UNIVERSIDADE REGIONAL DE BLUMENAU

CÂMPUS 1

CIÊNCIA DA COMPUTAÇÃO

NICOLE BRUCH E GABRIELLE CRISTINA BRAGA

INTRODUÇÃO A PROGRAMAÇÃO

IMPLEMENTAÇÃO DO JOGO CAÇA AO TESOURO

BLUMENAU

2025

SUMÁRIO

1 INTRODUÇÃO

Este trabalho tem como objetivo a implementação de um jogo simples em Java chamado Caça ao Tesouro, utilizando exclusivamente os conceitos aprendidos nas unidades 1 a 7 da disciplina de Introdução à Programação. O jogo é executado via terminal e simula uma escavação em uma ilha misteriosa, representada por uma matriz 8x8, onde estão escondidos 8 tesouros e 5 armadilhas.

O jogador deve inserir coordenadas para escavar o solo em busca dos tesouros, com um limite de até 25 tentativas. A cada escavação, o programa fornece uma mensagem sobre o que foi encontrado e atualiza o mapa visível. Ao final, o programa revela a posição de todos os tesouros e armadilhas e classifica o desempenho do jogador com base em sua pontuação.

2 FUNCIONALIDADES IMPLEMENTADAS

2.1 INICIALIZAÇÃO DO JOGO

Ao iniciar o jogo, duas matrizes 8x8 são criadas:

- Uma matriz real contendo a posição dos tesouros ('t'), armadilhas ('a') e areia ('~').
- Uma matriz visível contendo apenas areia ('~'), que será atualizada ao longo do jogo.

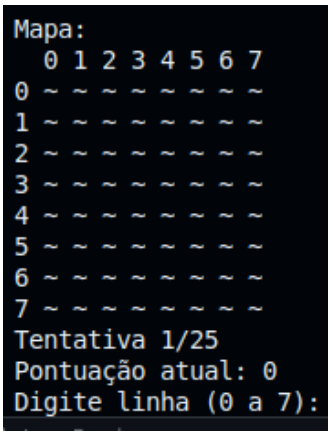
A colocação dos tesouros e armadilhas é feita de forma aleatória, utilizando a classe Random, garantindo que não haja sobreposição entre eles.

```
// cria um mapa com 8 linhas e 8 colunas, tipo uma tabela 8x8
  // cada posição guarda um caractere, que pode ser '~' pra areia, 't' pra tesouro
  // ou 'a' pra armadilha
  // esse é o mapa “real”, onde a gente esconde tudo
  char[][] mapa = new char[8][8];


  // cria outro mapa do mesmo tamanho, que é o que o jogador vai ver
  // inicialmente todo mundo vê só areia, aí conforme escava, a gente vai
  // mostrando o que tem de verdade
```

```
char[][] visivel = new char[8][8];


// conta quantas vezes o jogador já tentou escavar, começa zerado
int tentativas = 0;


// conta quantos tesouros o jogador já encontrou até agora, começa em zero
// também
int tesourosAchados = 0;


// guarda a pontuação do jogador, começa com zero, pode subir ou descer
// dependendo do que ele achar
int pontuacao = 0;


// scanner pra ler as entradas do usuario
Scanner scanner = new Scanner(System.in);


// cria um random pra sortear posições aleatórias no mapa, tipo pra colocar os
// tesouros e armadilhas sem ser previsível
Random random = new Random();


/*
 * aqui a gente tá preenchendo os dois mapas com areia ('~') em todas as
 * posições
 */
```

```
    // o primeiro for anda nas linhas (de 0 a 7)
    // o segundo for anda nas colunas dentro de cada linha (de 0 a 7)
    // mapa[i][j] quer dizer: “na linha i e coluna j do mapa”
    /*essa parte eu to preparando o mapa da caça ao tesouro
    vazio antes de colocar tesouros e armadilhas*/

    // esse for percorre as 8 linhas do mapa
    for (int i = 0; i < 8; i++) {
        // esse for percorre as 8 colunas de cada linha
        for (int j = 0; j < 8; j++) {
            // preenche o mapa escondido com areia
            mapa[i][j] = '~';
            // preenche o mapa visível com areia também
            visivel[i][j] = '~';
        }
}
```

