

# Orientação a objetos



**Introdução: Orientação a Objetos**  
**Compreender compilação, execução e entrada de dados**

**Roni Schanuel**

**27-04-2023**

**Firjan SENAI**



- A linguagem Java foi criada em 1992 na Sun Microsystem.
- Em 2008 foi adquirida pela Oracle Corporation.

### **Características:**

- Orientada a Objetos;
- Portabilidade;
- Segurança;
- Linguagem Simples;
- Alta Performance;
- Interpretada;
- Multiplataforma;
- Fortemente Tipada;



**Java1:** ano 1996 -primeira versão estável da linguagem Java foi o JDK (Java Development Kit) 1.0.2, em janeiro de 1996 com o codinome Oak.

**Java2:** ano 1998 – Neste versão houve um grande aumento das classes na biblioteca Java (API) entre outras características como: J2SE (Java 2 Standard Edition), J2EE (Java 2 Enterprise Edition) e J2ME (Java 2 Micro Edition).

**Java3:** ano 2000 – Incorporação do Corba. Inclusão das bibliotecas JNDI, JavaSound entre outros.

**Java4:** ano 2002 – Inclusão de suporte a IPV6, XML, imagens e outros recursos.

**Java5:** ano 2004 – Uma das versões mais utilizadas. Inserção de recursos como: Enumeradores, Autoboxing, Generics, for-each entre outros.

**Java6:** ano 2006 - A partir desta versão, as siglas J2SE, J2EE e J2ME foram substituídas pelas siglas **Java SE**, Java EE e Java ME respectivamente. Esta versão apresenta melhorias na parte de segurança e desempenho da máquina virtual.

**Java7** ano 2011 – Algumas características importantes: permite o uso de strings em condições do switch, inferência na criação de objetos com tipos genéricos, uma biblioteca para tratar entrada e saída e melhorias nos streams para XML e Unicode.

**Java8** ano 2014 – Melhoria na performance, manipulação de data e expressões como Lamba

**Java9** ano 2017 - melhoria de desempenho às aplicações, jshell, api de suporte ao HTTP 2.0 entre outros.

**Java10** ano 2018 - Inferência de tipos para variáveis locais, Garbage-Collector Interface entre outras melhorias.

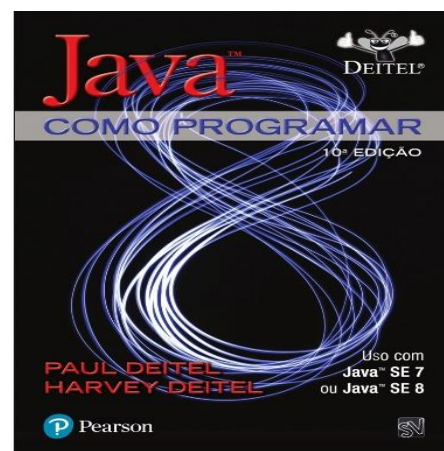
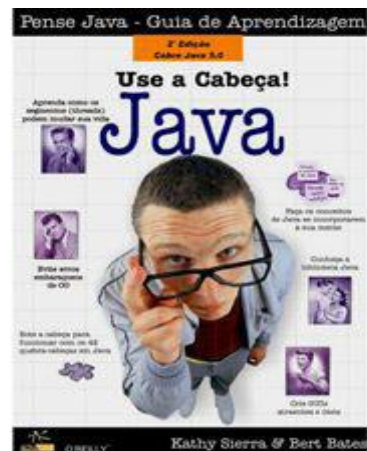
**Java11** ano 2018 – Anotações de tipo em expressões lambda, padronização do cliente HTTP

**Java12** ano 2019 – Novos métodos String, alterações de expressões no Switch, métodos transform entre outros.

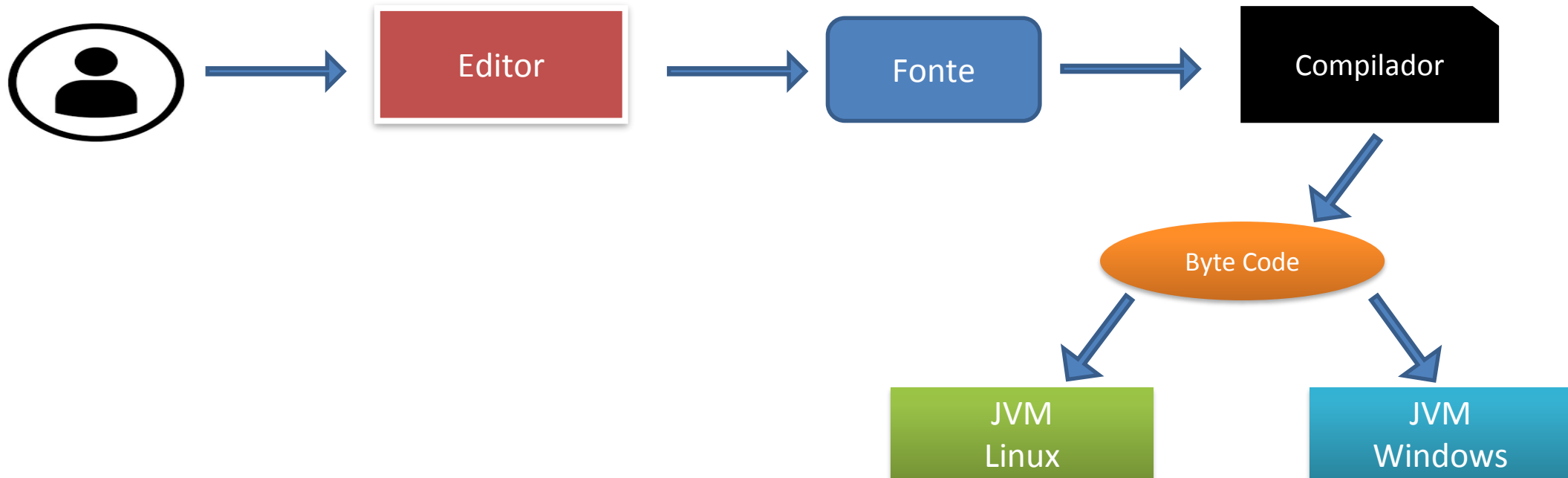
**Java 14** ano 2020 - Melhoria na utilização de alguns comandos como o instanceof, switch e outros.

**Java 17** ano 2021 - Implementação de classes seladas, atualizações e melhorias na linguagem.

**Java 18** ano 2022 - Melhorias de desempenho, estabilidade e segurança. UTF8 por padrão, Simple Web Server entre outros.



Um dos recursos do Java é a portabilidade do código gerado. Esta portabilidade é atingida através da utilização de bytecodes. Bytecode é um formato de código intermediário entre o código fonte, o texto que o programador consegue manipular, e o código de máquina, que o computador consegue executar. Na plataforma Java, o bytecode é interpretado por uma máquina virtual Java (JVM). A portabilidade do código Java é obtida à medida que máquinas virtuais Java estão disponíveis para diferentes plataformas. Assim, o código Java que foi compilado em uma máquina pode ser executado em qualquer máquina virtual Java, independentemente de qual seja o sistema operacional ou o processador que executa o código:



## **Java SE (Standard Edition)**

**JDK:** Java Developer's Kit, conjunto de ferramentas para desenvolvimento;

**JRE:** Java Runtime Environment, ambiente de interpretação e execução.

## **Java OPEN JDK**

O Java OPEN JDK é a versão free, no entanto, é preciso fazer atualizações sempre que uma nova versão for lançada. Caso não sejam feitas as atualizações, não serão mais feitas correções de bugs e nem instaladas novas funcionalidades que forem lançadas no programa.

## **Java LTS (Long Term Support)**

É uma versão paga. A empresa garante todas as atualizações para a versão usada em produção. Atualizações em seis meses.

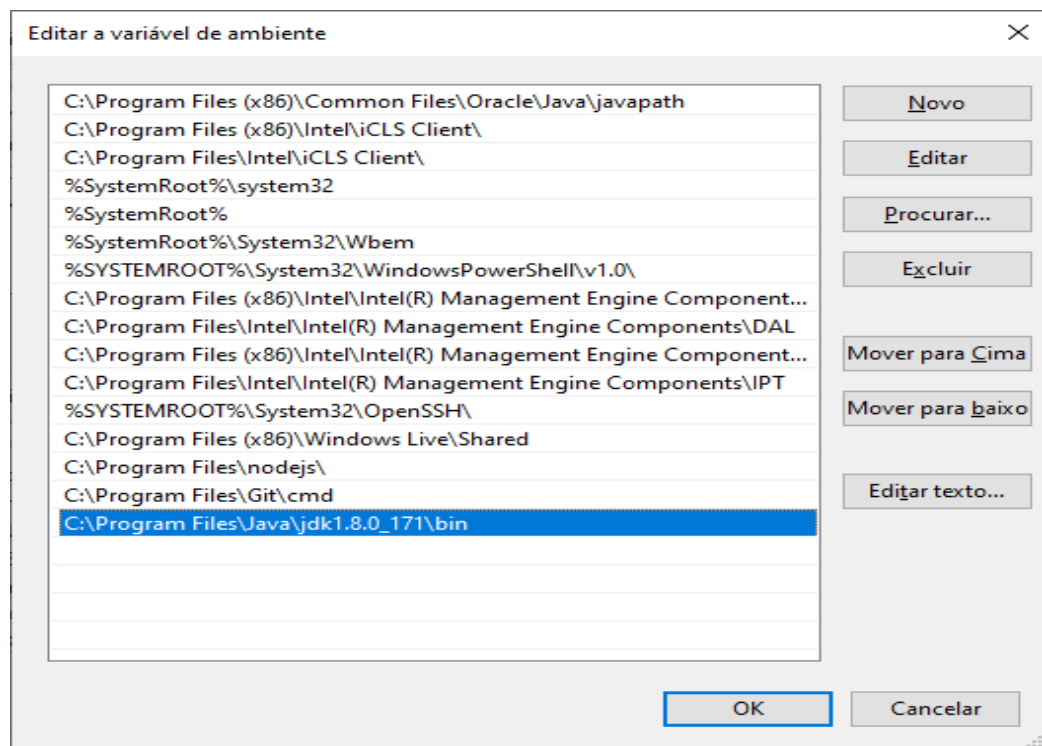
## **Algumas ferramentas do Java JDK**

- Compilador Java - javac
- Interpretador de aplicações Java - java
- Um gerador de documentação para programas - javadoc
- O manipulador de arquivos comprimidos - jar

O Path é uma variável de ambiente de um sistema operacional que fornece a uma aplicação uma lista de pastas onde procurar por arquivos executáveis.

