DANIEL ABELLA
www.daniel-abella.com

Daniel Abella

Profissional PMP, PMI-ACP e CSM, Gerente de Projetos no Virtus & UniFacisa e Professor do Curso de Sistemas de Informação da UniFacisa. Autor dos livros Gestão A3 (Ágil) e Scrum Arretado.

Conteúdo:

Capítulo 4

Eclipse IDE

Livro Java do Zero (Uma Viagem ao Mundo Java)

01 Introdução

- A IDE, em resumo, é basicamente a ferramenta de edição de código fonte, um "Word" da programação, oferecendo um conjunto de recursos projetados para auxiliar os programadores no desenvolvimento de software.
- Nesta seção, apresentaremos umas das IDEs mais usadas para desenvolvimento na linguagem Java, a IDE Eclipse.

História da IDE Eclipse

O Eclipse é uma IDE de código aberto desenvolvida inicialmente pela IBM no final dos anos 1990 como o projeto interno "VisualAge for Java". Em 2001, a IBM lançou o código-fonte como um projeto de código aberto chamado Eclipse, permitindo que desenvolvedores de todo o mundo contribuíssem para sua evolução. Desde então, o Eclipse se tornou uma das IDEs mais populares para o desenvolvimento Java, oferecendo recursos avançados, flexibilidade e personalização, além de servir como base para outras IDEs e frameworks importantes, como o Spring Tools Suite.

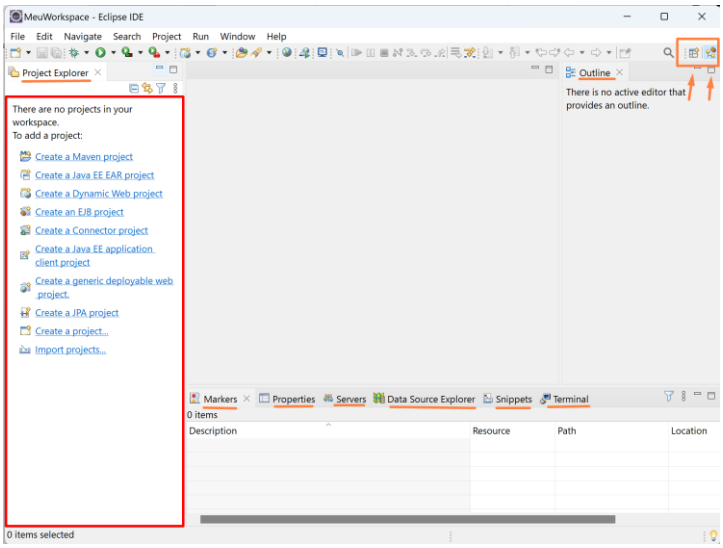
02 Workspace

- Workspace, em tradução literal, espaço de trabalho, é a pasta que vai abrigar todas as pastas dos projetos Java criados na IDE Eclipse. Assim que o Eclipse é iniciado, o Workspace deve ser selecionado.

- O Eclipse permite o conceito de múltiplos workspaces. Para criar um novo Workspace, basta informar um diretório na caixa de texto ou eleger um diretório clicando em Browse... . Agora, todos os novos projetos criados, estarão neste diretório associado ao workspace.
- Complementarmente, para abrir a IDE em um workspace existente, o procedimento é o mesmo, bastando informar um diretório na caixa de texto ou eleger um diretório clicando em Browse... . Se, este diretório informado já tiver sido usado como workspace, o Eclipse reconhece automaticamente.
- Quando, na tela de seleção de workspace é informado um, o Eclipse abre carregado com todos os projetos que estão dentro.

03 Perspectives e Views (Visões)

- Ao abrir algum Workspace, uma tela parecida como a seguir será apresentada. Note que, temos uma série de abas (todas destacadas por um sublinhado). Tais abas na terminologia do Eclipse se chamam Visões (Views).
- Perspectivas é um conjunto de views para um determinado propósito. A perspectiva em funcionamento do eclipse é a (referente a Java EE), observado em função do plano de fundo do ícone estar preenchido de azul claro. Para escolher uma outra perspectiva, basta clicar no botão .