Anexo A: Imagem ilustrativa da inicialização do jogo



2.2 ENCONTRO DE TESOURO

2.2.1 Posicionar os Tesouros no Início do Jogo

Antes do jogo começar, o programa utiliza a classe Random para posicionar 8 tesouros de forma aleatória em posições que ainda contenham apenas areia ('~'). Esse processo é feito antes do loop principal do jogo e garante que não haja sobreposição com armadilhas.

```
/*
     * aqui a gente cria uma variável pra contar quantos tesouros já foram colocados
     * no mapa
     */
    // começa zerada porque ainda não tem nenhum
    int colocados = 0;
    /*
     * enquanto a gente não tiver colocado os 8 tesouros, vai ficar
     * repetindo esse bloco de código
     */
    while (colocados < 8) {
        // escolhe uma linha aleatória de 0 a 7 usando o random
        // nextInt(8) quer dizer: pega um número aleatório entre 0 e 7 (8 não entra)
        int linha = random.nextInt(8);
        // escolhe uma coluna aleatória de 0 a 7 do mesmo jeito
        int coluna = random.nextInt(8);

        // verifica se a posição escolhida ainda está vazia, ou seja, tem só areia ('~')
        if (mapa[linha][coluna] == '~') {
            // se estiver vazia, coloca um tesouro ('t') naquela posição do mapa
```

```
            /*
             * isso quer dizer: no mapa, na linha X e coluna Y, agora tem um 't' que
             * representa o tesouro
             */
            mapa[linha][coluna] = 't';
            // aumenta o número de tesouros colocados em 1
            // quando chegar em 8, o while para
            colocados++;
        }
    }
```

2.2.2 Identificação de Tesouro Durante o Jogo

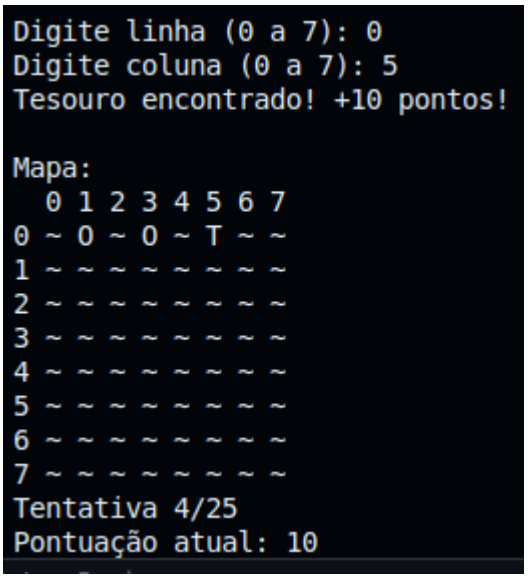
No loop principal do jogo, a cada escavação válida, o sistema verifica se a posição escavada contém um tesouro ('t') no mapa real. Se sim:

- Atualiza o mapa visível com 'T' (letra maiúscula).
- Soma +10 pontos na pontuação.
- Incrementa o contador de tesouros encontrados.
- Mostra a mensagem: "Tesouro encontrado! +10 pontos!"

```
            // se o jogador cavou e achou um tesouro
            if (mapa[linha][coluna] == 't') {
                /*
                 * mostra no mapa visível que ali tem um tesouro com T maiúsculo
                 * (pra destacar)
                 */
                visivel[linha][coluna] = 'T';
```

```
            // ganha 10 pontos por encontrar um tesouro
            pontuacao = pontuacao + 10;
            // aumenta o número de tesouros achados
            tesourosAchados++;
            System.out.println("Tesouro encontrado! +10 pontos!");
```

Anexo B: Imagem ilustrativa do encontro de tesouro



2.3 ENCONTRO DE ARMADILHA

2.3.1 Posicionar as armadilhas no Início do Jogo

Logo após o posicionamento dos tesouros, o programa sorteia 5 posições diferentes para inserir armadilhas ('a').

