PROGRAMAÇÃO ORIENTADA A OBJETOS

Nome:	**RGM:**
GABRIEL NEME HIGA SILVA DE OLIVEIRA	33586802

Pesquisa em POO

Divisão de Números Inteiros em Java

SÃO PAULO

2024

Nome: **RGM:**
GABRIEL NEME HIGA SILVA DE OLIVEIRA 33586802

Pesquisa em POO

Divisão de Números Inteiros em Java

Trabalho apresentado a Universidade Cruzeiro do Sul como requisito para um trabalho do curso Programação Orientada a Objetos.

Orientador: Prof. Fabio Luiz Peral

SÃO PAULO

2024

RESUMO

Este estudo aborda a implementação de um algoritmo para realizar divisão de números inteiros em Java. O objetivo geral da pesquisa é fornecer soluções eficientes e padronizadas para tais operações matemáticas. A importância da aplicação de normas técnicas no desenvolvimento de software é explorada através de bibliografia e métodos práticos. A pesquisa envolve o desenvolvimento de um algoritmo em Java que garanta a eficiência e confiabilidade da divisão de inteiros utilizando técnicas de programação apropriadas. O desenvolvimento de algoritmos é apoiado teoricamente por métodos de pesquisa bibliográfica combinados com métodos práticos de implementação e validação do código. Os resultados obtidos mostram que a implementação do algoritmo de divisão inteira em Java fornece uma solução confiável e eficiente para esta operação matemática. Aderir às boas práticas de programação e utilizar técnicas adequadas produz código de alta qualidade, o que contribui para a excelência no desenvolvimento de software.

Palavras-chaves: Algoritmo; Java; Divisão de números inteiros; Desenvolvimento de software.

SÃO PAULO

2024

SUMÁRIO

SÃO PAULO

2024

1 INTRODUÇÃO

A tecnologia está a avançar a um ritmo sem precedentes e o software é agora utilizado em quase todos os aspectos das nossas vidas – isto sublinha a necessidade de algoritmos que não sejam apenas eficientes, mas também fiáveis. Apesar de sua simplicidade, a divisão de inteiros é uma das operações matemáticas mais comuns utilizadas em aplicações do mundo real; desenvolver um algoritmo que possa executar esta operação com precisão e robustez exige considerações cuidadosas sobre sua implementação.

A pesquisa visa criar um algoritmo Java para divisão de números inteiros com o objetivo principal de fornecer uma solução de alta qualidade que garanta resultados consistentes em diferentes casos de uso, ao mesmo tempo que mantém as melhores práticas e padrões do setor. Isso é feito com o intuito de promover padronização e interoperabilidade entre nossa base de código.

2 DETALHAMENTO DO CÓDIGO

2.1.1 CAPTURA DE ENTRADA DO USUÁRIO

```
Scanner input = new Scanner(System.in);

        System.out.println("Insira o valor do dividendo: ");
        int dividendo = input.nextInt();

        System.out.println("Insira o valor do divisor: ");
        int divisor = input.nextInt();
```

O programa utiliza a classe Scanner para capturar o número digitado pelo usuário, que será empregado como dividendo na operação de divisão. Após solicitar ao usuário que insira o valor do dividendo, o programa lê essa entrada e a armazena em uma variável específica para posterior utilização no cálculo da divisão.

2.1.2 VERIFICAÇÃO DE DIVISÃO POR ZERO

```
if (divisor == 0) {
            System.out.println("-1");
        }
```

Nesta parte do código, uma verificação é realizada para garantir que o divisor não seja zero, uma vez que a divisão por zero não é definida. Se a condição de que o divisor é igual a zero for verdadeira, o programa imprime '-1' como resultado.

2.1.3 VERIFICAÇÃO DE DIVISÃO NEGATIVA

```
else if ((dividendo / divisor < 0)) {
            System.out.println("Valor encontrado: 0");
        }
```

Nessa parte específica do código, após garantir que o divisor não é zero, há uma verificação adicional. O programa analisa se o resultado da divisão é negativo. Se essa condição for verdadeira, o programa imprime '0' como resultado.

