14

Conhecendo o find action

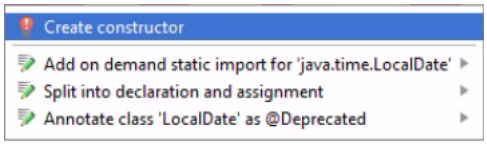
Transcrição

Criamos um funcionário `jose`, vamos criar um outro objeto da classe `Funcionario`. Mas antes de criarmos vamos ajustar um detalhe, que o de ficar chamando os *Setter* para inserir os valores nos atributos do objeto.

Seria mais fácil passarmos os valores pelo construtor. Vamos passar o valor `"José"`, `1` e `LocalDate.of(1990, 2, 10)` no construtor da classe `Funcionario`.

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

import br.com.alura.bytebank.model.Funcionario;

public class Principal {

    public static void main(String[] args) {

        System.out.println("Bem-vindo ao Bytebank");
        Funcionario jose = new Funcionario("José", 1, LocalDate.of(1990, 2, 10));

        jose.setNome("José");
        jose.setMatricula(1);
        jose.setDataNascimento(LocalDate.of(1990, 2, 10));

        System.out.println(jose);
    }
}
```

Agora podemos colocar o cursor do editor em qualquer argumento do construtor e usar o atalho "Alt + Enter", agora é só selecionar a opção ***Create constructor***



O *IntelliJ* de fato conseguiu criar o construtor, porém ele criou os parâmetros de uma forma não muito interessante, além de não conseguir associar os valores recebidos como parâmetros e os atributos da classe `Funcionario`. O construtor de `Funcionario` ficou assim:

```
// ...

public Funcionario(String josé, int i, LocalDate of) {
}

// ...
```

Nem sempre o *IntelliJ* vai fazer com que o comportamento que esperamos aconteça. Como poderíamos corrigir e ser mais objetivo? Podemos criar o construtor diretamente pela classe `Funcionario`.

Dentro de `Funcionario`, usamos o atalho "Alt + insert", selecionando a opção ***Constructor***. Na nova janela aberta ***Choose Fields to Initialize by Constructor*** podemos selecionar os atributos que queremos no construtor, vamos selecionar todos. Agora só clicar em "OK" para que o construtor seja criado.

```
// ...

public Funcionario(String nome, int matricula, LocalDate dataNascimento) {
    this.nome = nome;
    this.matricula = matricula;
    this.dataNascimento = dataNascimento;
}

// ...
```

Agora com um construtor correto, podemos remover as linhas que fazem uso de `jose.setNome()`, `jose.setMatricula()` e `jose.setDataNascimento()`.

Poderíamos selecionar as linhas que desejamos remover, apagar com as teclas de "Backspace" ou "Delete". Mas temos um atalho especifico para isso que é o "Ctrl + y". O atalho deleta as linhas selecionadas ou a linha em que o cursor está. Nesse caso, podemos selecionar todas as linhas que usam o *Setter* e o atalho "Ctrl + y".

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

import br.com.alura.bytebank.model.Funcionario;

public class Principal {

    public static void main(String[] args) {

        System.out.println("Bem-vindo ao Bytebank");
        Funcionario jose = new Funcionario("José", 1, LocalDate.of(1990, 2, 10));

        System.out.println(jose);
    }
}
```

Podemos agora criar um novo objeto da classe `Funcionario`, dessa vez a variável será chamada `maria`. Repare que vamos fazer exatamente o que fizemos com `jose`, com apenas algumas pequenas mudanças, como o próprio nome da variável do objeto e os valores dos atributos.

Então podemos utilizar o atalho "Ctrl + d", que serve para duplicar a linha que o cursor do editor está ou as linhas selecionadas.

Após duplicar a linha de instancia do funcionário `jose`, podemos alterar o nome da variável para `maria`, o valor do nome para `"Maria"`, matricula para `2` e a data de nascimento para `LocalDate.of(1991, 5, 15)`. Podemos também mostrar as suas informações com um `System.out.println(maria)`.