A variável de ambiente CLASSPATH do Java é uma lista de locais que são visitados na procura por arquivos de classes, tanto o interpretador Java como o compilador Java usam a CLASSPATH

Na imagem abaixo é exibida a configuração do Path do Java no Windows





Uma aplicação Java deve ter pelo menos uma classe que contenha um método chamado main(), o qual contém o primeiro código a ser executado para iniciar a aplicação. Usando um editor de texto inserimos o código e salvamos o arquivo com o nome **Exemplo.java** Conforme exemplo abaixo:

O nome do arquivo deve coincidir com o nome da classe

```
public class Exemplo {  
    public static void main(String[] args){  
        System.out.println("Olá Mundo");  
    }  
}
```

Acesse o terminal do Windows ou Linux e execute os comandos abaixo:

O compilador converte arquivos-fonte Java em bytecodes com o comando **javac**

**javac Exemplo.java**

Como resultado teremos um arquivo bytecode com o mesmo nome do arquivo mas com a extensão **.class**:

O interpretador Java é chamado com o aplicativo java.exe. Ele é usado para interpretar o bytecode arquivo .class

Para execução basta digitar.

**java Exemplo**

O método `main()` é a primeira função que será executada no programa. Ele é **public** o que quer dizer que ele é visível globalmente, **void** porque não tem retorno, **static** o que significa que não precisamos criar objetos e também recebe um array de objetos do tipo **String**.

Quem chama o método **main** é o inicializador quando interpretamos o bytecode. O único argumento do método **main** serve para armazenar em cada entrada do array os parâmetros digitados pelo usuário após o nome da classe a ser interpretada.

Vamos alterar nossa classe **Exemplo** conforme abaixo e compilar e executar passando argumentos

```
public class Exemplo {  
    public static void main(String[] args){  
        System.out.println("Olá Mundo");  
        System.out.println(args[0]);  
        System.out.println(args[1]);  
        System.out.println(args[2]);  
    }  
}
```

**javac Exemplo.java**

**java Exemplo Celular TV Geladeira**

```
c:\teste>java Exemplo Celular TV Geladeira  
Olá Mundo ??  
Celular  
TV  
Geladeira
```

1) Crie uma classe no bloco de notas com o nome Exemplo2. Imprima seu nome em uma linha e sobrenome em outra linha usando o comando “System.out.print()”. Sabendo que os caracteres `\n` representam quebra de linhas.

2) Utilize os caracteres abaixo no Exemplo2 no lugar do `\n` para ver o resultado:

`\t`

`\b`

`\\`

`\'`

`\"`

É um ambiente de desenvolvimento integrado, combinando ferramentas, recursos que facilitam o desenvolvimento de aplicações.

## **Eclipse**

- Multiplataforma
- Suporte a Plugins
- Pacotes de desenvolvimento para Java Web e Desktop
- Muito utilizada no mercado

## **NetBeans**

- Possui suporte para criação de interfaces para aplicações web, desktop e mobile.
- Multiplataforma
- Muito utilizada em instituições de ensino

## **IntelliJ**

- Multiplataforma
- Possui um ótimo assistente de código
- Suporte nativo ao Kotlin
- Uso de plugins: É possível desenvolver em diferentes tecnologias com o IntelliJ (Python, Dart, etc) com o uso de plugins;

## **VsCode**

- Simples
- Diversas extensões disponíveis

Utilizaremos o Eclipse para desenvolvimento das aplicações em Java.

Link para download <https://www.eclipse.org/downloads/>

## Workspace

Workspace é o espaço físico onde você está trabalhando, ou seja, espaço em disco onde tudo do seu projeto será armazenado.

Eclipse trabalha sobre o conceito de workspaces múltiplos:

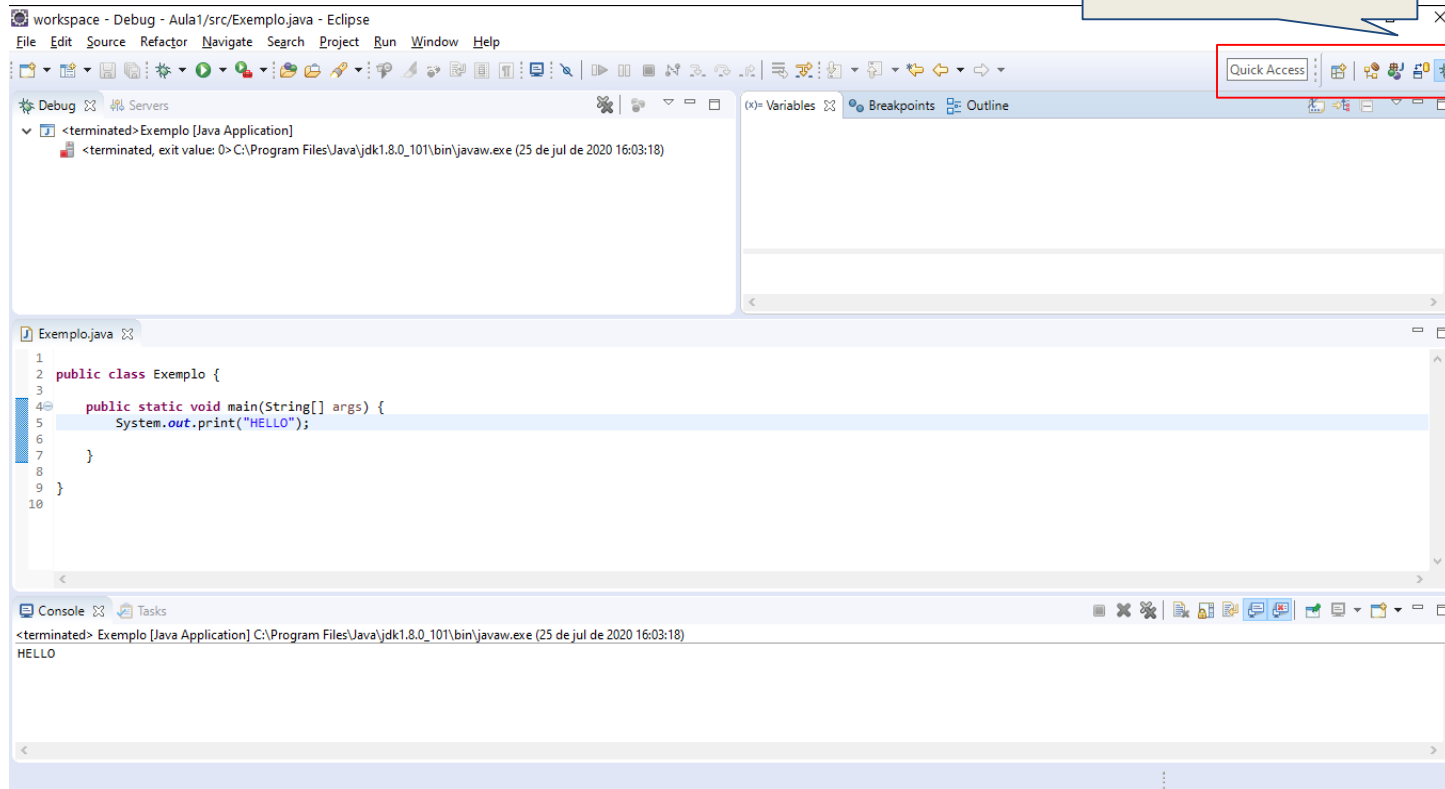
- ao criar um novo workspace, o mesmo é criado zerado
- pode-se alternar entre os workspaces: **File -> Switch Workspace -> Other**
- cada workspace possui uma pasta **.metadata** que armazena as configurações do mesmo.



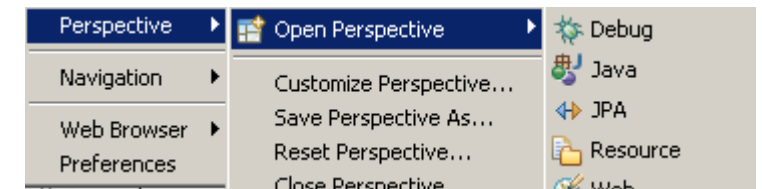
A perspectiva define quais e como surgem as visões que estão associadas.

Ex: Java EE, Debug, Java, Team Synchronizing.