É necessário que essas armadilhas não sobreponham os tesouros ou outras armadilhas já colocadas. Isso é garantido pela verificação de que a posição contenha apenas areia ('~').

```
// aqui adiciono 5 armadilhas em lugares aleatórios (mas nao pode colocar em
    // cima de tesouro)
    // zera pra contar agora as armadilhas
    colocados = 0;
```

```
// esse while vai até colocar as 5 armadilhas
while (colocados < 5) {
    // sorteia uma linha aleatória de 0 a 7
    int linha = random.nextInt(8);
    // sorteia uma coluna aleatória de 0 a 7
    int coluna = random.nextInt(8);     /*
     * verifica se nessa posição do mapa ainda tem só areia
     * (ou seja, não tem tesouro nem armadilha)
     */
    if (mapa[linha][coluna] == '~') {       /*
         * se tiver só areia, coloca uma armadilha ali
         * ('a' é o símbolo da armadilha)      */
        mapa[linha][coluna] = 'a';      /*
         * aumenta o contador das armadilhas colocadas
         * pra saber quantas já tem              */
        colocados++;
    }
}
```

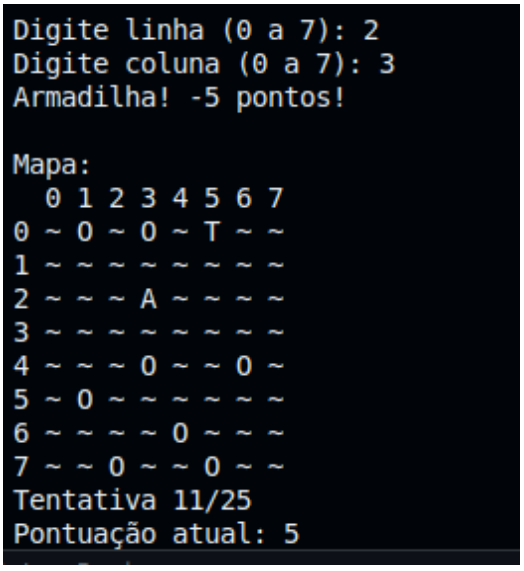
2.3.2 Identificação da Armadilha Durante o Jogo

Durante o loop principal do jogo, se o jogador escavar uma posição que contenha uma armadilha ('a'), o sistema:

- Atualiza o mapa visível com 'A' (letra maiúscula).
- Subtrai 5 pontos da pontuação do jogador.
- Exibe o feedback: "Armadilha! -5 pontos!"

```
            // se não tinha tesouro, mas tinha uma armadilha ali
            // o símbolo 'a' quer dizer armadilha
        } else if (mapa[linha][coluna] == 'a') {
            * marca com A maiúsculo no mapa visível pra mostrar que o jogador
            * caiu numa armadilha
            */
            visivel[linha][coluna] = 'A';
            // perde 5 pontos por ter caído na armadilha
            pontuacao = pontuacao - 5;
            System.out.println("Armadilha! -5 pontos!");
            // se não tinha nem tesouro nem armadilha, ou seja, só areia
```

Anexo C: Imagem ilustrativa do encontro de armadilha



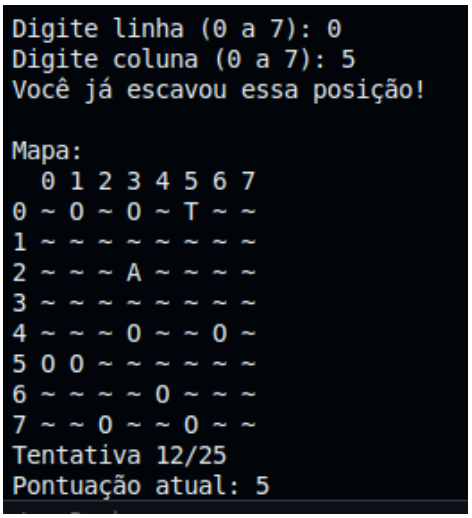
2.4 TENTATIVA DE ESCAVAR POSIÇÃO JÁ EXPLORADA

O programa verifica se o jogador está tentando escavar uma posição que já foi escavada (qualquer posição diferente de ~ no mapa visível).

Caso isso ocorra, uma mensagem é exibida e a tentativa não é contabilizada:

```
// aqui verifica se o jogador digitou algo fora do mapa
        // se for menor que 0 ou maior que 7, é inválido
        if (linha < 0 || linha > 7 || coluna < 0 || coluna > 7) {
            /*
             * se for inválido, avisa o jogador e pula pro próximo loop
             * (não conta como tentativa)
             */
            System.out.println("Coordenadas inválidas!");
            // aqui verifica se o jogador já escavou esse lugar antes
            // se já tiver qualquer coisa diferente de '~', é porque ele já cavou ali
        } else if (visivel[linha][coluna] != '~') {
            /*
             * se já cavou, avisa o jogador e volta pro começo do loop
             * (também não conta como tentativa)
             */
            System.out.println("Você já escavou essa posição!");
        } else {
            /*
             * se chegou aqui, a escavada é válida, então aumenta o número de
             * tentativas em 1
             */
            tentativas++;
```