2.1.4 CÁLCULO DA DIVISÃO

```
    else {
        System.out.println("Valor calculado: " + dividendo / divisor);
    }
```

Nessa última parte do código, depois de confirmar que o divisor não é zero e que o resultado da divisão não é negativo, o programa executa o cálculo da divisão dos dois números fornecidos pelo usuário. Após calcular o resultado, o programa o imprime na tela.

2.2.1 CLASSE

```
public class DivisaoDeNumeros {
```

A classe principal do programa, responsável por executar a lógica da divisão de números digitados pelo usuário.

2.2.2 MÉTODOS

```
public static void main(String[] args) {
```

Este método é o ponto de entrada do programa. Ele controla o fluxo de execução e chama outros métodos conforme necessário para realizar a divisão dos números digitados.

```
input.nextInt();
```

O método nextInt() é chamado em um objeto Scanner para capturar o próximo número inteiro digitado pelo usuário a partir da entrada padrão.

2.2.3 ATRIBUTOS

```
int dividendo = input.nextInt();
int divisor = input.nextInt();
```

Aqui, a variável “dividendo” é do tipo int, o que significa que ela armazenará um número inteiro. Ela é utilizada para guardar o valor fornecido pelo usuário que será utilizado como dividendo na operação de divisão. Da mesma forma, a variável “divisor” também é do tipo int, sendo utilizada para armazenar o valor fornecido pelo usuário que será utilizado como divisor na operação de divisão.

3 FUNÇÕES

3.1 FUNÇÃO PRINCIPAL [MAIN]

```
    public static void main(String[] args) {
```

- Lógica interna da função: Esta função controla o fluxo principal do programa. Ela captura os valores do dividendo e do divisor digitados pelo usuário, verifica se o divisor é zero, se o resultado da divisão é negativo e, finalmente, calcula e imprime o resultado da divisão.

3.2 CAPTURA DE ENTRADA DO USUÁRIO

```
   Scanner input = new Scanner(System.in);

        System.out.println("Insira o valor do dividendo: ");
        int dividendo = input.nextInt();

        System.out.println("Insira o valor do divisor: ");
        int divisor = input.nextInt();
```

-Lógica interna da função: Utiliza um objeto da classe Scanner para capturar os números digitados pelo usuário, que serão utilizados como dividendo e divisor na operação de divisão.

3.3 VERIFICAÇÃO DE DIVISÃO POR ZERO

```
if (divisor == 0) {
            System.out.println("-1");
        }
```

-Parâmetros de entrada: O divisor (int).
-Lógica interna da função: Verifica se o divisor é igual a zero. Se for, imprime "-1" como resultado, indicando que a operação não pode ser realizada.
-Exemplo de uso: Se o usuário inserir "0" como divisor, a função imprimirá "-1".

3.4 VERIFICAÇÃO DE DIVISÃO NEGATIVA

```
else if ((dividendo / divisor < 0)) {
            System.out.println("Valor encontrado: 0");
        }
```

-Parâmetros de entrada: O dividendo (int) e o divisor (int).

-Lógica interna da função: Verifica se o resultado da divisão é negativo. Se for, imprime "0" como resultado.

-Exemplo de uso: Se a divisão entre o dividendo e o divisor resultar em um número negativo, a função imprimirá "0".

3.5 CÁLCULO DA DIVISÃO

```
    else {
       System.out.println("Valor calculado: " + dividendo / divisor);
    }
```

-Parâmetros de entrada: O dividendo (int) e o divisor (int).

-Lógica interna da função: Realiza o cálculo da divisão dos dois números fornecidos. Se o divisor não for zero e o resultado da divisão não for negativo, imprime o resultado da divisão.

-Exemplo de uso: Se o usuário inserir "10" como dividendo e "2" como divisor, a função imprimirá "5".

4 CONCLUSÃO

Nesta pesquisa, exploramos as funcionalidades necessárias para implementar um algoritmo de divisão de números em Java. Aprendemos sobre validação de entrada, lógica de divisão e tratamento de casos especiais, como divisão por zero e resultados negativos. Isso nos permitirá criar um algoritmo robusto e eficiente para realizar operações de divisão em Java.

5 REFERÊNCIAS

262588213843476. Algoritmo: Código simples resolvendo um problema de divisão, em Java. Disponível em: <https://gist.github.com/alexandregama/747582>. Acesso em: 14 maio. 2024.