Alterna entre as perspectivas e adiciona novas

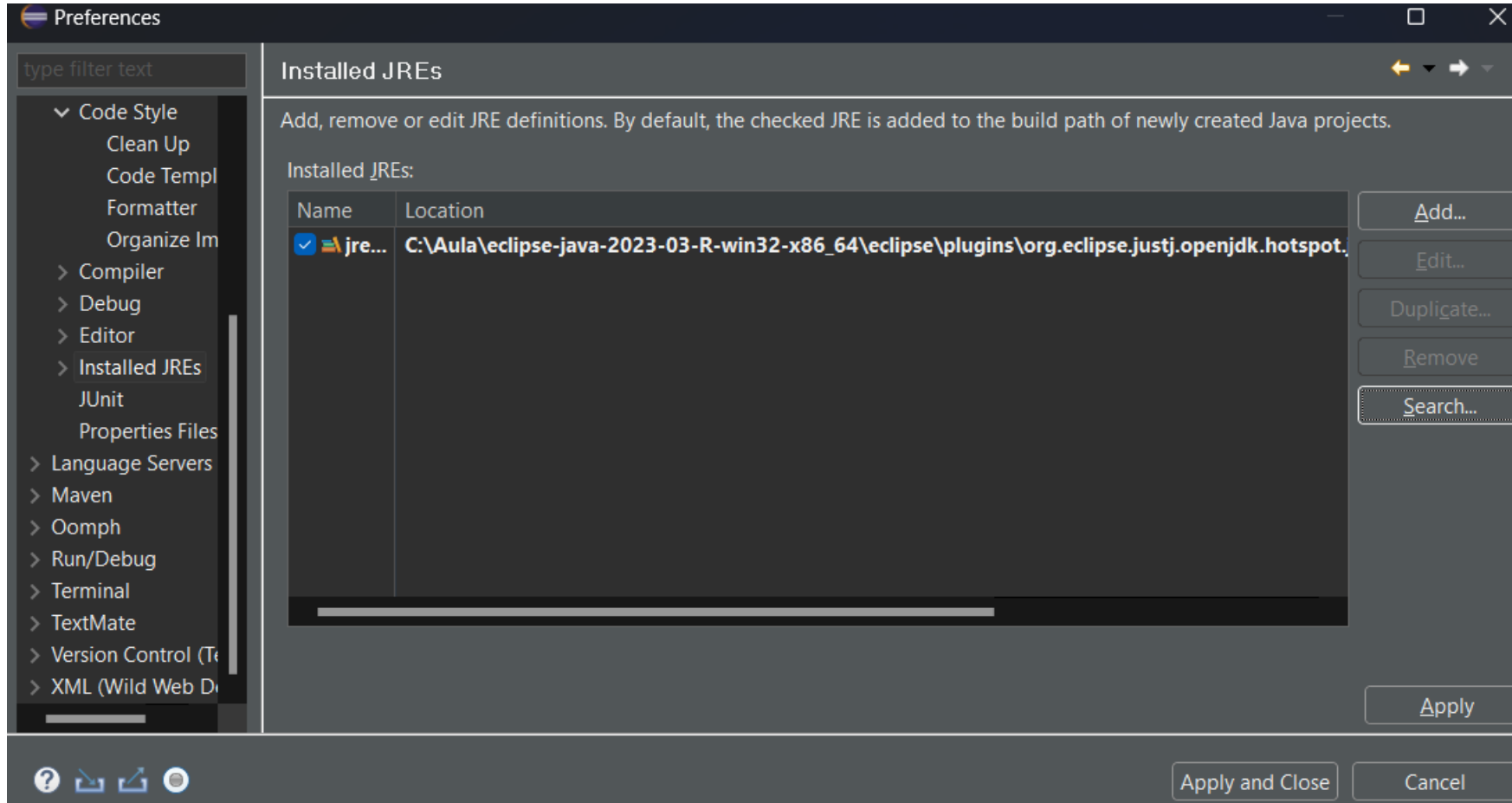


No menu - Window - Perspective  
Também é possível alterar a perspectiva



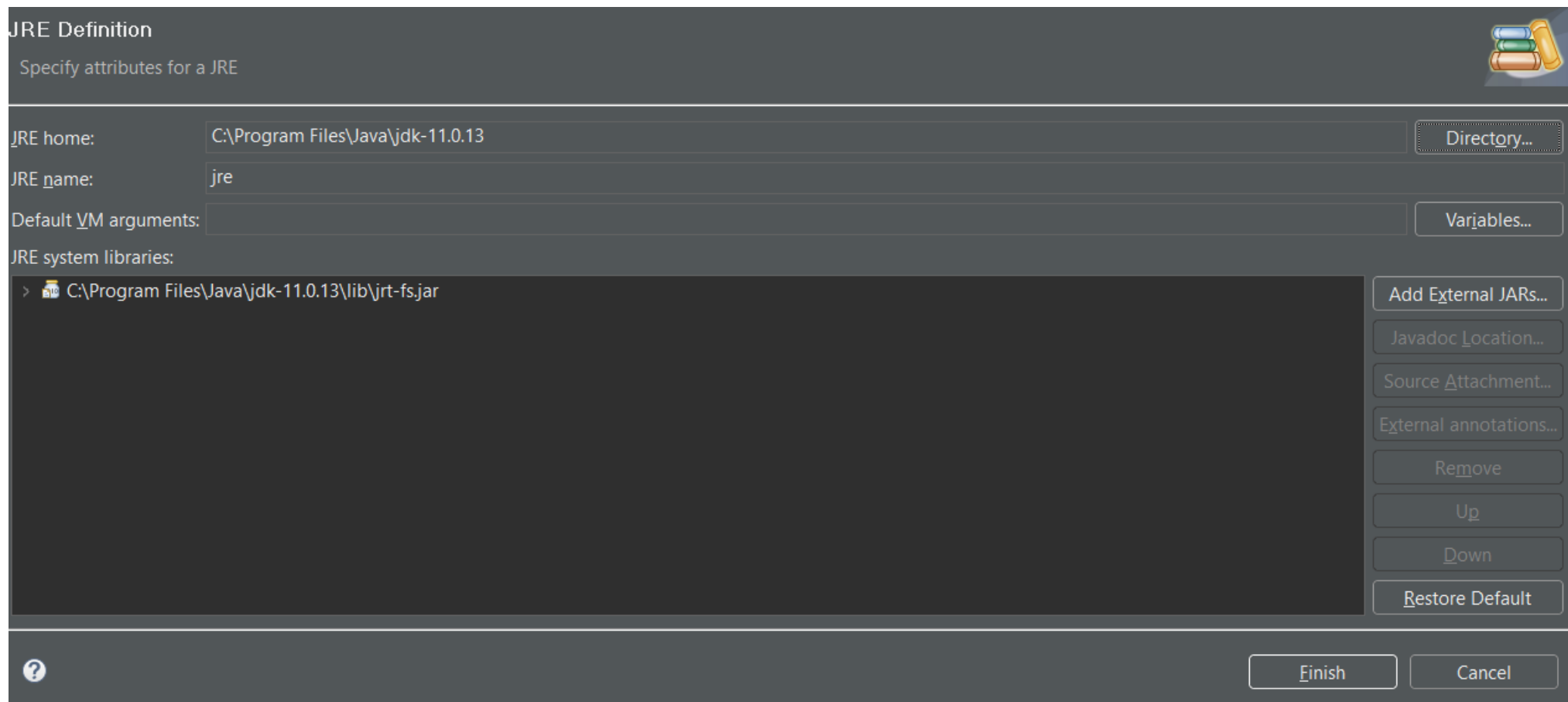
## Configurando o Java

Clique no menu Window – Preferences – Java – Installed JREs - Edit



## Configurando o Java

Altere para a JDK conforme a imagem abaixo, seguindo o caminho especificado da instalação da jdk.



JRE Definition

Specify attributes for a JRE

JRE home: C:\Program Files\Java\jdk-11.0.13 Directory...

JRE name: jre

Default VM arguments: Variables...

JRE system libraries:

- > C:\Program Files\Java\jdk-11.0.13\lib\jrt-fs.jar

Add External JARs...

Javadoc Location...

Source Attachment...

External annotations...

Remove

Up

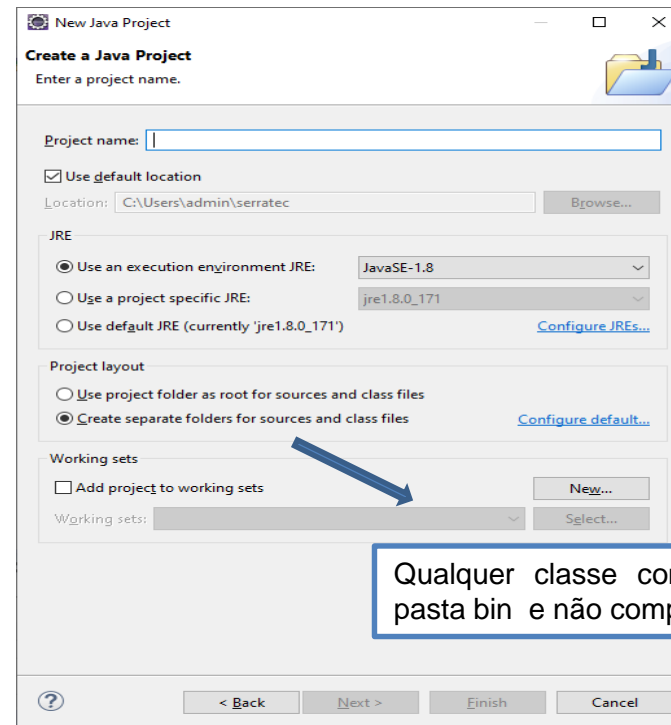
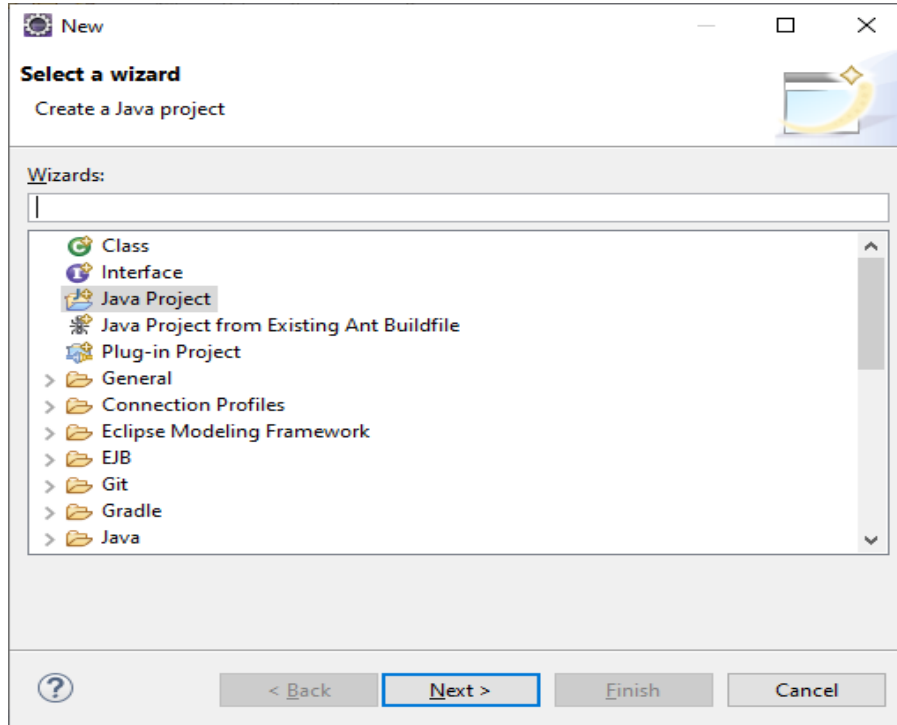
Down