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

import br.com.alura.bytebank.model.Funcionario;

public class Principal {

    public static void main(String[] args) {

        System.out.println("Bem-vindo ao Bytebank");
        Funcionario jose = new Funcionario("José", 1, LocalDate.of(1990, 2, 10));
        Funcionario maria = new Funcionario("Maria", 2, LocalDate.of(1991, 5, 15));

        System.out.println(maria);
        System.out.println(jose);
    }
}
```

Acabamos colocando o `System.out.println(maria)` antes do `System.out.println(jose)`. Caso queira mostrar as informações dos funcionários na ordem de criação, em vez de copiar e colar a linha da `maria` depois da `jose`, podemos usar o atalho "Alt + Shit + Seta para baixo", para mover a linha da `maria` para baixo. O atalho também funciona com a "Seta para cima" do teclado, isso se desejar mover a linha para cima.

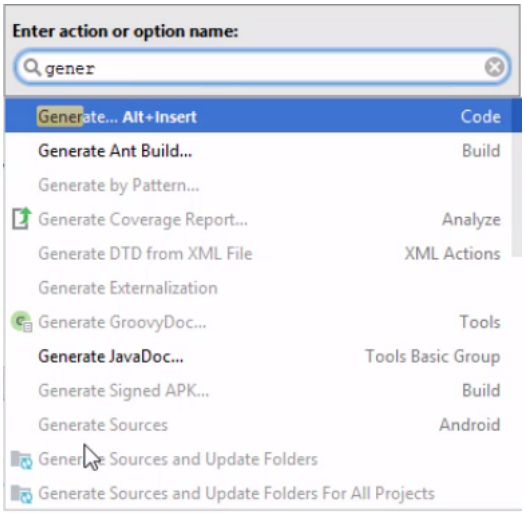
```
package br.com.alura.bytebank;

import br.com.alura.bytebank.model.Funcionario;

public class Principal {

    public static void main(String[] args) {

        System.out.println("Bem-vindo ao Bytebank");
        Funcionario jose = new Funcionario("José", 1, LocalDate.of(1990, 2, 10));
        Funcionario maria = new Funcionario("Maria", 2, LocalDate.of(1991, 5, 15));

        System.out.println(jose);
        System.out.println(maria);
    }
}
```

Podemos executar o programa com "Shift + F10", será mostrado no Console os valores dos dois objetos que criamos.

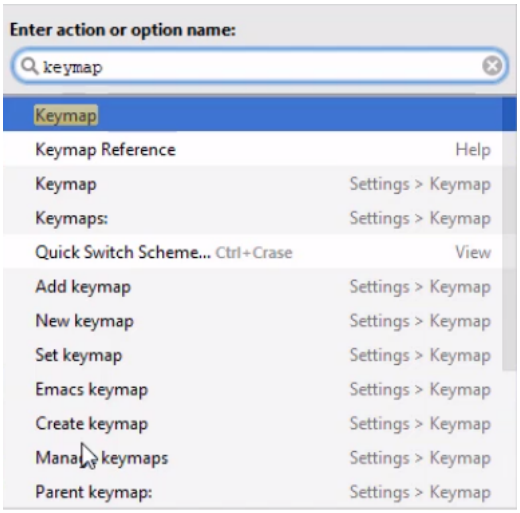
Aprendemos os primeiro passos com o *IntelliJ IDEA*. Mas ainda temos um atalho que é muito útil conhecido como ***Find Action***. O *Find Action* é um atalho que permite utilizar todos os recursos da IDE com um único atalho.

O atalho *Find Action* é o "Ctrl + Shift + a". Ao usar o atalho, um campo de busca aparecerá, onde podemos escrever o que desejamos da IDE.

Se procurarmos por "*generate*", o *Find Action* vai nos retornar tudo relacionado ao termo buscado, inclusive o próprio atalho "Alt + insert".



Também serve para configurações da própria IDE. Podemos buscar por "*keymap*" para que ele nos retorne o acesso a área de configuração do *Keymap*.



O *Find Action* é um ferramenta muito poderosa, que ajuda bastante quando precisamos acessar algo que não lembramos o caminho, atalhos e qualquer coisa que o *IntelliJ* possa oferecer. Um detalhe do *Find Action* é que, caso você queira buscar algo, será necessário fazer em inglês.