Anexo D: Imagem ilustrativa da tentativa de escavar posição já explorada



2.5 *TELA DE VITÓRIA*

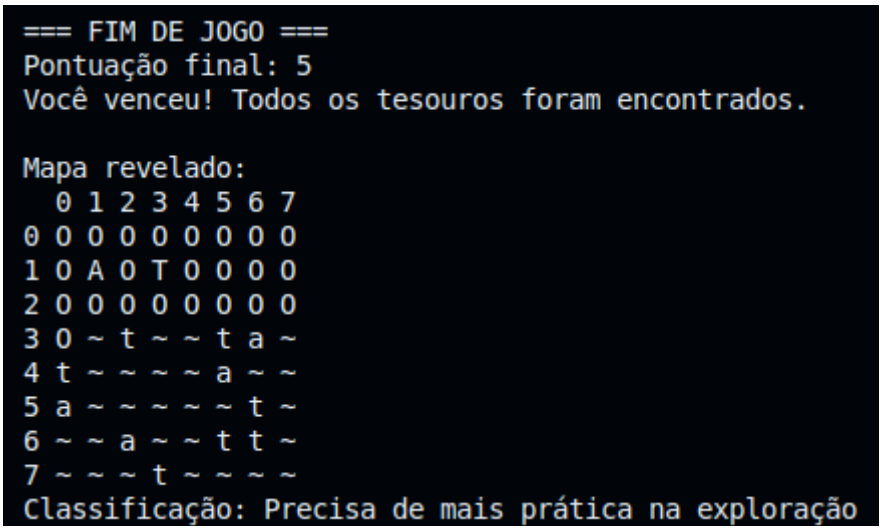
O jogo termina com vitória quando o jogador encontra todos os tesouros escondidos no mapa antes de atingir o limite de 25 escavações. Ao atingir esse objetivo, o programa:

- Exibe a mensagem: "Você venceu! Todos os tesouros foram encontrados."
- Mostra o mapa completo, revelando todos os tesouros (t) e armadilhas (a) que não haviam sido escavados.
- Exibe a pontuação final.
- Apresenta uma classificação final baseada na pontuação alcançada:
    - 70+ pontos: Explorador Lendário
    - 50–69 pontos: Caçador de Tesouros Experiente
    - 30–49 pontos: Aventureiro Iniciante
    - Menos de 30 pontos: Precisa de mais prática na exploração

```
// quando sair do while, significa que o jogo acabou
System.out.println("\n=== FIM DE JOGO ===");
// mostra a pontuação final
System.out.println("Pontuação final: " + pontuacao);
```

```java
// aqui verifica se o jogador achou os 8 tesouros

// se achou os 8 tesouros, ganhou

if (tesourosAchados == 8) {

    System.out.println("Você venceu! Todos os tesouros foram encontrados.");

    // se não, mostra que perdeu (acabaram as tentativas)
```

Anexo E: Imagem ilustrativa da tela de vitória



OBSERVAÇÃO: Para fins de documentação e ilustração no relatório, o número de tesouros no código foi temporariamente reduzido de 8 para 1, a fim de facilitar a vitória e permitir capturas de tela rápidas.

- Como consequência, no print da tela final, é exibida uma pontuação baixa (apenas 5 pontos) e a classificação aparece como "Precisa de mais prática na exploração", o que é esperado nessa simulação proposital.

2.6 TELA DE DERROTA COM REVELAÇÃO DO MAPA COMPLETO

Se o jogador atingir as 25 tentativas sem encontrar todos os tesouros:

- O jogo exibe uma mensagem de derrota.