Restore Default

? Finish Cancel

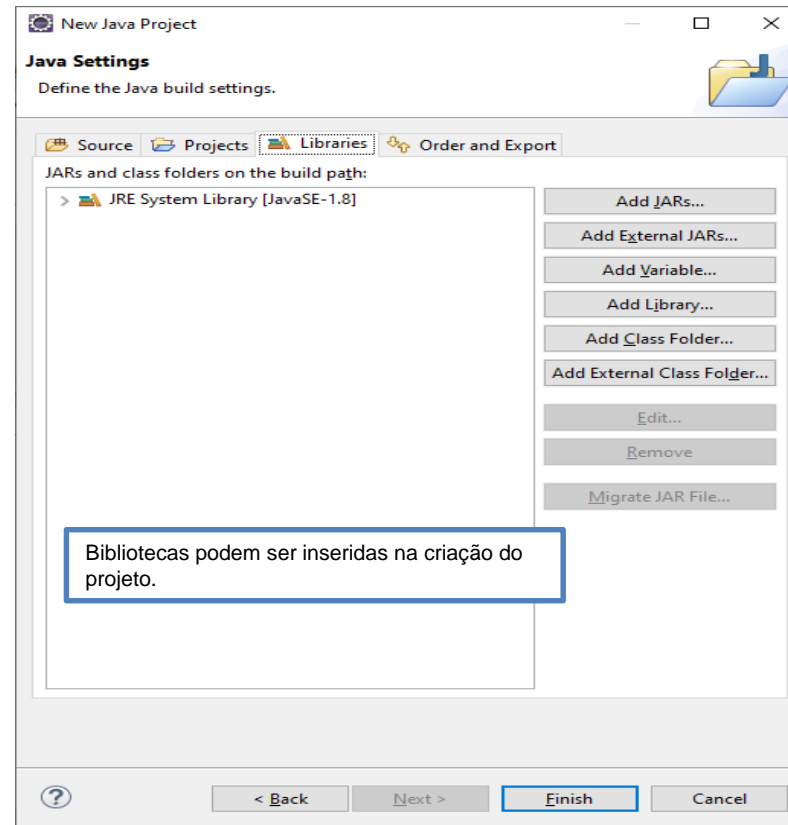
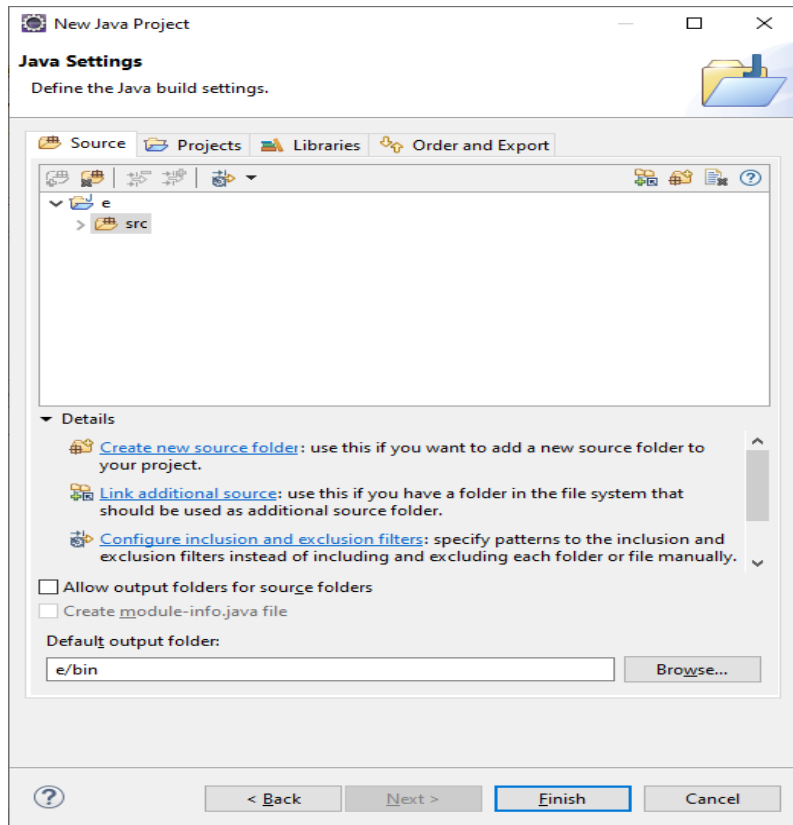


Para criar um novo projeto após abrir o eclipse utilize CTRL + N  
Selecione Java Project

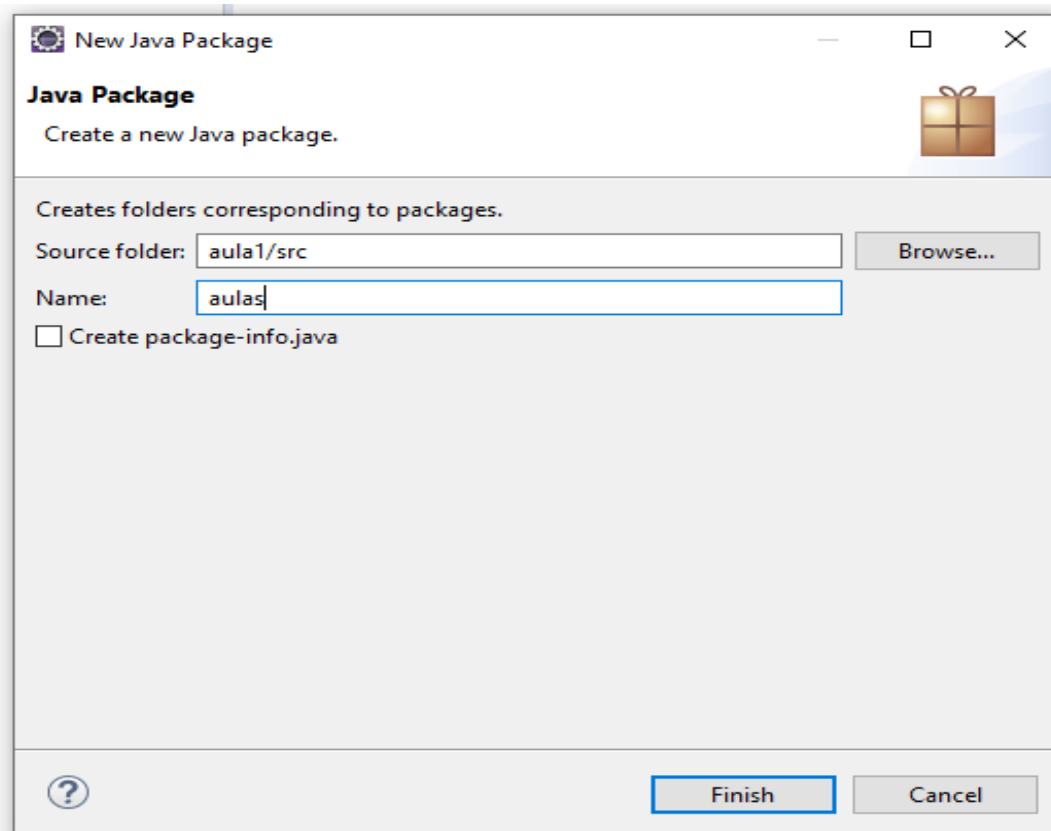
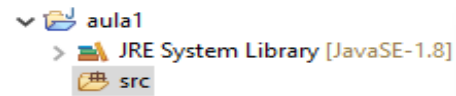


Qualquer classe compilada será armazenadas na pasta bin e não compiladas na pasta src.