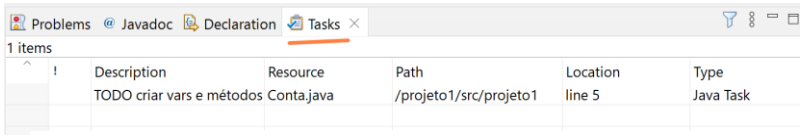
04 View Task

- A visão Task ajuda a rastrear comentários que iniciam com **//TODO**, que são usados comumente para lembrar ao desenvolvedor sobre alguma tarefa a ser feita, conforme exemplo abaixo.

Conta.java

```
1  package projeto1;
2
3  public class Conta {
4
5      //TODO criar vars e métodos
6  }
7
```

- Para verificar todos os comentários iniciados com //TODO, devemos ir ao menu Window, clicamos no submenu Show View e escolher a opção Task. A view com todos os comentários //TODO será a seguinte:



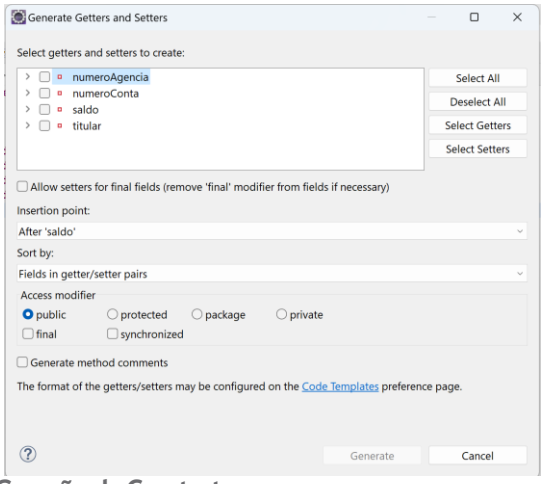
Boa Prática no Eclipse

Imagine um Workspace com diversos projetos abertos. Este cenário pode culminar em um alto uso de memória do computador, tornando-o lento. Neste sentido, o Eclipse permite você fechar projetos que não estão em uso. Fechar o torna momentaneamente indisponível, não apaga absolutamente nada. Para isto, botão direito no projeto a ser fechado e clique em Close Project. Quando necessitar usá-lo novamente, basta botão direito no projeto e Open Project.

05 Geração de Código

Geração de Getters e Setters

- Para gerar os getters e setters, basta realizar uma das operações:
- Atalho Alt + Shift + S e selecionar Generate Getters and Setters... ; ou
- Botão direito na área em branca da classe, clica no menu Source e depois no submenu Generate Getters and Setters... .
- Ao clicar em Generate Getters and Setters... , a tela a seguir é apresentada. Caso queira gerar getters e setters para todos os atributos, basta clicar em Select All e confirmar no botão Generate. Pronto, gerado 😊



Geração de Construtores

- Para gerar um construtor, basta realizar uma das operações:
- Atalho Alt + Shift + S e selecionar Generate Constructor using Fields... ; ou
- Botão direito na área em branca da classe, clica no menu Source e depois no submenu Generate Constructor using Fields... .
- Ao clicar em Generate Constructor using Fields... , uma tela é apresentada, devendo eleger quais campos serão usados para gerar o construtor. Uma vez escolhidos, clique no botão . Pronto, gerado 😊.
- PS: Se você quiser 1 construtor com 4 campos e 1 construtor com 3 campos, precisamos repetir os procedimentos supracitados duas vezes, um para cada construtor.

Geração de Construtor Padrão (Default)

- Para gerar um construtor padrão (default), basta realizar as operações descritas acima e, não eleger nenhum campo e clicar no botão Generate . Alternativamente, podemos na classe, na área útil dentro da classe, fazer o comando Ctrl + SPACE selecionar a opção Conta() - Constructor usando clique ou a tecla . Enter

Outras Opções

- Ao executar o atalho Alt + Shift + S , podemos temos algumas operações interessantes de geração:
- A opção Generate hashCode() and equals()... gera os métodos hashCode e equals baseado nas variáveis selecionadas; e
- A opção Generate toString()... gera o método toString com base nas variáveis selecionadas.

