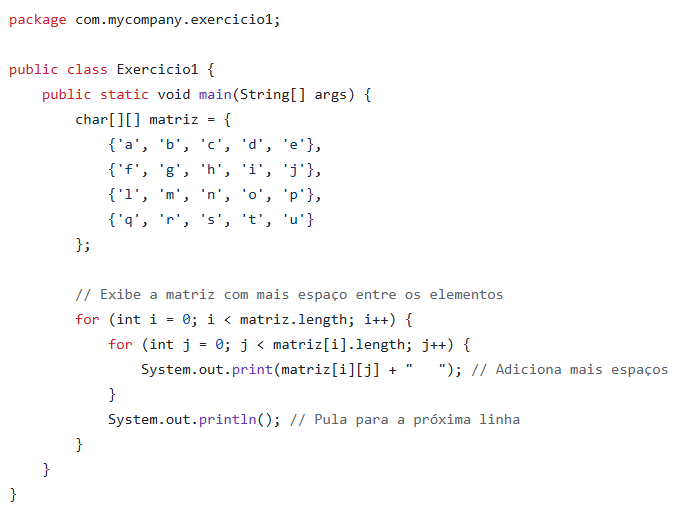
1. Crie programas em Java que crie e exiba as seguintes matrizes abaixo:

**A)** 

EXPLICAÇÂO:

Esse código cria uma matriz de 4 linhas por 5 colunas cheia de letras do alfabeto, já organizadas. O objetivo é mostrar essa matriz na tela, mas com um espaçamento maior entre as letras, só pra deixar tudo mais bonito e fácil de enxergar.

Primeiro, ele define a matriz assim:

char[][] matriz = {

{'a', 'b', 'c', 'd', 'e'},

{'f', 'g', 'h', 'i', 'j'},

{'l', 'm', 'n', 'o', 'p'},

{'q', 'r', 's', 't', 'u'}

};

Depois, vem a parte que exibe os elementos. Ele usa dois loops for pra passar por cada letra da matriz. O primeiro for cuida das linhas, e o segundo cuida das colunas. Pra cada letra, ele dá um print com três espaços extras, assim:

System.out.print(matriz[i][j] + " ");

Esses espaços extras deixam tudo mais espaçado. No final de cada linha, tem um println pra pular pra próxima linha, mantendo o formato certinho.

No final, quando você roda o programa, ele mostra isso no console:

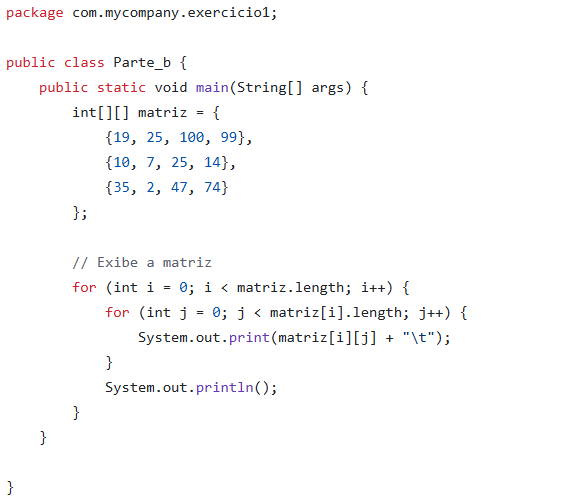
a b c d e

f g h i j

l m n o p

q r s t u

Ou seja, ele só pega a matriz e mostra bonitinha, com mais espaço entre as letras.

**B)** ****

EXPLICAÇÂO:

Esse código cria uma matriz com 3 linhas e 4 colunas, já preenchida com números inteiros. Ele simplesmente exibe a matriz na tela de forma organizada, com os números alinhados.

Primeiro, ele define a matriz assim:

int[][] matriz = {

{19, 25, 100, 99},

{10, 7, 25, 14},

{35, 2, 47, 74}

};

Depois, o programa usa dois **loops for** para passar por todos os números da matriz. O primeiro **for** cuida das linhas, e o segundo **for** vai pelos números de cada linha. Cada número é exibido usando System.out.print, com um tab (\t) no final pra criar um espacinho maior e alinhar tudo.