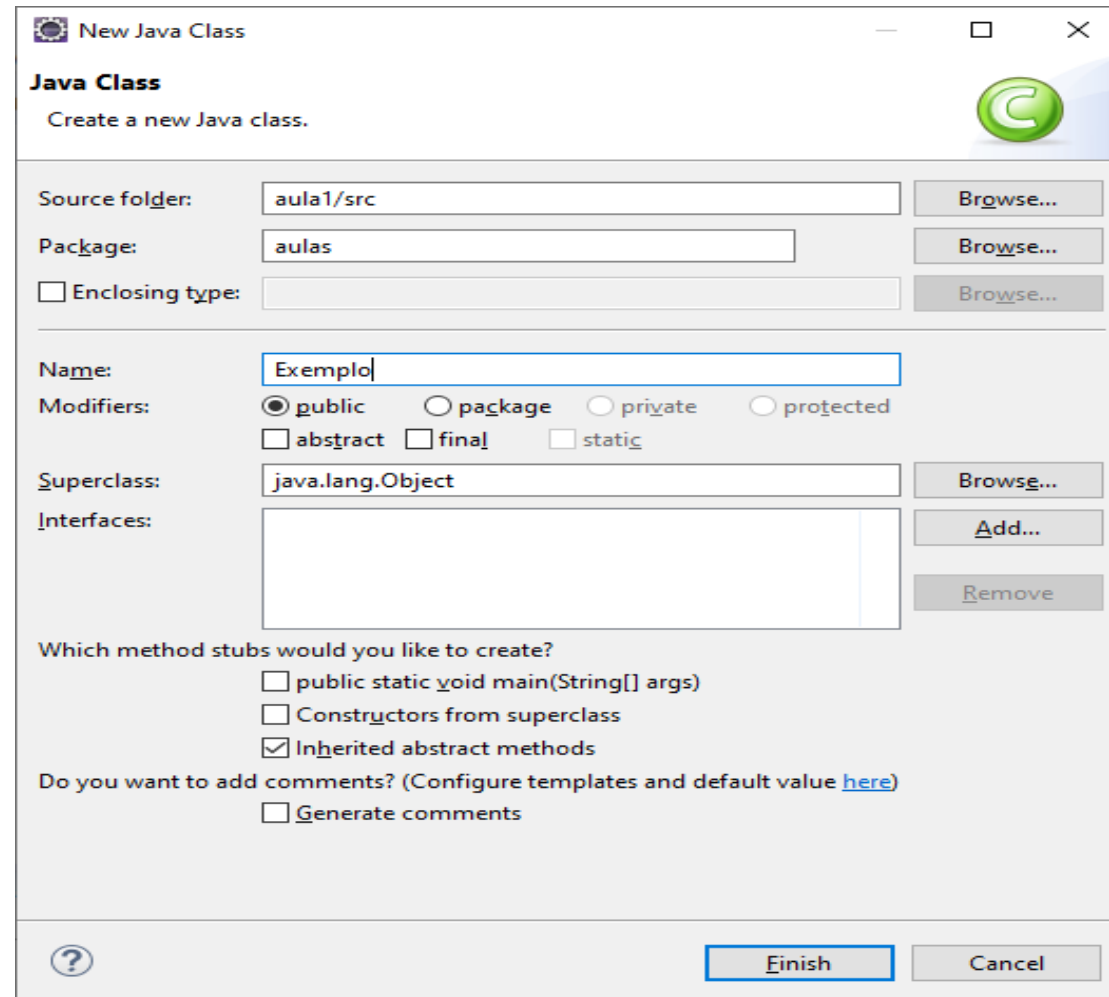
Depois de inserir o novo do projeto e clicar em next selecione a fonte do projeto. Por padrão a pasta src é a fonte, mas podemos criar outras pastas fontes mas em geral não fazemos modificação.



São utilizados para organizar as classes da sua aplicação e ajuda na reutilização de código.  
Para criar um pacote no eclipse – Botão direito no src – **New - Package**



Para criar uma classe no eclipse – Botão direito no pacote aulas **New - Class**



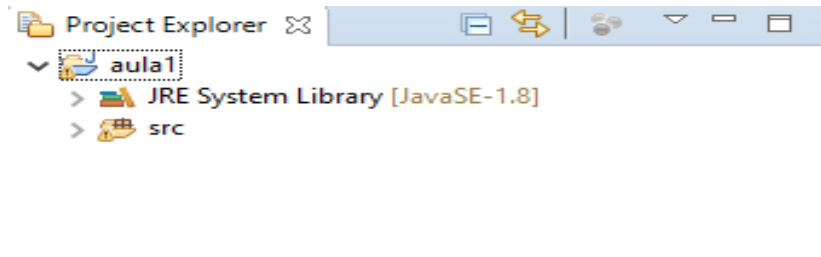
The screenshot shows the 'New Java Class' dialog box in the Eclipse IDE. The dialog has a title bar with the Eclipse logo and the text 'New Java Class'. Below the title bar, there's a section titled 'Java Class' with the instruction 'Create a new Java class.' and a green Eclipse logo. The main area of the dialog contains several fields and options:

- Source folder:** A text field containing 'aula1/src' and a 'Browse...' button.
- Package:** A text field containing 'aulas' and a 'Browse...' button.
- Enclosing type:** An unchecked checkbox followed by an empty text field and a 'Browse...' button.
- Name:** A text field containing 'Exemplo'.
- Modifiers:** A group of radio buttons for 'public' (selected), 'package', 'private', and 'protected'. Below them are checkboxes for 'abstract', 'final', and 'static'.
- Superclass:** A text field containing 'java.lang.Object' and a 'Browse...' button.
- Interfaces:** An empty list box with an 'Add...' button and a disabled 'Remove' button.
- Which method stubs would you like to create?** A section with three checkboxes: 'public static void main(String[] args)' (unchecked), 'Constructors from superclass' (unchecked), and 'Inherited abstract methods' (checked).
- Do you want to add comments? (Configure templates and default value [here](#))** A section with a checkbox for 'Generate comments' (unchecked).

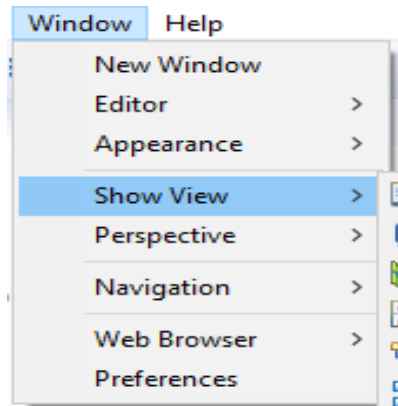
At the bottom of the dialog, there is a question mark icon on the left, and 'Finish' and 'Cancel' buttons on the right.

- **Pacotes:** ele deve ser escrito de forma semelhante a um endereço web, só que de trás para frente e ao final, indicamos um nome (ou um conjunto de nome), que classifica as classes agrupadas. (Ex.: “br.com.serratec.model”, “br.com.serratec.view”)
- **Classes e Interfaces:** nomes das classes e interfaces iniciam com uma letra maiúscula, sendo simples e descritivo. Caso seja nome composto utiliza-se o padrão *CamelCase*. (Ex.: “Usuario”, “ContaCorrente”)
- **Métodos:** os métodos seguem o mesmo padrão das classes, com a diferença que a primeira letra é minúscula. Como os métodos executam alguma ação, procure usar verbos para seu nome. (Ex.: “imprimirValor”, “executar”, “calcularMedia”)
- **Variáveis:** a convenção é a mesma adotada para métodos, com nomes curtos e significativos (ex.: “nome”, “nota”, “mediaAluno”). Evitar variáveis com apenas um caracter, a não ser que seja índice em repetições ou vetores (Ex.: “x”, “y”, “i”). Em constantes todas as letras deve estar em maiúsculas e separadas por “\_” (Ex.: “JUROS”, “DATA\_CORTE”).

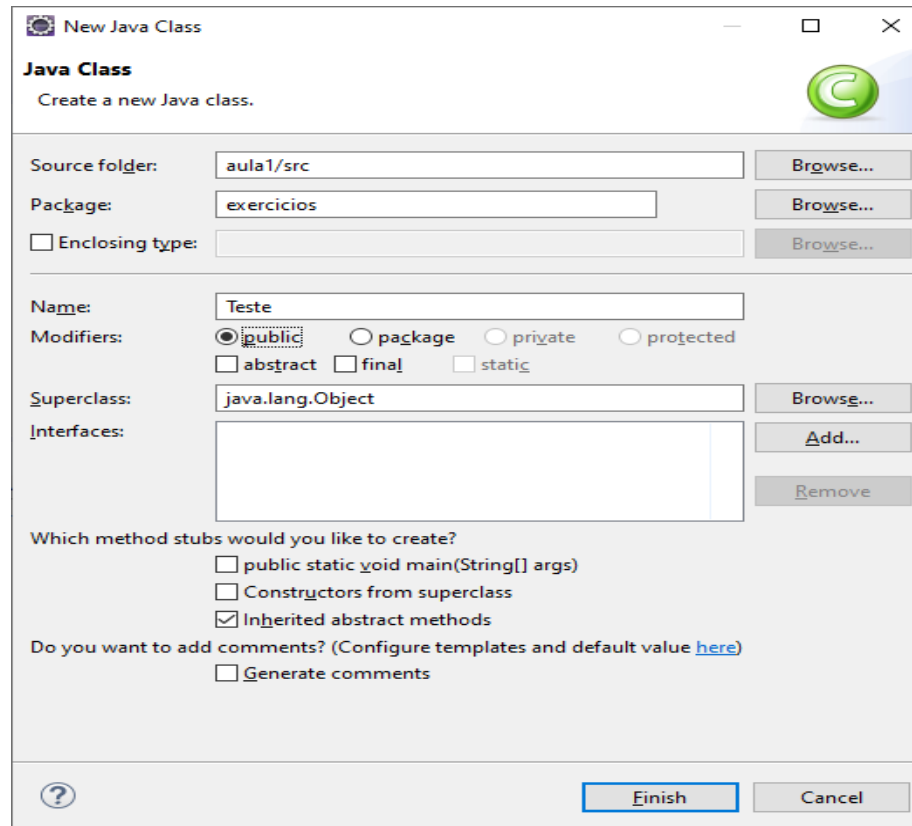
Exibe as pasta, pacotes do projeto e configurações da linguagem.



No menu inferior o Eclipse contém abas que são exibidas de acordo com o tipo de projeto criado. Essas abas podem ser customizadas no **Window -Show View**.



Podemos criar um pacote e a classe ao mesmo basta especificar o nome do pacote na criação da classe.