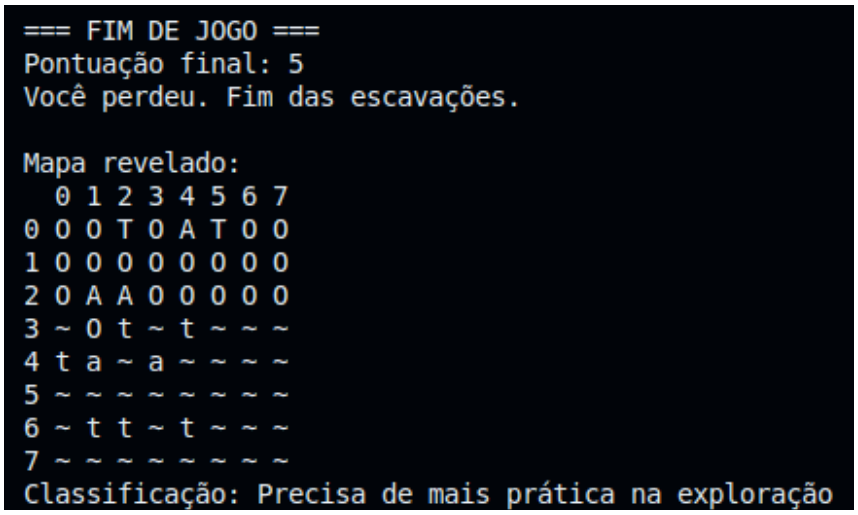
- Todas as posições com tesouros e armadilhas restantes são reveladas.
- Exibe também a pontuação final e a classificação do jogador.

```
} else {
    System.out.println("Você perdeu. Fim das escavações.");
}


/*
 * mostra o texto dizendo que agora o mapa real vai ser mostrado
 * (com todos os tesouros e armadilhas)
 */
System.out.println("\nMapa revelado:");
// esse for percorre todas as posições do mapa (linha por linha)
for (int i = 0; i < 8; i++) {
    // esse for percorre todas as posições do mapa (coluna por coluna)
    for (int j = 0; j < 8; j++) {
        /*
         * se naquela posição tinha um tesouro escondido
         * e o jogador não cavou ali, mostra o 't' agora
         */
        if (mapa[i][j] == 't' && visivel[i][j] == '~') {
            visivel[i][j] = 't';
        }
        /*
         * mesma ideia: se tinha armadilha escondida e
```

```
            * o jogador não descobriu, mostra agora
             */
            if (mapa[i][j] == 'a' && visivel[i][j] == '~') {
                visivel[i][j] = 'a';
            }
        }
    }

    /* mostra o mapa completo revelado, imprimindo o número das colunas */
    System.out.println("  0 1 2 3 4 5 6 7");
    // começa o for que vai passar por cada linha do mapa (de 0 até 7)
    for (int i = 0; i < 8; i++) {
        // antes de mostrar o conteúdo da linha, imprime o número da linha
        // esse número fica do lado esquerdo
        System.out.print(i + " ");
        // agora outro for que vai andar pelas colunas dessa linha
        for (int j = 0; j < 8; j++) {
            // imprime o símbolo que tá nessa posição do mapa visível
            // pode ser areia (~), tesouro (T), armadilha (A), vazio (O), etc.
            // o espaço depois do símbolo é só pra ficar bonitin
            System.out.print(visivel[i][j] + " ");
        }
        // depois de imprimir os 8 símbolos da linha, pula pra próxima linha
        System.out.println();
```

```
    }


    // mostra a classificação do jogador dependendo da pontuação
    System.out.print("Classificação: ");
    // se for maior ou igual a 70 pontos
    if (pontuacao >= 70)
        System.out.println("Explorador Lendário!");
    // se for maior ou igual a 50 pontos
    else if (pontuacao >= 50)
        System.out.println("Caçador de Tesouros Experiente!");
    // se for maior ou igual a 30 pontos
    else if (pontuacao >= 30)
        System.out.println("Aventureiro Iniciante");
    // se nao for maior que nenhum, significa que ele é menor que todos
    else
        System.out.println("Precisa de mais prática na exploração");


    scanner.close();
  }
}
```

Anexo F: Imagem ilustrativa da tela de derrota com revelação do mapa completo



3 CONCLUSÃO

Desenvolver o jogo *Caça ao Tesouro* foi uma experiência muito interessante e prática para aplicar os principais conceitos vistos ao longo da disciplina. Durante a implementação, foi possível colocar em prática o uso de matrizes, estruturas de repetição e decisão, além de validar entradas do usuário e trabalhar com a classe Random para gerar posições aleatórias no mapa.

Mais do que isso, o projeto ajudou a entender melhor como organizar o código de forma lógica e clara, como pensar no fluxo de um programa do início ao fim, e como tornar a interação com o usuário mais intuitiva, mesmo usando apenas o terminal. O sistema de pontuação e a classificação no final deram um toque a mais no desafio e tornaram o jogo mais divertido de jogar e testar.