System.out.print(matriz[i][j] + "\t");

Quando termina de exibir uma linha inteira, ele usa System.out.println() pra pular pra próxima:

System.out.println();

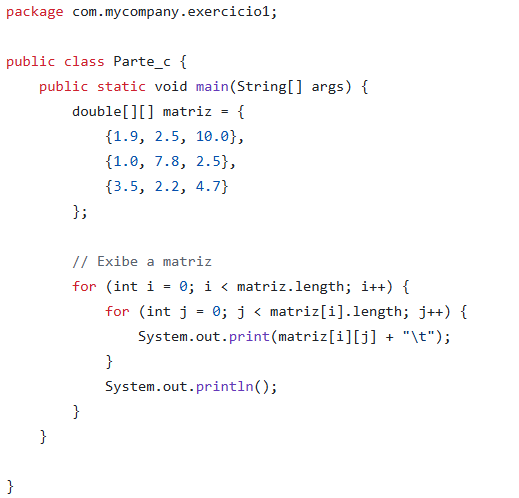
No final, o programa imprime a matriz no console assim:

19 25 100 99

10 7 25 14

35 2 47 74

**C)**



EXPLICAÇÂO:

Esse código cria uma matriz de 3 linhas por 3 colunas cheia de números decimais (double). Ele só serve pra exibir essa matriz na tela de forma organizada, com os números alinhados.

Primeiro, a matriz é definida assim:

double[][] matriz = {

{1.9, 2.5, 10.0},

{1.0, 7.8, 2.5},

{3.5, 2.2, 4.7}

};

Depois, ele usa dois **loops for** pra passar por todos os números da matriz. O primeiro **for** cuida das linhas, enquanto o segundo passa pelos elementos de cada linha. Cada número é mostrado com um tab (\t) pra criar espaço e manter o alinhamento:

System.out.print(matriz[i][j] + "\t");

Quando termina de exibir todos os números de uma linha, ele usa System.out.println() pra pular pra próxima:

System.out.println();

No final, a matriz aparece assim no console:

1.9 2.5 10.0

1.0 7.8 2.5

3.5 2.2 4.7

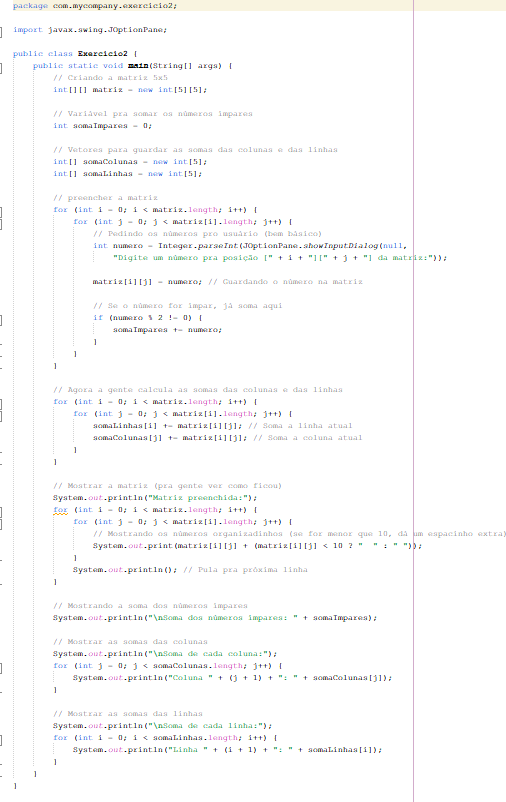
O programa é bem direto: ele pega a matriz e exibe tudo na tela, com os números organizados.

2. Faça um algoritmo que preencha uma matriz 5x5 de inteiros e escreva:

 a) a soma dos números ímpares fornecidos;

 b) a soma de cada uma das 5 colunas;

 c) a soma de cada uma das 5 linhas;



EXPLICAÇÂO:

Esse código cria uma matriz de 5 linhas por 5 colunas, onde o usuário vai preencher os números de cada célula. O programa também soma os números ímpares da matriz e as somas das linhas e colunas.

Primeiro, a matriz é definida assim:

int[][] matriz = new int[5][5];

Além disso, são criadas variáveis para armazenar as somas:

* somaImpares: para armazenar a soma dos números ímpares.
* somaColunas: para armazenar a soma de cada coluna.
* somaLinhas: para armazenar a soma de cada linha.

Depois, o programa preenche a matriz, pedindo ao usuário pra inserir os números:

int numero = Integer.parseInt(JOptionPane.showInputDialog(null,

"Digite um número pra posição [" + i + "][" + j + "] da matriz:"));

Se o número for ímpar (numero % 2 != 0), ele já adiciona esse número à variável somaImpares.

Depois de preencher a matriz, o código calcula as somas de cada linha e de cada coluna, usando dois loops:

somaLinhas[i] += matriz[i][j];

somaColunas[j] += matriz[i][j];

Por fim, o programa exibe a matriz na tela, com os números organizados. Pra fazer isso, ele usa o seguinte:

System.out.print(matriz[i][j] + (matriz[i][j] < 10 ? " " : " "));

E quando termina de exibir uma linha, ele pula pra próxima:

System.out.println();

O programa também exibe as somas dos números ímpares, das linhas e das colunas.