**New Java Class**

Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

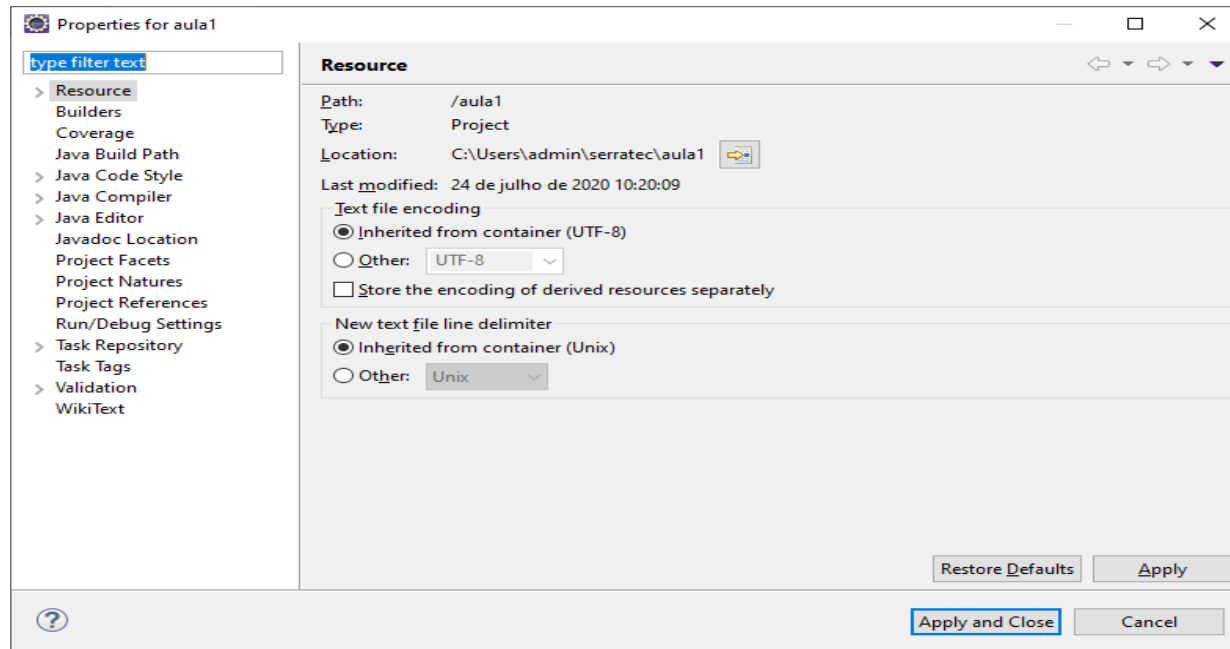
Which method stubs would you like to create?

☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))  
☐ Generate comments

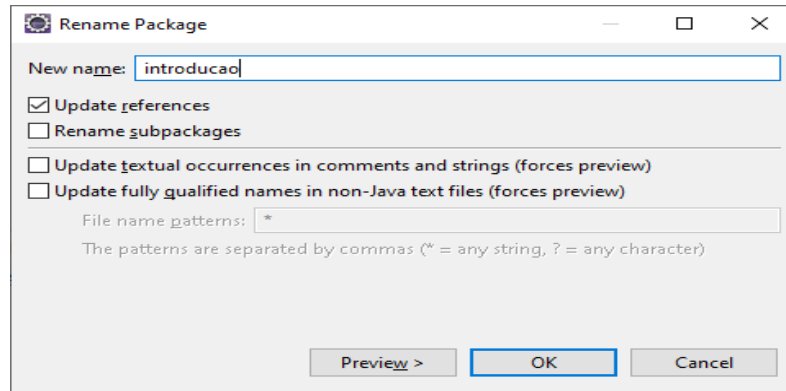
Clique com o botão direito sobre o projeto **Properties – Resources**

Nesta tela visualizamos algumas configurações e localização do projeto no sistema operacional.

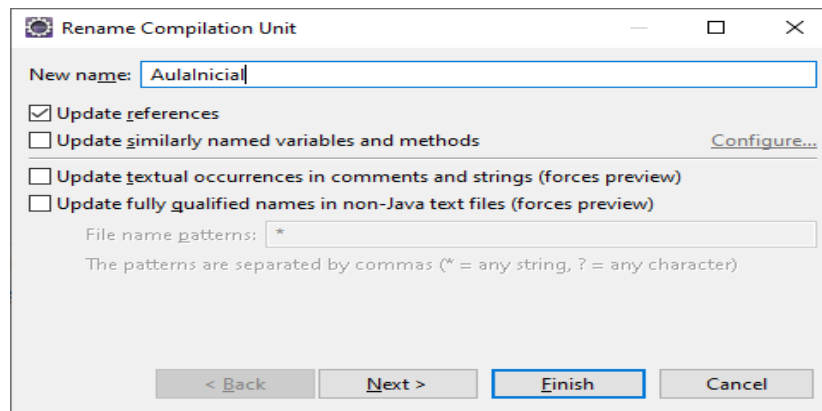




Para alterar o nome de um pacote, clique com o botão direito sobre o pacote **Refactor - Rename**



Para alterar o nome de uma classe, clique com o botão direito sobre a classe **Refactor - Rename**



Para renomear uma variável de forma simples, quando existem várias ocorrências, vamos utilizar o exemplo abaixo onde queremos alterar o nome da variável **valor** para **preco**.

Selecione a variável ou qualquer referência de **valor**, pressione **ALT+SHIFT+R** digite o novo nome e pronto as referências da variável foram modificadas.

```
public class Teste {  
    public static void main(String[] args) {  
        int valor = 100;  
        valor = 200;  
        System.out.println(valor);  
    }  
}
```

```
public class Teste {  
    public static void main(String[] args) {  
        int preco = 100;  
        preco = 200;  
        Sys Press Enter to refactor. Options... ▼  
    }  
}
```

Primeiro exemplo no Eclipse

Para execução pressione CTRL + F11

```
package aulas;

public class Exemplo {
    public static void main(String[] args) {
        System.out.println("Hello World !!");
    }
}
```

Segundo exemplo no Eclipse Criar uma classe com o nome **Exemplo2**

**Utilizando os atalhos**

**CTRL + Barra de espaço** – Completa determinado comando ou trecho de código.

**main + Barra de espaço** – Insere o método main

**syso + Barra de espaço** – Insere System.out.println

```
package aulas;

public class Exemplo2 {
    public static void main(String[] args) {
        System.out.println("Programação Java 1");
    }
}
```

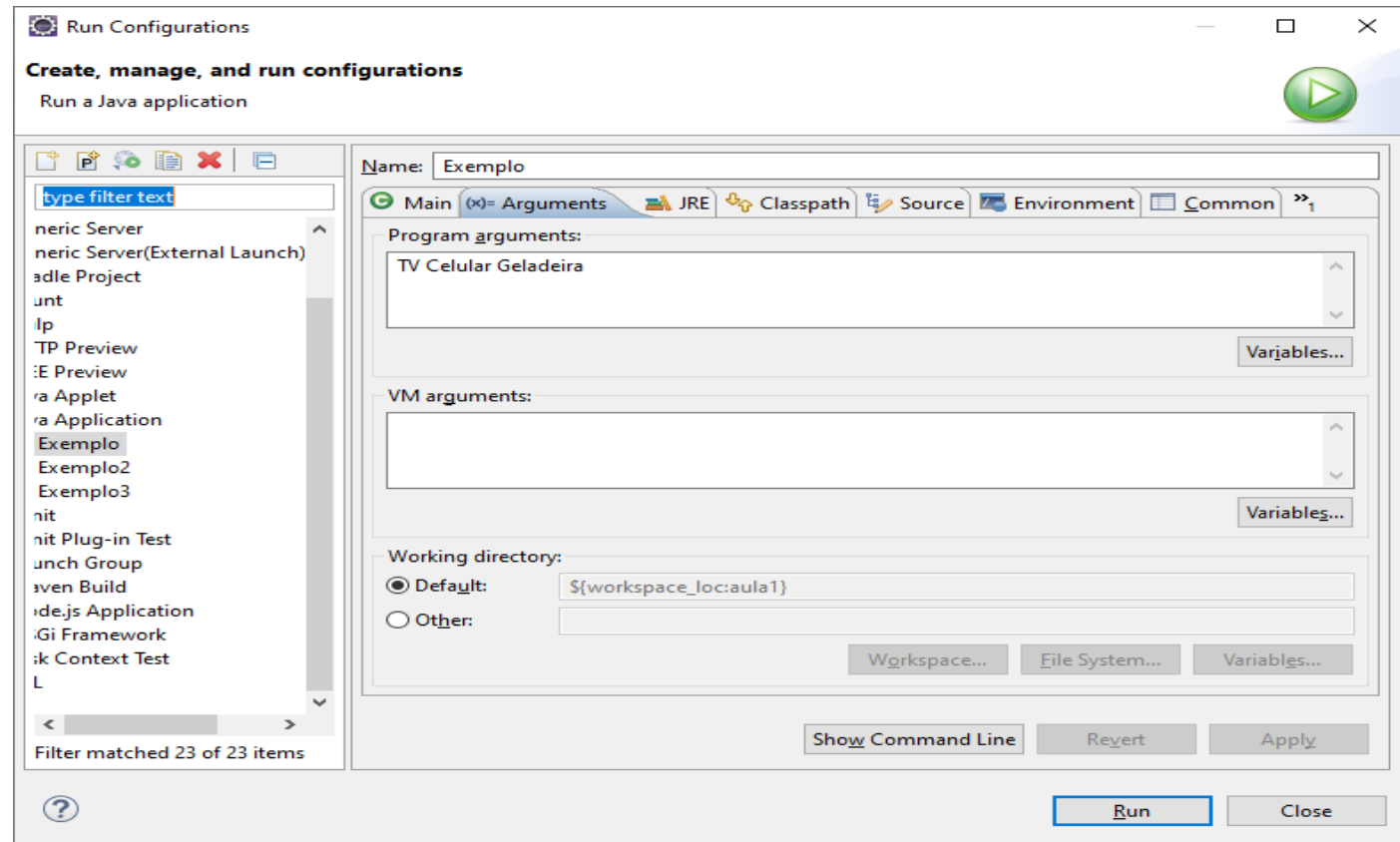
No exemplo utilizado em linha de comando que passamos argumentos para o método main o mesmo exemplo é implementado no Eclipse.

```
package aulas;

public class Exemplo {

    public static void main(String[] args) {
        System.out.println("Hello World !!");
        System.out.println(args[0]);
        System.out.println(args[1]);
        System.out.println(args[2]);
    }
}
```

No menu, clique Run Configurations e preencha os argumentos.



