Código

package lista;

public class Lista06 {

public static void main(String[] args) {

int[][] num = {{1,1,1},{4,4,4},{9,9,9}};

boolean passou = false;

if (num[1][1] == 4) {

passou = true;

System.out.println(passou);

} else {

System.out.println(passou);

}

if (passou) {

for (int i = 0; i <= 2; i++) {

for (int j = 0; j <= 2; j++) {

if (i == j) { // Corrigido: Verifica se i é igual a j, não j == 0

System.out.println("\n");

}

System.out.println(num[i][j]); // Corrigido: Acesso aos elementos corretos num[i][j]

}

}

System.out.println("Tam num -> " + num.length);

} else {

System.out.println("\nErro");

}

}

}

1. **Qual a saida em System.out.println(num[i][i];?**

R: A saída em `System.out.println(num[i][i]);` será a seguinte:

1

4

9

Isso ocorre porque o código itera sobre a matriz `num` utilizando dois loops `for`, um para percorrer as linhas (variável `i`) e outro para percorrer as colunas (variável `j`).

Quando utilizamos `System.out.println(num[i][i]);`, estamos imprimindo o elemento da matriz localizado na posição `[i][i]`. Como `i` representa tanto a linha quanto a coluna atual, a instrução imprime os elementos da diagonal principal da matriz `num`.

No caso da matriz fornecida no código corrigido, a diagonal principal contém os valores 1, 4 e 9. Portanto, a saída será:

1

4

9

**B- Que tipo de validação faz a variável passou e por que é importante no** codigo?

R: A variável `passou` é utilizada como um indicador booleano para registrar se a condição `num[1][1] == 4` foi verdadeira. Ela serve como uma forma de controle para determinar se um determinado trecho de código deve ser executado ou não.

No código fornecido, se a condição `num[1][1] == 4` for verdadeira, a variável `passou` é definida como `true`. Isso significa que o bloco de código dentro do primeiro `if` será executado e imprimirá o valor de `passou`, que no caso será `true`. Por outro lado, se a condição for falsa, o bloco de código dentro do `else` será executado e imprimirá o valor de `passou`, que no caso será `false`.

A validação da variável `passou` é importante para controlar o fluxo do programa e garantir que apenas o trecho de código adequado seja executado com base na condição fornecida. Isso ajuda a evitar a execução desnecessária de blocos de código e melhora a eficiência e legibilidade do programa.

**C- Qual tamanho sera impresso em num.length?**

R: O tamanho impresso em `num.length` será 3.

No código fornecido, a variável `num` é uma matriz bidimensional com 3 linhas e 3 colunas. Portanto, `num.length` retorna o tamanho da primeira dimensão da matriz, que é o número de linhas. Neste caso, como existem 3 linhas na matriz, o valor retornado será 3.

**D- Modifique o codigo para imprimir a saida "1,4,9,1,4,9,1,4,9"**

R: Para imprimir a sequência "1, 4, 9, 1, 4, 9, 1, 4, 9", é necessário fazer algumas modificações no código fornecido. Aqui está a versão modificada:

java  
public class Lista06 {

public static void main(String[] args) {

int[][] num = {{1, 1, 1}, {4, 4, 4}, {9, 9, 9}};

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

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

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

}

}

}

}

**E-Crie os métodos criaMatriz() e mostraMatriz(int[][]n) para deixar o main() mais "limpo" e devolver exatamente as mesmas saídas que o código acima faz.**

**R: public class Lista06 {**

public static void main(String[] args) {

int[][] num = criaMatriz();

mostraMatriz(num);

}

public static int[][] criaMatriz() {

int[][] num = {{1, 1, 1}, {4, 4, 4}, {9, 9, 9}};

return num;

}

public static void mostraMatriz(int[][] n) {

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

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

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

}

}

}

}

**F- No método criaMatriz() instancie o objeto Scanner(System.in) para permitir a entrada de dados via teclado.**R: Desculpe pelo equívoco anterior. Se você deseja permitir a entrada de dados via teclado no método `criaMatriz()`, é necessário passar o objeto `Scanner` como parâmetro para o método. Aqui está o código atualizado:

```java

import java.util.Scanner;

public class Lista06 {

public static void main(String[] args) {

int[][] num = criaMatriz(new Scanner(System.in));

mostraMatriz(num);

}

public static int[][] criaMatriz(Scanner entrada) {

int[][] num = new int[3][3];

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

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

System.out.print("Digite o valor para a posição [" + i + "][" + j + "]: ");

num[i][j] = entrada.nextInt();

}

}

return num;

}

public static void mostraMatriz(int[][] n) {

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

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

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

}

}

}

}

```

Agora, o método `criaMatriz()` recebe um objeto `Scanner` como parâmetro, permitindo a entrada de valores para preencher a matriz a partir do teclado.