3.Crie em Java uma matriz 3x5 de inteiros, preencha a matriz e depois:

a) Informe se a matriz possui elementos repetidos;

b) A quantidade de números pares;

c) A quantidade de números ímpares;



EXPLICAÇÂO:

Esse código cria uma matriz 3x5 onde o usuário preenche os números de cada célula. Ele conta quantos números pares e ímpares existem na matriz, e verifica se há números repetidos.

Primeiro, a matriz é criada assim:

int[][] matriz = new int[3][5];

Depois, são criadas variáveis para armazenar as quantidades de números pares e ímpares:

* qtdPares: armazena a quantidade de números pares.
* qtdImpares: armazena a quantidade de números ímpares.

E também um vetor auxiliar elementos para armazenar todos os elementos da matriz e verificar repetições.

O programa então preenche a matriz pedindo os números ao usuário:

int numero = Integer.parseInt(JOptionPane.showInputDialog(

"Digite um número para a posição [" + i + "][" + j + "] da matriz:"));

A cada número inserido, ele verifica se é par ou ímpar:

if (numero % 2 == 0) {

qtdPares++;

} else {

qtdImpares++;

}

O código também verifica se há números repetidos, comparando cada número com os seguintes:

if (elementos[i] == elementos[j]) {

temRepetidos = true;

break;

}

Ao final, ele exibe a matriz organizada no terminal:

System.out.print(matriz[i][j] + (matriz[i][j] < 10 ? " " : " "));

Quando a linha termina, ele pula para a próxima com:

System.out.println();

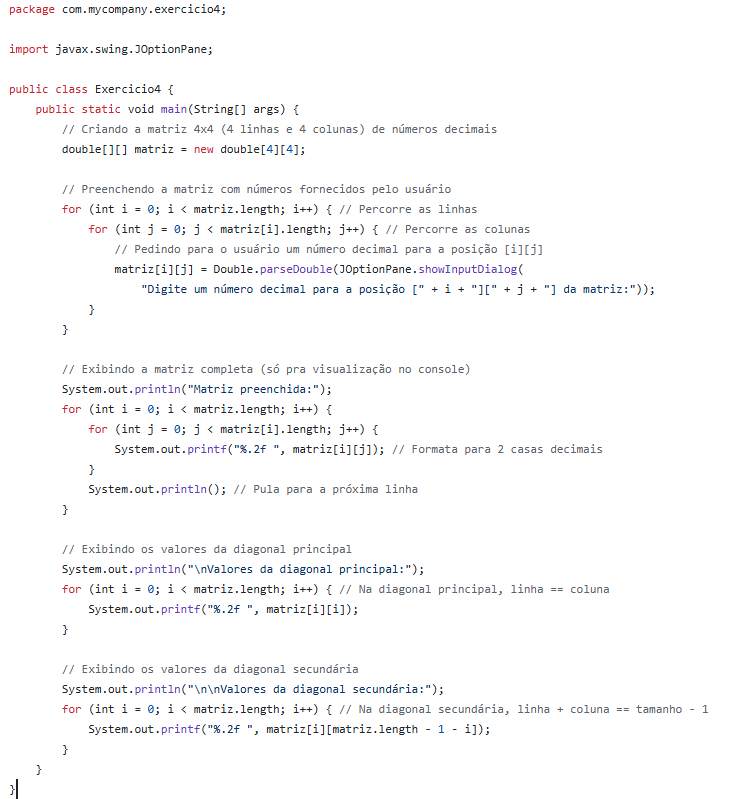
E mostra os resultados no terminal:

* Se há números repetidos.
* A quantidade de números pares.
* A quantidade de números ímpares.

4.Crie em Java uma matriz 4x4 de decimais, preencha a matriz e depois:

a) Exiba os valores da sua diagonal principal;

b) Exiba os valores da sua diagonal secundária;



EXPLICAÇÂO:

Esse código cria uma matriz 4x4 de números decimais onde o usuário preenche os valores. Depois, ele exibe a matriz, os valores da diagonal principal e os valores da diagonal secundária.

Primeiro, a matriz é criada dessa forma:

double[][] matriz = new double[4][4];

O programa então preenche a matriz com números decimais fornecidos pelo usuário, usando um laço for para percorrer as linhas e colunas:

matriz[i][j] = Double.parseDouble(JOptionPane.showInputDialog(

"Digite um número decimal para a posição [" + i + "][" + j + "] da matriz:"));

Em seguida, a matriz é exibida no console com a formatação de duas casas decimais:

System.out.printf("%.2f ", matriz[i][j]);

A matriz é impressa linha por linha, e cada número é mostrado com duas casas decimais.