## Visualizando a implementação de uma classe

Para exibir a implementação da classe **System** pressione a tecla **CTRL** sobre e clique na opção Open Implementation. Caso seja exibida a mensagem **Class Not Found**, clique na opção **Change Attached Source** - No diretório da sua jdk, procure pelo arquivo **src.zip**

## Indentação e Alertas

No código abaixo ao digitar o comando **if CTRL+Espaço** um bloco é inserido e automaticamente o texto abaixo é indentado.

**CTRL + SHIFT + F** – Serve para indentar o seu código

O Eclipse emite alertas através do ícone 

```
public class Exemplo2 {  
    public static void main(String[] args) {  
        System.out.println("Programação Java 1");  
        String nome = "Roni";  
        if (nome == null) {  
            System.out.println("nome nulo");  
        }  
    }  
}
```

## Importações

**CTRL + SHIFT + O** – Serve para importar um recurso de outro pacote.

```
import java.time.LocalDateTime;
import java.util.Date;
import java.util.List;

public class Roni {

    public static void main(String[] args) {
        Date data = new Date();
        List lista;
        LocalDateTime dataHora;
    }
}
```

## Replicação de Código

**CTRL+ALT+↓** - replica uma ou várias linhas de código para linha abaixo.

**CTRL+ALT+↑** - replica uma ou várias linhas de código para linha acima.

```
public class Exemplo2 {  
    public static void main(String[] args) {  
  
        System.out.println("Programação Java 1");  
        System.out.println("Programação Java 1");  
        System.out.println("Programação Java 1");  
        System.out.println("Programação Java 1");  
        System.out.println("Programação Java 1");  
  
    }  
}
```

## Blocos



**ALT+SHIFT+A** – Modo seleção em blocos.

```
public class Exemplo2 {  
    public static void main(String[] args) {  
  
        System.out.println("Java"+1);  
        System.out.println("Java"+2);  
        System.out.println("Java"+3);  
        System.out.println("Java"+4);  
        System.out.println("Java"+5);  
    }  
}
```

## Apagar

**CTRL + d** – Apagar uma linha.

**CTRL + Del** – Apagar a próxima instrução.

**CTRL + Backspace** – Apagar instrução anterior.

## Movimentação

**ALT + ↓** - Move linha para baixo.

**ALT + ↑** - Move linha para cima.

Após a seleção preencha com zeros.

```
public class Exemplo2 {  
    public static void main(String[] args) {  
  
        System.out.println("Java"+1000);  
        System.out.println("Java"+2000);  
        System.out.println("Java"+3000);  
        System.out.println("Java"+4000);  
        System.out.println("Java"+5000);  
    }  
}
```



## Marcar Ocorrências



Permite visualizar as ocorrências de uma determinada variável ou comando.

```
public class Teste {  
    public static void main(String[] args) {  
        System.out.println("Java" +1000);  
        System.out.println("Java"+2000);  
        System.out.println("Java"+3000);  
        System.out.println("Java"+4000);  
        System.out.println("Java"+5000);  
    }  
}
```

## Seleção

**SHIFT+ALT + ↑** Seleciona um bloco.

## Navegação

**CTRL + SHIFT + R** - Pesquisa por classes ou arquivos em todos projetos.

**CTRL + SHIFT + T** - Pesquisa por classes de projetos e do Java.

**CTRL + M** - Janela Inteira.

**CTRL + W** - Fechar janela atual

**CTRL + SHIFT + W** - Fechar todas janelas

**CTRL + PG DOWN** - Próxima aba.

**CTRL + PG UP** - Aba anterior.

**CTRL + E** - Exibe um caixa de diálogo para busca de uma classe.

**CTRL + Q** - O cursor vai para o local da última edição.

## Zoom

**CTRL++** Aumentar zoom.

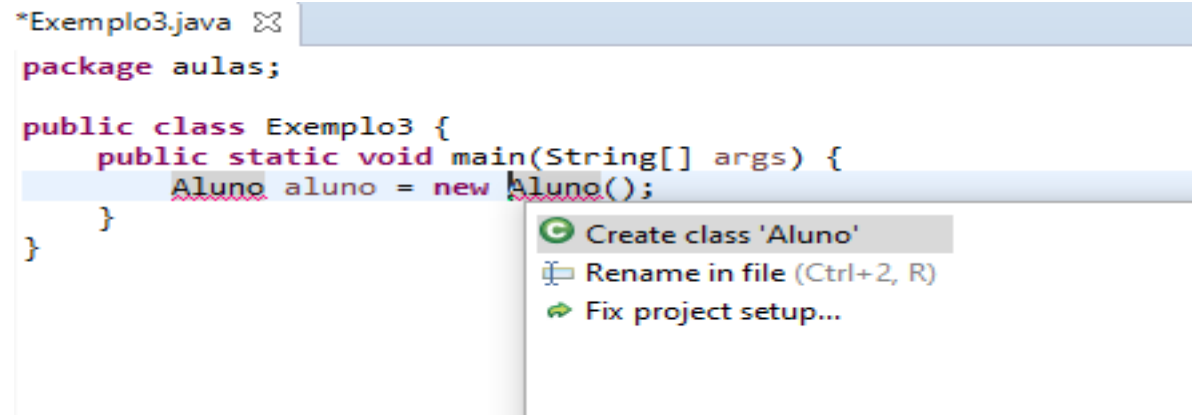
**CTRL--** Diminuir zoom.

## Atalhos

**CTRL+SHIFT+ L** - Exibe todos os atalhos do Eclipse

## Manipulação de Erros

**CTRL + 1** – Realiza correções automaticamente.



```
*Exemplo3.java ✖
package aulas;

public class Exemplo3 {
    public static void main(String[] args) {
        Aluno aluno = new Aluno();
    }
}
```

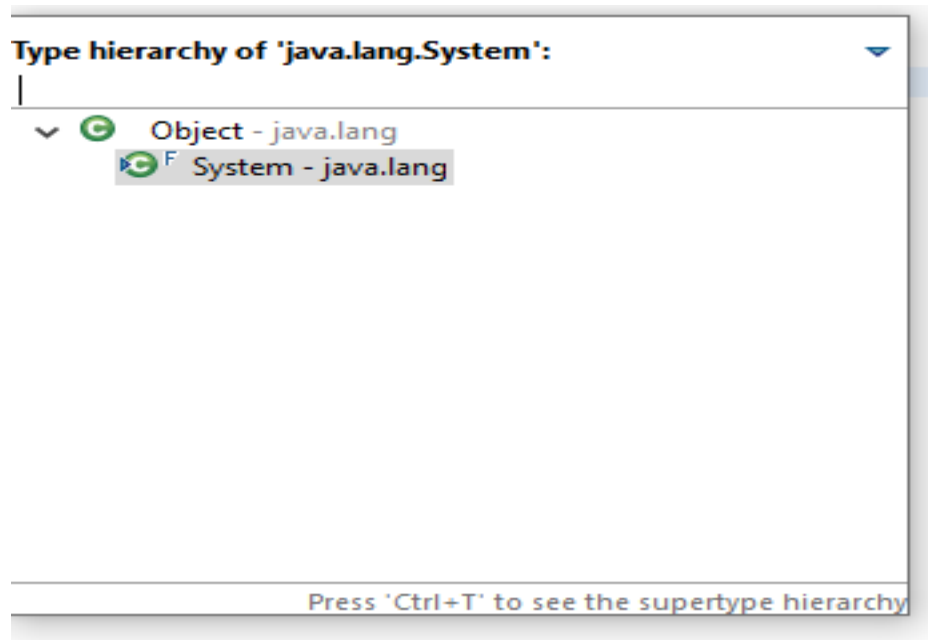
- Create class 'Aluno'
- Rename in file (Ctrl+2, R)
- Fix project setup...

