01

Navegando entre tabs

Transcrição

Vamos entrar em um cenário diferente, onde temos um desenvolvedor fazendo parte do nosso time, e ele desenvolveu novas funcionalidades em nosso projeto, ou seja, um código que não sabemos como funciona. Vamos aprender a lidar com esse tipo de situação.

Acessando o *Project*, notamos que a estrutura é a mesma da qual já estávamos utilizando, mas foi adicionado um novo pacote chamado `registrador`.

Primeiro vamos dar uma olhada na nossa classe principal. Notamos que o código está bem diferentes do que fizemos anteriormente, temos instâncias de uma classe `Pagamento`.

```
package br.com.alura.bytebank;

import br.com.alura.bytebank.model.Pagamento;
import br.com.alura.bytebank.model.Tipo;
import br.com.alura.bytebank.registrador.RegistroDePagamento;

import java.util.Arrays;
import java.util.List;

public class Principal {
    public static void main(String[] args) {
        Pagamento almoco = new Pagamento(Tipo.DEBITO, 20, "Almoço no feijuca");
        Pagamento videogame = new Pagamento(Tipo.CREDITO, 1000, "PS4");

        List<Pagamento> pagamentos = Arrays.asList(almoco, videogame);

        RegistroDePagamento registrador = new RegistroDePagamento();
        registrador.registra(pagamentos);
    }
}
```

Executando, vemos que tudo funciona, e o resultado são algumas informações sobre pagamentos. Vamos acessar a classe `Pagamento`.

Poderíamos usar o atalho "Ctrl + n" e pesquisar pela classe `Pagamento`, mas apesar de ser fácil, existe uma maneira mais simples de fazer essa navegação. Podemos colocar o cursor do editor em cima da classe que queremos acessar, e usar o atalho "Ctrl + b". Com esse atalho, ele vai direto para a classe `Pagamento`.

```
package br.com.alura.bytebank.model;

import java.time.LocalDate;

public class Pagamento {

    private String descricao;
```

```
    private Tipo tipo;
    private double valor;
    private LocalDate data;

    public Pagamento(Tipo tipo, double valor, String descricao) {
        this.tipo = tipo;
        this.valor = valor;
        this.descricao = descricao;
        this.data = LocalDate.now();
    }

    public Tipo getTipo() {
        return tipo;
    }

    public double getValor() {
        return valor;
    }

    public void setValor(double valor) {
        this.valor = valor;
    }

    @Override
    public String toString() {
        return "Pagamento{" +
                "descricao='" + descricao + '\'' +
                ", tipo=" + tipo +
                ", valor=" + valor +
                ", data=" + data +
                '}';
    }
}
```

Podemos ver que a classe `Pagamento` contém os atributos `descricao`, `tipo`, `valor` e `data`. E analisando mais atentamente o atributo `tipo` vemos que ele do tipo de uma classe chamada `Tipo`. Podemos acessá-la da mesma forma com o atalho "Ctrl + b".

```
package br.com.alura.bytebank.model;

public enum Tipo {
    DEBITO, CREDITO, DINHEIRO
}
```

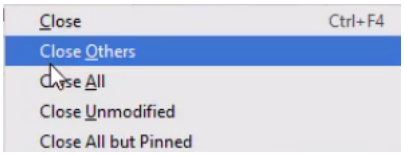
A classe `Tipo` é um **Enum**, e possui algumas constantes `DEBITO`, `CREDITO` e `DINHEIRO` para distinguir os tipos de pagamentos da aplicação.

Repare que agora que acessamos as classes `Principal`, `Pagamento` e `Tipo` apareceram três abas na parte superior do editor do código. Essas abas são chamadas de ***Tabs***. Todas as classes que acessarmos aparecerá também nas *Tabs*.

Você pode acessar as classes clicando com o mouse nas *Tabs*. Porém, o interessante é usar o atalho "Alt + Seta (Direita / Esquerda)". Dessa forma, é bem simples navegar entre as classes já abertas.

Configurando atalho

Já vimos como as classes `Pagamento` e `Tipo` funcionam, podemos fechá-las e deixar apenas a `Principal`. Para fechar as outras classes, pressionamos com o botão direito do mouse na classe `Principal` e selecionamos a opção ***Close Others***.

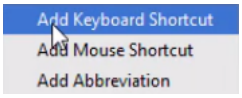


Você deve estar esperando o atalho para essa função, não é mesmo? Infelizmente o *IntelliJ* não possui um atalho para essa funcionalidade em específico.

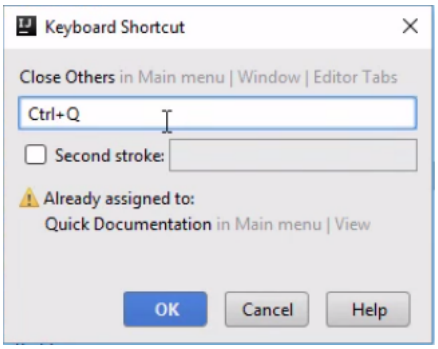
Como podemos fazer para usar um atalho para essa opção? Poderíamos acessar as configurações do *Keymap* usando o *Find Action*.

No *Keymap* podemos buscar pela ação *Close Others*. Podemos notar que o *Close Others* do contexto ***Editor Tabs*** não possui de fato, um atalho configurado.

Podemos clicar na linha com o botão direito do mouse e selecionar a opção ***Add Keyboard Shortcut***.



Agora basta colocar algum atalho de preferência e clicar em "OK" para associar a ação. Mas como saber se o atalho já está sendo usado ou não? Ao colocar um atalho, caso ele já esteja sendo usado, por exemplo "Alt + q", o próprio *IntelliJ* emite um alerta.



No nosso caso, essa função não é tão importante para associá-la a um atalho, por isso não vamos associar. Vale ressaltar que caso você julgue alguma ação ser importante ter um atalho, sempre verifique se ele não está sendo usado, evitando perder outras funções.

Vamos acessar novamente as classes `Principal`, `Pagamento` e `Tipo` para que elas apareçam nas *Tabs*. Caso queira fechar a *Tab* que você esteja acessando atualmente, isso pode ser feito com o atalho "Ctrl + F4". E se preferir fechar todas as classes abertas nas *Tabs*, podemos usar o atalho "Alt + Shift + x".

Acessando novamente a classe `Principal`, podemos verificar que além da instância da classe `Pagamento`, temos uma lista chamada `pagamentos` e uma classe chamada `RegistroDePagamento`.

Podemos acessar a classe `RegistroDePagamento` com o atalho "Ctrl + b". Acessando, vemos que é uma classe extremamente difícil de ser entendida. Veremos como o *IntelliJ* pode nos ajudar a organizar melhor esse código.