(Color.YELLOW);
        g.drawString("Usando as setas do teclado", 125,
            instrucao.getPy() + 3*instrucao.getAltura());
        g.drawString("fazer o ponto azul", 160,
            instrucao.getPy() + 4*instrucao.getAltura());
        g.drawString("chegar ao ponto verde.", 140,
            instrucao.getPy() + 5*instrucao.getAltura());
    }
}
```

FONTE: Elaborada pelo autor

## APÊNDICE C – CLASSE CENÁRIO JOGO

Figura 29: Classe Cenário Jogo Parte 01

```

1  package br.com.orm.icaruslabyrinthus;
2
3  import br.com.orm.icaruslabyrinthus.ferramentas.CenarioPadrao;
4  import br.com.orm.icaruslabyrinthus.ferramentas.Elemento;
5  import br.com.orm.icaruslabyrinthus.ferramentas.Texto;
6  import br.com.orm.icaruslabyrinthus.ferramentas.Util;
7  import java.awt.Color;
8  import java.awt.Font;
9  import java.awt.Graphics2D;
10
11  /**
12   *
13   * @author orm
14   */
15  public class CenarioJogo extends CenarioPadrao{
16
17      enum Estado {
18          JOGANDO, GANHOU, PERDEU, INICIO_ANIMACAO, FIM_ANIMACAO
19      }
20
21      private static final int LARG = 25;
22      private int dx, dy;
23      private int temporizador = 0;
24      private Elemento saida;
25      private Elemento ponto;
26      private Elemento[] nivel;
27      private int contadorNivel;
28      private Estado estado = Estado.JOGANDO;
29      private Texto texto = new Texto(new Font("Arial", Font.PLAIN, 25));
30      //ANIMACAO
31      private int duracaoAnimacao = 2;
32      private int iniAnimacao = 0;
33      private int fimAnimacao = Nivel.fases[0].length + duracaoAnimacao;
34
35      public CenarioJogo(int largura, int altura) {
36          super(largura, altura);
37      }
38
39      @Override
40      public void carregar() {
41          System.out.println("carregar"+fimAnimacao);
42          //define direcao inicial
43          dx = -1;
44          contadorNivel = 0;
45
46          saida = new Elemento(0, 0, LARG, LARG);
47          saida.setAtivo(true);
48          saida.setCor(Color.GREEN);
49
50          ponto = new Elemento(0, 0, LARG, LARG);
51          ponto.setAtivo(true);
52          ponto.setCor(Color.BLUE);
53          ponto.setVel(4);
54
55          char[][] nivelSelecionado = Nivel.fases[Jogo.nivel];
56
57          int total = 0;
58

```

FONTE: Elaborada pelo autor

Figura 30: Classe Cenário Jogo Parte 02

```

59     for (int linha = 0; linha < nivelSelecioneado.length; linha++) {
60         for (int coluna = 0; coluna < nivelSelecioneado[0].length; coluna++) {
61             if (nivelSelecioneado[linha][coluna] == '0') {
62                 total++;
63             } else if (nivelSelecioneado[linha][coluna] == '1') {
64                 ponto.setPx(LARG * coluna);
65                 ponto.setPy(LARG * linha);
66             } else if (nivelSelecioneado[linha][coluna] == '2') {
67                 saida.setPx(LARG * coluna);
68                 saida.setPy(LARG * linha);
69             }
70         }
71     }
72
73     nivel = new Elemento[total];
74
75     for (int linha = 0; linha < nivelSelecioneado.length; linha++) {
76         for (int coluna = 0; coluna < nivelSelecioneado[0].length; coluna++) {
77             if (nivelSelecioneado[linha][coluna] == '0') {
78                 Elemento e = new Elemento();
79                 e.setAtivo(true);
80                 e.setCor(Color.RED);
81
82                 e.setPx(LARG * coluna);
83                 e.setPy(LARG * linha);
84
85                 e.setAltura(LARG);
86                 e.setLargura(LARG);
87
88                 nivel[contadorNivel++] = e;
89             }
90         }
91     }
92 }
93
94 @Override
95 public void descarregar() {
96 }
97
98 @Override
99 public void atualizar() {
100     if (estado == Estado.GANHOU || estado == Estado.PERDEU) {
101         return;
102     }
103     if (estado == Estado.INICIO_ANIMACAO || estado == Estado.FIM_ANIMACAO) {
104         if (estado == Estado.INICIO_ANIMACAO) {
105             iniAnimacao++;
106             if (iniAnimacao == fimAnimacao) {
107                 descarregar();
108                 carregar();
109                 estado = Estado.FIM_ANIMACAO;
110             }
111         } else {
112             iniAnimacao--;
113             if (iniAnimacao == 0) estado = Estado.JOGANDO;
114         }
115         return;
116     }
117 }

```

FONTE: Elaborada pelo autor

Figura 31: Classe Cenário Jogo Parte 03