Depois, o código exibe os valores da diagonal principal da matriz. A diagonal principal é formada pelos elementos onde o índice da linha é igual ao índice da coluna:

System.out.printf("%.2f ", matriz[i][i]);

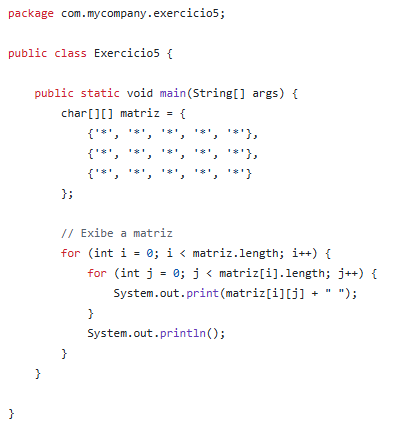
E finalmente, o código exibe os valores da diagonal secundária. A diagonal secundária é formada pelos elementos onde a soma do índice da linha e da coluna é igual ao tamanho da matriz menos 1:

System.out.printf("%.2f ", matriz[i][matriz.length - 1 - i]);

Ao final, o programa exibe a matriz preenchida, seguida pelas diagonais principal e secundária.

5.Crie programas em Java que crie e exiba as seguintes figuras abaixo utilizando

**A)**

****

EXPLICAÇÂO:

Esse código cria uma matriz 3x5 de caracteres, onde cada elemento é um asterisco (\*). Ele apenas exibe a matriz na tela, com cada elemento separado por um espaço, e cada linha sendo impressa uma abaixo da outra.

Primeiro, a matriz é definida assim:

char[][] matriz = {

{'\*', '\*', '\*', '\*', '\*'},

{'\*', '\*', '\*', '\*', '\*'},

{'\*', '\*', '\*', '\*', '\*'}

};

Aqui, a matriz tem 3 linhas e 5 colunas, e cada célula é preenchida com o caractere \*.

O programa então percorre a matriz com dois loops for:

for (int i = 0; i < matriz.length; i++) {

for (int j = 0; j < matriz[i].length; j++) {

System.out.print(matriz[i][j] + " ");

}

System.out.println();

}

O primeiro for percorre as linhas da matriz, e o segundo percorre as colunas de cada linha. Dentro do loop, o caractere da posição atual é impresso seguido por um espaço.

Ao final de cada linha, System.out.println() é chamado para pular para a próxima linha da matriz.

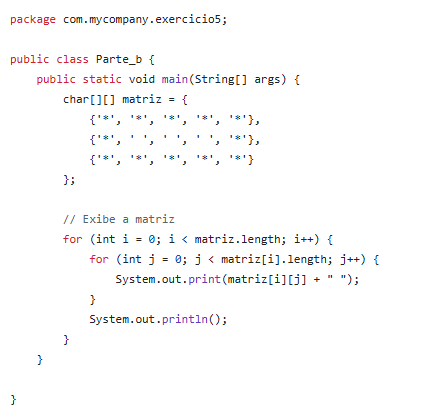
A saída da matriz será:

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

**B)**



EXPLICAÇÂO:

Esse código cria uma matriz 3x5 de caracteres, mas desta vez com um padrão mais interessante. Ele cria uma figura que se assemelha a um "quadrado" com bordas feitas de asteriscos (\*) e o centro vazio, representado por espaços.

A matriz é definida assim:

char[][] matriz = {

{'\*', '\*', '\*', '\*', '\*'},

{'\*', ' ', ' ', ' ', '\*'},

{'\*', '\*', '\*', '\*', '\*'}

};

Aqui, temos 3 linhas e 5 colunas. A primeira e a última linha são completamente preenchidas com asteriscos (\*), enquanto a linha do meio tem asteriscos nas extremidades e espaços no meio.

O programa percorre a matriz com dois loops for para exibir a matriz na tela:

for (int i = 0; i < matriz.length; i++) {

for (int j = 0; j < matriz[i].length; j++) {

System.out.print(matriz[i][j] + " ");

}

System.out.println();

}

O primeiro for percorre as linhas, e o segundo percorre as colunas de cada linha. Dentro do loop, o caractere da posição atual é impresso seguido por um espaço. Quando o loop termina de imprimir os caracteres de uma linha, ele pula para a próxima linha.

