02

Menos erros, mais produtividade

Dominar frameworks e a linguagem de programação é importante, mas não é tudo. Um desenvolvedor Java passa grande parte do seu dia trabalhando na IDE de sua escolha e, portanto, uma boa utilização dessa ferramenta implica em menos desperdício de tempo.

Ainda que evitar usar o mouse traga um ganho consideravelmente pequeno por operação, pense em quantas vezes as repetimos durante um dia de trabalho: o tempo acumulado é considerável. Se somarmos a isso todos os imports errados que deixaremos de fazer e as facilidades para manter o código limpo, a diferença de produtividade é notável.

O Eclipse, como qualquer outra IDE, foi desenhada para auxiliar o desenvolvedor nas suas tarefas cotidianas e, nesse curso, veremos desde os atalhos mais básicos até customizações avançadas, passando, no caminho, pelas refatorações e pela tão útil perspectiva de debug.

Tudo o que é repetitivo, isto é, tudo o que pode ser feito por uma máquina deve ser feito por uma: o trabalho do desenvolvedor é fazer a parte criativa, a solução do problema do mundo real, e não ficar se preocupando com detalhes de sintaxe, atribuições, etc.

Por ser um curso de utilização de ferramenta, é importante que o aluno pratique intensamente o que for aprendido aqui atentando para escrever sempre o mínimo possível, abusar do uso dos atalhos e customizações e evitar o uso do mouse. Leve como um laboratório e pratique tudo o que for ensinado para, mais tarde, escolher sua forma favorita de trabalho na IDE.

Para começar, vejamos os pontos mais simples, mas de maior impacto no nosso dia-a-dia.

Abra a classe `Principal` no editor do Eclipse e maximize-a com ctrl + M. Dentro dela digite main e ctrl + espaço e note que ele gerará o método main completo para você:

```
public class Principal {

    public static void main(String[] args) {

    }
}
```

Criemos, então, um objeto do tipo `Funcionario` , sempre começando pela parte à direita do igual. Essa regra vale para todas as atribuições a variáveis, abuse dela! Escreva, dentro do `main` fazendo `new Fun` e navegando entre os parâmetros do construtor chamado com tab.

```
public class Principal {

    public static void main(String[] args) {
        new Funcionario("José", 25, dataNascimento);
    }
}
```

A data de nascimento, contudo, é do tipo `java.util.Calendar` e, portanto, teremos que fazer o import do tipo e declarar a variável, certo? Felizmente, por conta da declaração no construtor do Funcionario, o Eclipse já sabe o tipo certo para essa variável! Basta, em qualquer lugar da linha da variável `dataNascimento` , fazer o ctrl + 1 e escolher Create local variable 'dataNascimento': ele importará o `java.util.Calendar` e criará a variável dataNascimento, sem inicializá-la.

Use o esc ou o enter para sair da seleção (os contornos das variáveis).

```
    public static void main(String[] args) {
        Calendar dataNascimento;
        new Funcionario("José", 25, dataNascimento);
    }
```

Inicializemos a variável adicionando `= new Greg` e selecionando o construtor que recebe ano, mês e dia. Note, de novo, que o import é feito automaticamente e você pode navegar entre as variáveis contornadas com tab.

```
    public static void main(String[] args) {
        Calendar dataNascimento = new GregorianCalendar(1985, 3, 18);
```

```
        new Funcionario("José", 25, dataNascimento);
    }
```

Após cada edição que realizamos, é sempre recomendável que salvemos nosso trabalho, para evitar perdas em caso de algum problema com o computador ou uma queda de energia. Para salvar nossas alterações, pressione ctrl + S. Repita esse comando sempre que necessário.

Desça para a linha do `Funcionario` e atribua esse objeto a uma variável usando, novamente, o ctrl + 1 com Assign statement to new local variable e deixe o nome padrão:

```
    public static void main(String[] args) {
        Calendar dataNascimento = new GregorianCalendar(1985, 3, 18);
        Funcionario funcionario = new Funcionario("José", 25, dataNascimento);
    }
```

Finalmente, para terminar nosso primeiro método, imprima o funcionario com o auxílio do template que escreve o `System.out.println()` para você. Basta escrever `syso` e completar, passando o `funcionario` para o método. Não escreva a palavra `funcionario` : comece a digitar e já pressione ctrl + espaço. O resultado final será:

```
public static void main(String[] args) {
    Calendar dataNascimento = new GregorianCalendar(1985, 3, 18);
    Funcionario funcionario = new Funcionario("José", 25, dataNascimento);
    System.out.println(funcionario);
}
```

Para ver o resultado, execute o `main` usando o comando ctrl + F11 e preste atenção à View do Console.

```
Funcionario: José
```

Mas como isso aconteceu? A impressão do objeto foi mais bonita do que o usual porque o método `toString` da classe `Funcionario` está sobrescrito.

Para abrir a classe `Funcionario` , vá sobre o tipo (não a variável) e aperte F3. Ele abrirá a classe `Funcionario` para você e você verá a sobrescrita do método `toString` .

Agora, para alternar entre as abas, use ctrl + pgDown para ir para a direita, ou ctrl + pgUp para ir para a esquerda. Experimente esses atalhos.

Para saber mais

Na versão do Eclipse para Mac OS boa parte dos atalhos são diferentes. Os atalhos equivalentes aos apresentados nessa seção na versao Mac OS são: Maximizar ou minimizar: ctrl + M Quick Fix: command + 1 Ir para a declaração da classe: fn + F3 Navegar entre abas: ctrl + fn + cima e ctrl + fn + baixo