```

118     if (Jogo.controleTecla[Jogo.Tecla.ESQUERDA.ordinal()]) {
119         dx = -1;
120     } else if (Jogo.controleTecla[Jogo.Tecla.DIREITA.ordinal()]) {
121         dx = 1;
122     }
123     if (dx != 0) {
124         dy = 0;
125     }
126     if (Jogo.controleTecla[Jogo.Tecla.CIMA.ordinal()]) {
127         dy = -1;
128     } else if (Jogo.controleTecla[Jogo.Tecla.BAIXO.ordinal()]) {
129         dy = 1;
130     }
131     if (dy != 0) {
132         dx = 0;
133     }
134
135     if (temporizador >= 20) {
136         temporizador = 0;
137
138         ponto.setPx(ponto.getPx() + LARG * dx); // POSIÇÃO ponto EIXO X
139         ponto.setPy(ponto.getPy() + LARG * dy); // POSIÇÃO ponto EIXO Y
140
141         if (Util.saiu(ponto, largura, altura)) {
142             ponto.setAtivo(false);
143             estado = Estado.PERDEU;
144         } else {
145             // COLISÃO COM CENÁRIO
146             for (int i = 0; i < contadorNivel; i++) {
147                 if (Util.colide(ponto, nivel[i])) {
148                     ponto.setAtivo(false);
149                     nivel[i].setCor(Color.YELLOW);
150                     estado = Estado.PERDEU;
151                     break;
152                 }
153             }
154             if (Util.colide(ponto, saida)) {
155                 // PAUSA DA MOVIMENTAÇÃO
156                 temporizador = -10;
157                 if (Jogo.nivel + 1 == Nivel.fases.length) {
158                     estado = Estado.GANHOU;
159                 } else {
160                     Jogo.nivel++;
161                     estado = Estado.INICIO_ANIMACAO;
162                 }
163             }
164         }
165     } else {
166         temporizador += ponto.getVel();
167     } // FIM if (temporizador >= 20) {
168
169 }
170

```

FONTE: Elaborada pelo autor

Figura 32: Classe Cenário Jogo Parte 04

```

171
172 @Override
173 public void desenhar(Graphics2D g) {
174     for (Elemento e : nivel) {
175         if (e == null) break;
176         e.desenhaRect(g);
177     }
178     saida.desenhaOval(g);
179     ponto.desenhaOval(g);
180
181     if (estado == Estado.INICIO_ANIMACAO || estado == Estado.FIM_ANIMACAO) {
182         char[][] nivelSelecioneado = Nivel.fases[Jogo.nivel];
183         int limite = 0;
184         g.setColor(Color.LIGHT_GRAY);
185         for (int linha = 0; linha < nivelSelecioneado.length; linha++) {
186             limite++;
187             if (limite >= iniAnimacao) break;
188             for (int coluna = 0; coluna < nivelSelecioneado[0].length; coluna++) {
189                 g.fillRect(LARG * coluna, LARG * linha, LARG, LARG);
190             }
191         }
192     } else if (estado == Estado.GANHOU || estado == Estado.PERDEU) {
193         if (estado == Estado.GANHOU) {
194             g.setColor(Color.YELLOW);
195             texto.desenha(g, "DESAFIO COMPLETO!", 110, 160);
196         } else {
197             g.setColor(Color.BLACK);
198             g.fillRect(ponto.getPx() + LARG * 2, ponto.getPy() + LARG * 2, 450, 450);
199             g.fillRect(ponto.getPx() - 450 - LARG, ponto.getPy() - 450 - LARG, 450, 450);
200             g.fillRect(ponto.getPx() + LARG * 2, ponto.getPy() - 450 - LARG, 450, 450);
201             g.fillRect(ponto.getPx() - 450 - LARG, ponto.getPy() + LARG * 2, 450, 450);
202             g.setColor(Color.YELLOW);
203             texto.desenha(g, "GAME OVER!", 160, 160);
204             texto.desenha(g, "PARA REINICIAR APERTE ENTER!", 20, 420);
205         }
206     }
207 }
208 }
209

```

FONTE: Elaborada pelo autor