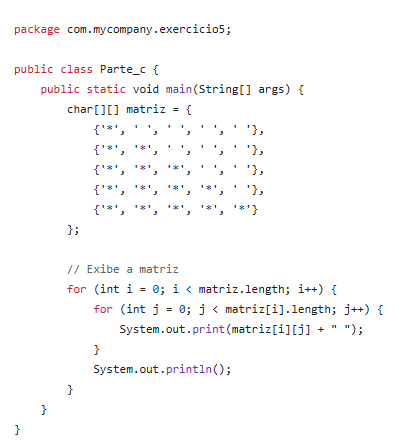
A saída da matriz será:

\* \* \* \* \*

\* \*

\* \* \* \* \*

**C)**



EXPLICAÇÂO:

Esse código cria uma matriz 5x5, onde os elementos são organizados para formar uma figura triangular com asteriscos (\*). A matriz é preenchida de tal maneira que a figura parece um triângulo crescente de asteriscos, onde cada linha tem mais asteriscos do que a anterior.

Aqui está como a matriz é definida:

char[][] matriz = {

{'\*', ' ', ' ', ' ', ' '},

{'\*', '\*', ' ', ' ', ' '},

{'\*', '\*', '\*', ' ', ' '},

{'\*', '\*', '\*', '\*', ' '},

{'\*', '\*', '\*', '\*', '\*'}

};

O programa percorre essa matriz e imprime os elementos linha por linha com espaços entre eles:

for (int i = 0; i < matriz.length; i++) {

for (int j = 0; j < matriz[i].length; j++) {

System.out.print(matriz[i][j] + " ");

}

System.out.println();

}

Quando o programa é executado, ele exibe a seguinte figura no console:

\*

\* \*

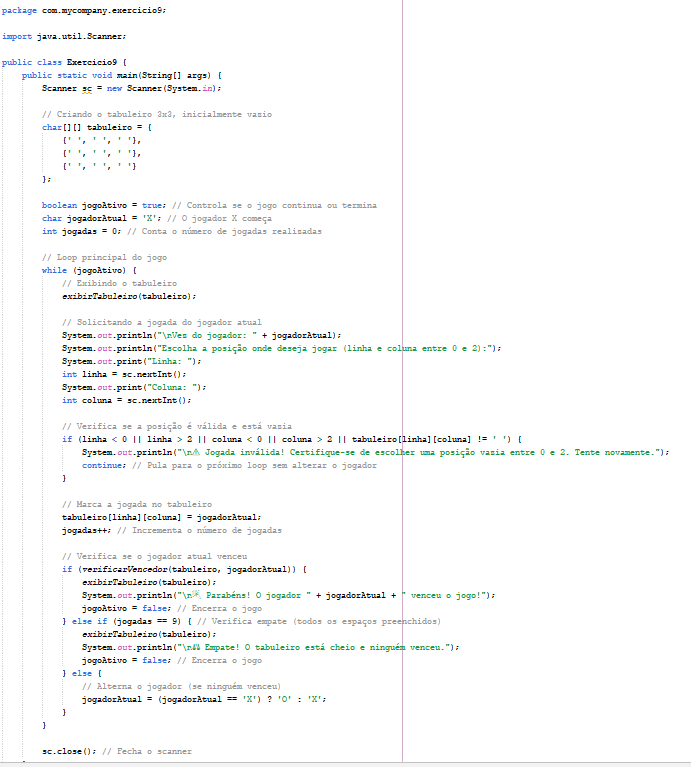
\* \* \*

\* \* \* \*

\* \* \* \* \*

9) criar um jogo em uma matriz representando:

* um tabuleiro no jogo da velha;
* jogador X;
* jogador O.
* Cada jogador tem sua vez de opção de ataque fazendo a opção na matriz;
* No final mostra o vencedor ou empate.





EXPLICAÇÂO:

Esse código é um jogo de X e O em Java. Ele começa criando o tabuleiro 3x3 vazio, onde os jogadores vão marcar suas jogadas. O jogo vai rodar em um laço, onde a cada rodada o tabuleiro é mostrado, e o jogador escolhe onde quer jogar, dizendo a linha e a coluna. O código verifica se a posição escolhida é válida (se está dentro do tabuleiro e se a célula não está ocupada). Se for inválido, o jogador tem que tentar de novo.

Quando a jogada é válida, o símbolo do jogador (X ou O) é colocado no tabuleiro e o número de jogadas aumenta. O código então verifica se o jogador venceu, checando as linhas, colunas e diagonais. Se alguém ganhou, o jogo acaba e é mostrado quem venceu. Se não tiver vencedor e todas as 9 jogadas forem feitas, o jogo empata. Se ainda não acabou, o código passa a vez para o outro jogador, alternando entre X e O. No final, o scanner é fechado e o jogo termina.