Implementação de um Algoritmo de Cálculo de Dias Úteis em Java

No contexto da programação moderna, um dos desafios recorrentes está na manipulação e cálculo de datas, especialmente quando o objetivo é identificar dias úteis, ignorando feriados e finais de semana. Neste artigo, iremos abordar um exemplo prático de como implementar um cálculo de dias úteis utilizando a linguagem Java.

Introdução

O código que será discutido trata de um cenário onde o usuário passa uma data de vencimento de título (dataVenctoTitulo), e o sistema precisa calcular o número de dias úteis entre a data atual e a data informada. Isso é crucial em sistemas financeiros, especialmente para cálculo de prazos e penalidades.

Estrutura da Classe

A classe principal que realiza o cálculo está localizada no pacote br.java.diasUteis. Abaixo, detalhamos cada uma das funções relevantes, abordando sua lógica e como os métodos são utilizados no cálculo.

1. Importação das Bibliotecas

```
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
```

Para o cálculo de datas e identificação dos dias da semana, utilizamos a API de data e hora (java.time) introduzida no Java 8. Essa API fornece classes como LocalDate, que representa uma data sem fuso horário, e DayOfWeek, que enumera os dias da semana. Além disso, o DateTimeFormatter é utilizado para converter as datas no formato desejado.

2. Método main

```
public static void main(String[] args) {
    result_Calcula_Regra_DiasUteis_ODM("2024-07-05");
}
```

No método principal, é invocado o cálculo de dias úteis a partir de uma data fixa. No exemplo acima, a data de vencimento utilizada é "2024-07-05". Esse valor é passado como argumento para o método responsável pelo cálculo dos dias úteis.

3. Método result_Calcula_Regra_DiasUteis_ODM

Este método é o ponto central da lógica de cálculo dos dias úteis entre duas datas. Vamos detalhar os passos:

a. Recebimento da Data e Conversão

```
String data_request_calculo = dataVenctoTitulo;
DateTimeFormatter converter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
LocalDate data_convertida_dataVenctoTitulo = LocalDate.parse(data_request_calculo,
converter);
```

A data é recebida como uma string no formato "yyyy-MM-dd", comum em muitas aplicações. Usamos o DateTimeFormatter para garantir que a string seja corretamente convertida para um objeto do tipo LocalDate.

b. Obtendo a Data Atual

```
LocalDate data_hoje = LocalDate.now();
```

A data atual é capturada utilizando o método now() da classe LocalDate, o que nos permite comparar e calcular a diferença em dias úteis.

c. Lógica do Cálculo de Dias Úteis

```
long diasUteis = 0;

for (LocalDate date = data_hoje; date.isBefore(data_convertida_dataVenctoTitulo); date =
date.plusDays(1)) {
    DayOfWeek diaDaSemana = date.getDayOfWeek();
    if (diaDaSemana != DayOfWeek.SATURDAY && diaDaSemana != DayOfWeek.SUNDAY) {
        diasUteis++;
    }
}
```

Aqui, iniciamos o cálculo de dias úteis com uma variável diasUteis inicialmente zerada. Através de um loop, percorremos cada dia entre a data atual (data_hoje) e a data de vencimento (data_convertida_dataVenctoTitulo). A cada iteração, verificamos o dia da semana utilizando o método getDayOfWeek(). Se o dia não for sábado (SATURDAY) ou domingo (SUNDAY), o contador de dias úteis é incrementado.

Essa lógica é eficiente para excluir automaticamente finais de semana do cálculo.

d. Retorno do Resultado

```
int resultado_diasUteis = (int) diasUteis;
System.out.println("resultado_diasUteis: " + resultado_diasUteis);
return resultado_diasUteis;
```

Ao final do cálculo, o número total de dias úteis é impresso no console e retornado como resultado do método.

Testando a Implementação

Para testar o código, basta alterar a data de vencimento fornecida no método main. Isso permitirá calcular a diferença em dias úteis para diversas datas e verificar o comportamento do algoritmo.

Exemplo de saída para a data "2024-07-05":

```
---------------------------------
Data Operação ODM:     2024-10-09
Data Vencto do Titulo: 2024-07-05
resultado_diasUteis: 66
```

Neste caso, a data atual é 09 de outubro de 2024, e a data de vencimento é 05 de julho de 2024. O sistema calcula que há 66 dias úteis entre essas duas datas.

Conclusão

Este algoritmo fornece uma base sólida para o cálculo de dias úteis, ignorando automaticamente os finais de semana. Para uma aplicação mais robusta, pode ser interessante incluir a verificação de feriados nacionais e regionais, o que tornaria o cálculo mais preciso.

Por fim, a API de datas do Java é bastante poderosa e facilita muito a manipulação de datas complexas em sistemas corporativos e financeiros.

EducaCiência FastCode para a comunidade