Templates de Código

Existe nesta IDE a possibilidade de gerar alguns códigos baseados em um *template* pré-definido. Por exemplo, se em uma classe criada, for digitado o texto **main**, seguido do comando Ctrl + Space + Enter , será gerado o método **main** todinho para você. A seguir, alguns *templates* de código que podem ser uma mão na roda na sua vida profissional:

- main
- syso
- switch
- for
- foreach

06 Usando Dependências Externas

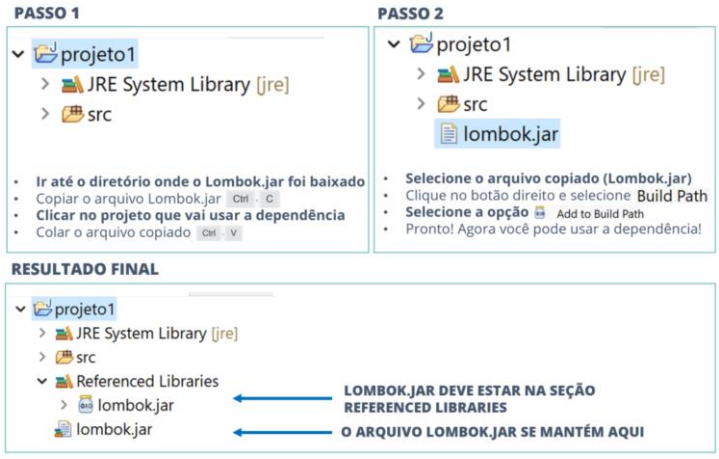
- Para explicar como usar Dependências Externas, usaremos como exemplo o Lombok, que ajuda a remover Boilerplate.

Instalação do Lombok

Para usar o Lombok, precisamos instalá-lo para poder que ele seja reconhecido na IDE (Eclipse ou qualquer outra). Faça um duplo clique no arquivo lombok.jar baixado e a tela a seguir será apresentada.

Caso o Lombok detecte automaticamente a sua IDE (indicado no quadrado abaixo), basta apenas clicar no botão "Install / Update" e depois em "Quit Installer". Caso não tenha detectado, clique em "Specify Location" e informe a pasta raiz da sua IDE. Posteriormente, basta apenas clicar no botão "Install / Update" e depois em "Quit Installer". Este passo é particular ao Lombok, não acontece com 99% das outras dependências.

ADICIONAR DEPENDÊNCIAS EXTERNAS NO ECLIPSE



Show Local History

Imagine a situação em que, você escreveu uma classe Conta e estava tudo funcionando adequadamente. E, depois você fez algumas alterações e de repente, passa a apresentar problemas. No Eclipse, existe um histórico de todos os arquivos. Ou seja, **toda vez que você salva um arquivo do seu projeto, este é colocado em um histórico interno**, nada relacionado a controle de versões como Git. Para acessar este histórico interno, com a classe aberta, clique com botão direito sobre a área em branco, clique na opção Team e depois em Show Local History. Uma view chamada History será apresentada com todo o histórico deste arquivo, que pode ser qualquer recurso do projeto. Ainda vai salvar você de alguma bronca, não esqueça!

07 Teclas de Atalho

TECLAS DE ATALHO	AÇÃO
Ctrl + Shift + R	Abrir um Resource (Recurso)
Ctrl + PgDown Ctrl + PgUp	Navegar entre as Abas dos Arquivos Abertos
Ctrl + W Ctrl + Swift + W	Fechar Window (Janela) ou Todas as Janelas
Ctrl + O Ctrl + O + O	Exibe os métodos da classe atual. Se repetir este comando, exibe a hierarquia dos métodos.
Ctrl + 1	Correção rápida
Ctrl + Shift + F	Formatar um Código
Ctrl + SPACE	Code complete
Ctrl + Shift + O	Organize (Organizar) Imports (Coloca os imports necessários e retira os não usados)
Ctrl + /	Comenta linha atual
Alt + Shift + Z	Zeta Project. Envolve código com try/catch