**G- Como você modificaria este código para permitir á variável num receber as notas de aluno a cada trimestre e, ao final, devolver como resultado se ele passou ou não e qual seria a sua média anual?**

R: Para modificar o código para receber as notas do aluno a cada trimestre e calcular a média anual, você pode fazer as seguintes alterações:

```java

import java.util.Scanner;

public class NotasAluno {

public static void main(String[] args) {

double[] notasTrimestre = new double[3];

Scanner entrada = new Scanner(System.in);

// Receber as notas de cada trimestre

for (int i = 0; i < 3; i++) {

System.out.print("Digite a nota do " + (i + 1) + "º trimestre: ");

notasTrimestre[i] = entrada.nextDouble();

}

// Calcular a média anual

double mediaAnual = calcularMediaAnual(notasTrimestre);

// Verificar se o aluno passou ou não

boolean passou = verificarAprovacao(mediaAnual);

// Exibir o resultado

System.out.println("Média anual: " + mediaAnual);

if (passou) {

System.out.println("O aluno passou de ano!");

} else {

System.out.println("O aluno não passou de ano.");

}

}

public static double calcularMediaAnual(double[] notasTrimestre) {

double soma = 0;

for (double nota : notasTrimestre) {

soma += nota;

}

return soma / notasTrimestre.length;

}

public static boolean verificarAprovacao(double mediaAnual) {

return mediaAnual >= 7.0;

}

}

```

Nesse código, a variável `notasTrimestre` é um vetor que armazena as notas do aluno em cada trimestre. O usuário é solicitado a inserir as notas de cada trimestre usando um loop `for`. Em seguida, a média anual é calculada chamando o método `calcularMediaAnual`, que recebe o vetor de notas como parâmetro e retorna a média. A função `verificarAprovacao` é usada para determinar se o aluno passou ou não, com base na média anual (considerando a média de aprovação como 7.0).

Por fim, os resultados são exibidos na tela, mostrando a média anual e se o aluno passou ou não.

**H- Qual a diferença entre uma matris notasTrimestrais[2][2] e os vetores primeirotrimestre[2], segundoTrimestre e terceiroTrimestre[2] no programa acima?**

R: No código fornecido, a diferença entre a matriz `notasTrimestrais[2][2]` e os vetores `primeiroTrimestre[2]`, `segundoTrimestre[2]` e `terceiroTrimestre[2]` é a estrutura de dados utilizada para armazenar as notas dos trimestres.

A matriz `notasTrimestrais[2][2]` é uma matriz bidimensional de tamanho 2x2, ou seja, possui duas linhas e duas colunas. Ela é utilizada para armazenar as notas dos trimestres em um formato tabular, onde cada elemento da matriz representa a nota de um determinado trimestre.

Por outro lado, os vetores `primeiroTrimestre[2]`, `segundoTrimestre[2]` e `terceiroTrimestre[2]` são vetores unidimensionais de tamanho 2. Cada vetor representa as notas de um trimestre específico. Ou seja, `primeiroTrimestre` armazena as notas do primeiro trimestre, `segundoTrimestre` armazena as notas do segundo trimestre e `terceiroTrimestre` armazena as notas do terceiro trimestre.

A diferença fundamental entre as duas abordagens é a forma como as notas são organizadas e acessadas. Com a matriz `notasTrimestrais`, é possível acessar uma nota específica indicando a linha e a coluna correspondente, enquanto com os vetores individuais, cada vetor representa as notas de um trimestre, tornando mais direto o acesso às notas de cada período separadamente.

Em resumo, a matriz `notasTrimestrais` organiza as notas de todos os trimestres em uma única estrutura de dados multidimensional, enquanto os vetores individuais `primeiroTrimestre`, `segundoTrimestre` e `terceiroTrimestre` armazenam as notas de cada trimestre separadamente em estruturas de dados unidimensionais.