TECLAS DE ATALHO	AÇÃO
F3	Mostra a declaração da variável
Ctrl + L	Direciona para determinada linha no código
Ctrl + D	Delete (exclui) a linha atual
Ctrl + M	Maximize/Minimize aba atual
Ctrl + F11 Alt + Shift + X + J	Executar (X = eXecution, J = Java Class)
Ctrl + Shift + I	Debug: Executa a função Inspect
F5 Step Into	Debug: Para na primeira linha de código do método que estamos executando
F6 Step Over	Debug: Segue para a próxima linha de código
F7 Step Return	Debug: Retorna a linha seguinte do método que chamou o método em depuração
F8 Resume	Debug: Segue com a execução até o próximo breakpoint (ou fim da execução)

08 Refactoring

- Refactoring é uma prática no Desenvolvimento de Software, que envolve a reestruturação do código existente para melhorar sua qualidade, legibilidade, manutenção e desempenho, sem alterar seu comportamento externo.
- A função mais importante é a de renomear nomes de pacotes, variáveis, classes e variáveis. Para isto, basta selecionar o que você quer mudar e fazer a seguinte operação: Alt + Shift + R + R .Esta opção renomeia, bem como altera onde está sendo usado. É uma mão na roda!
- Se você estiver alterando uma variável de instância, perceba que, ao executar o atalho acima, aparecerá a opção de, ao renomear a variável, também ajustar o getter e setter relacionado.
- Todas as opções de Refactoring, estão no menu Refactoring da própria IDE.

09 Depuração de Código

- A depuração de código, também conhecida como debugging, se refere a um processo essencial durante o Desenvolvimento de Software, envolvendo a identificação e correção de erros (bugs) no código para garantir que o programa funcione conforme o esperado.
- A depuração permite que você pause a execução do programa em pontos específicos e inspecione o estado do código em tempo de execução, como os valores das variáveis, fluxo de execução, pilha de chamadas, entre outros. Ainda é possível acompanhar a execução do programa através de pontos de interrupção definidos.
- A seguir, temos o processo sugerido dos 4 passos para o processo de debbuging.

- Para este processo, as seguintes teclas de atalho podem ser extremamente úteis.

TECLAS DE ATALHO	AÇÃO
Ctrl + Shift + I	Inspect - Permite inspecionar o valor de uma variável em um ponto de interrupção ou durante a execução passo a passo.
F5 Step Into	Step Into - Permite entrar em um método, avançando uma linha de cada vez. Se houver uma chamada de método na linha atual, o depurador entrará nesse método.
F6 Step Over	Step Over - Permite avançar para a próxima linha de execução. Se houver uma chamada de método na linha atual, o depurador executará esse método completamente sem entrar nele.
F7 Step Return	Step Return - Permite retornar ao chamador anterior. Isso é útil quando você deseja sair de um método e retornar ao método que o chamou.
F8 Resume	Resume - Permite retomar a execução normal do programa após um ponto de interrupção, até atingir o próximo ponto de interrupção ou o término do programa.
Ctrl + Shift + B	Toggle Breakpoint - Permite definir ou remover um ponto de interrupção (breakpoint) em uma linha de código. Um ponto de interrupção é um local onde você deseja que a execução do programa seja pausada durante a depuração.
Ctrl + Shift + D	Display - Permite exibir o valor de uma expressão ou variável durante a depuração, mesmo sem um ponto de interrupção.

Um excelente recurso não detalhado neste artefato, mas descrito no livro é o *Working Set*, que é como se fosse uma pasta que agrupa projetos dentro de um Workspace.

Working Sets permitem agrupar e exibir apenas os recursos relevantes de um projeto, como arquivos e pastas, em vez de mostrar toda a estrutura do projeto, ajudando a reduzir a desorganização visual e simplificar a visualização do código, tornando mais fácil encontrar e trabalhar com os arquivos necessários.