**I - E se a regra do negocio não prevesse a quantidade certa de notas para cada trimestre, como seria a implementação do código?**

R: Se a quantidade de notas para cada trimestre não for fixa e pode variar, seria necessário utilizar uma estrutura de dados flexível, como um ArrayList, para armazenar as notas dos trimestres. Assim, a implementação do código seria ajustada da seguinte maneira:

import java.util.ArrayList;

import java.util.Scanner;

public class NotasTrimestrais {

public static void main(String[] args) {

Scanner entrada = new Scanner(System.in);

ArrayList<ArrayList<Double>> notasTrimestrais = new ArrayList<>();

System.out.println("Informe as notas dos trimestres (digite -1 para encerrar):");

int trimestre = 1;

while (true) {

ArrayList<Double> notasTrimestre = new ArrayList<>();

System.out.println("Trimestre " + trimestre + ":");

System.out.println("Digite as notas (digite -1 para encerrar o trimestre):");

while (true) {

double nota = entrada.nextDouble();

if (nota == -1) {

break; // Encerra a leitura do trimestre

}

notasTrimestre.add(nota);

}

if (notasTrimestre.isEmpty()) {

break; // Encerra a leitura de notas trimestrais

}

notasTrimestrais.add(notasTrimestre);

trimestre++;

}

// Realizar cálculos e exibir resultados

int numTrimestres = notasTrimestrais.size();

double somaNotas = 0.0;

int numNotas = 0;

for (ArrayList<Double> notasTrimestre : notasTrimestrais) {

for (Double nota : notasTrimestre) {

somaNotas += nota;

numNotas++;

}

}

double mediaAnual = somaNotas / numNotas;

System.out.println("\nResultados:");

System.out.println("Número de trimestres: " + numTrimestres);

System.out.println("Número total de notas: " + numNotas);

System.out.println("Média anual: " + mediaAnual);

}

}Nesse código, utilizamos um **ArrayList<ArrayList<Double>>** chamado **notasTrimestrais** para armazenar as notas de cada trimestre. A leitura das notas é feita iterativamente, permitindo ao usuário digitar quantas notas desejar para cada trimestre. O código também realiza o cálculo da média anual com base nas notas fornecidas.

Dessa forma, o programa se adapta à quantidade variável de notas para cada trimestre, garantindo a flexibilidade necessária.

**J- Diante do erro java.lang.ArrayIndexOutOfboundsExcption como linguagem Java permite evitar que o programa seja encerrado, ou seja, se "recuperar do erro"?**

R: A classe Java `ArrayIndexOutOfBoundsException` é uma exceção que ocorre quando você tenta acessar um índice inválido em um array. Essa exceção ocorre quando você tenta acessar um elemento em um índice que está fora dos limites do array (ou seja, menor que 0 ou maior ou igual ao comprimento do array).

Para evitar que o programa seja encerrado abruptamente quando ocorre uma exceção, você pode utilizar mecanismos de tratamento de exceção para recuperar-se do erro. Em Java, isso é feito utilizando blocos `try-catch` para capturar exceções específicas e executar um código alternativo em caso de erro.

Aqui está um exemplo de como você pode tratar a exceção `ArrayIndexOutOfBoundsException` em seu programa:

```java

import java.util.Scanner;

public class ExemploTratamentoExcecao {

public static void main(String[] args) {

Scanner entrada = new Scanner(System.in);

int[] numeros = {1, 2, 3, 4, 5};

System.out.print("Digite o índice do elemento a ser acessado: ");

int indice = entrada.nextInt();

try {

int elemento = numeros[indice];

System.out.println("Elemento encontrado: " + elemento);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Erro: Índice inválido. O array não possui um elemento nesse índice.");

}

System.out.println("Programa continua executando após o tratamento da exceção.");

}

}

```

No exemplo acima, utilizamos o bloco `try-catch` para envolver o código que pode lançar a exceção `ArrayIndexOutOfBoundsException`. Se ocorrer essa exceção, o bloco `catch` é executado, exibindo uma mensagem de erro apropriada. Dessa forma, o programa não será encerrado abruptamente, e você tem a oportunidade de tratar o erro de forma adequada.

É importante ressaltar que, ao tratar exceções, é recomendado identificar a exceção específica que está sendo tratada para garantir que o tratamento seja aplicado apenas a esse tipo de exceção. Isso ajuda a manter o código organizado e permite que diferentes tipos de exceções sejam tratados de maneira apropriada.