## Execução

**CTRL + F11** – Executa o código.

**F11** – Modo debug.

**CTRL + T** – Exibe a estrutura de herança de um elemento.



## Localização e Substituição

**CTRL + F**

**Move a linha**

**ALT + UP/DOWN**

**Atalho e recursos**

**ALT + SHIFT + S**

**Busca por qualquer recurso**

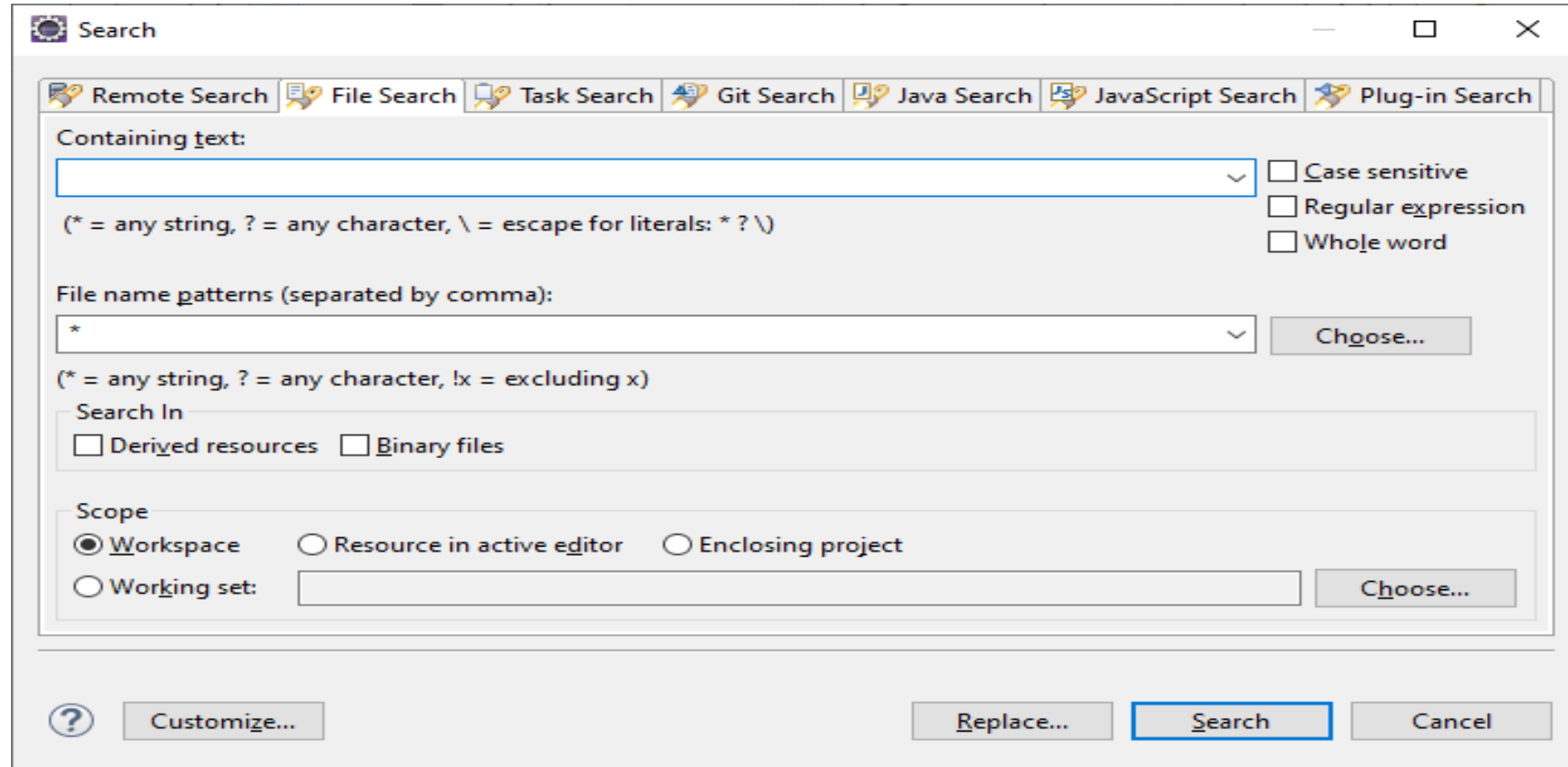
**CTRL + 3**

**No exemplo abaixo estou pesquisando pelo console**








## Buscas

Podemos realizar buscar nos projetos e em conteúdo dos arquivos através do menu **Search**.



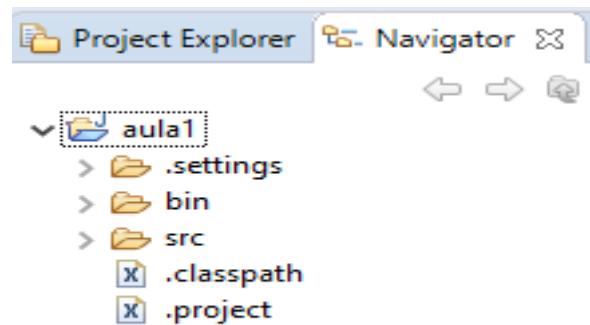
## Pastas do Projeto

Dentro do diretório como o nome do projeto onde o Workspace foi criado temos a estrutura de pastas. abaixo:

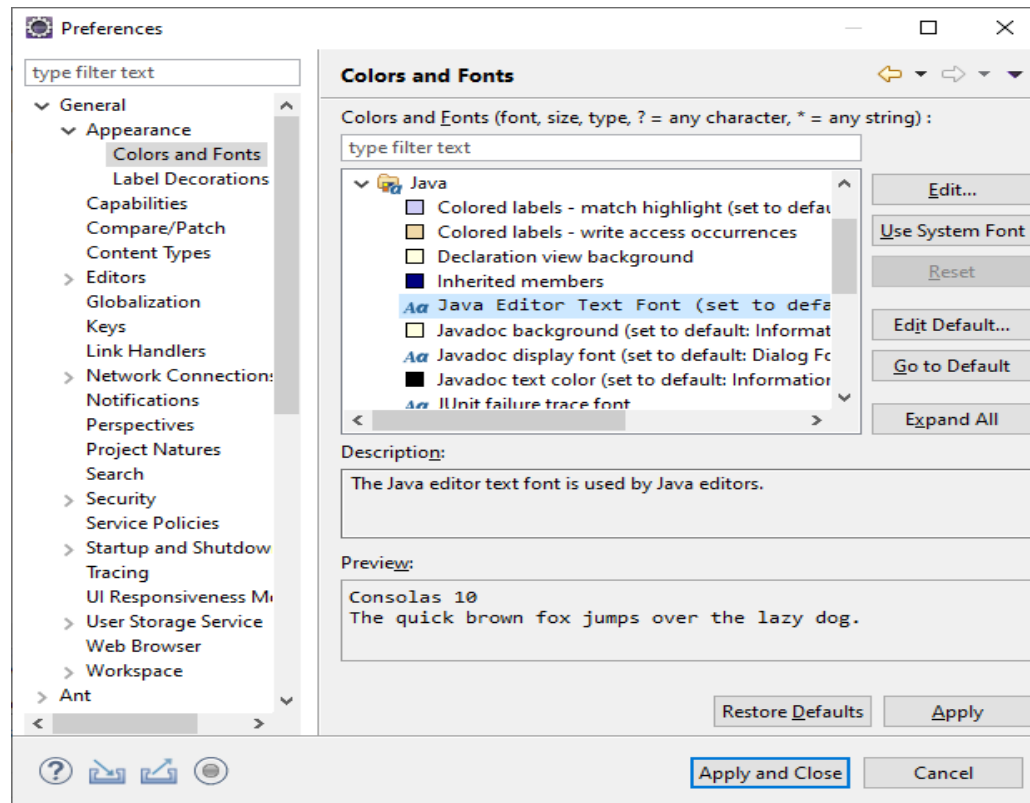
 .settings	24/07/2020 10:20	Pasta de arquivos	
 bin	25/07/2020 11:09	Pasta de arquivos	
 src	24/07/2020 23:28	Pasta de arquivos	
 .classpath	24/07/2020 10:20	Arquivo CLASSPA...	1 KB
 .project	24/07/2020 10:20	Arquivo PROJECT	1 KB

- A pasta **bin** contém os arquivos .class
- A pasta **src** os arquivos .java
- O arquivo **.classpath** serve para informar onde serão armazenados os arquivos .class e .java
- O arquivo **.project** é utilizado pelo eclipse para configurações referente ao projeto.

Para visualizarmos a estrutura de pastas no Eclipse pressione CTRL+3 digite **navigator**



No menu **Window – Preferences** conforme imagem abaixo fazemos a customização da fonte no editor do Eclipse. No exemplo vamos alterar o tamanho da fonte para 12 e negrito.



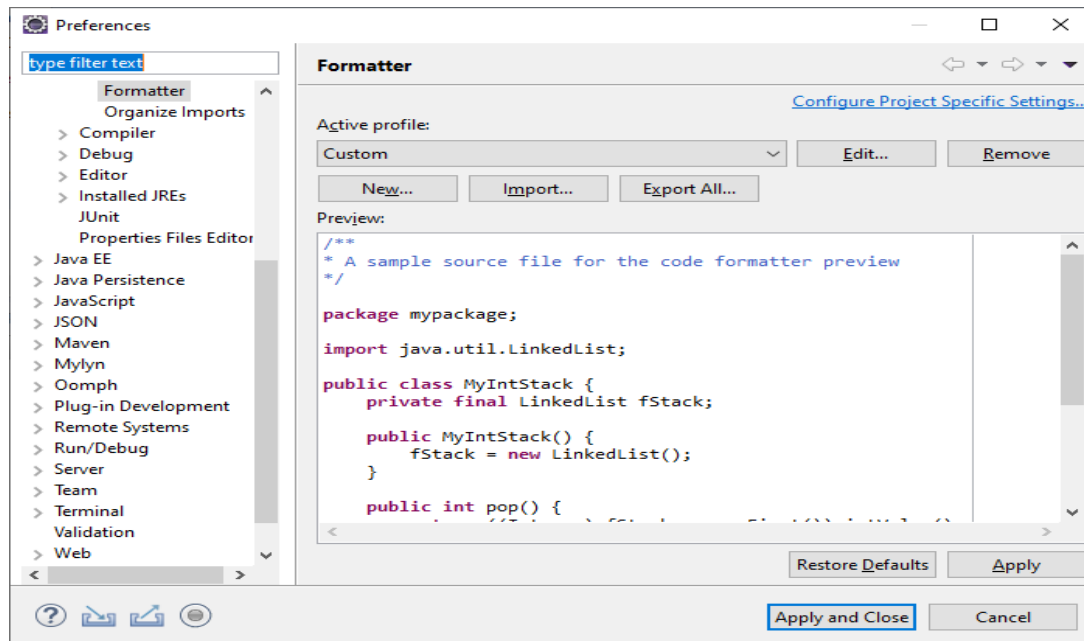


## Alterando o tamanho da coluna

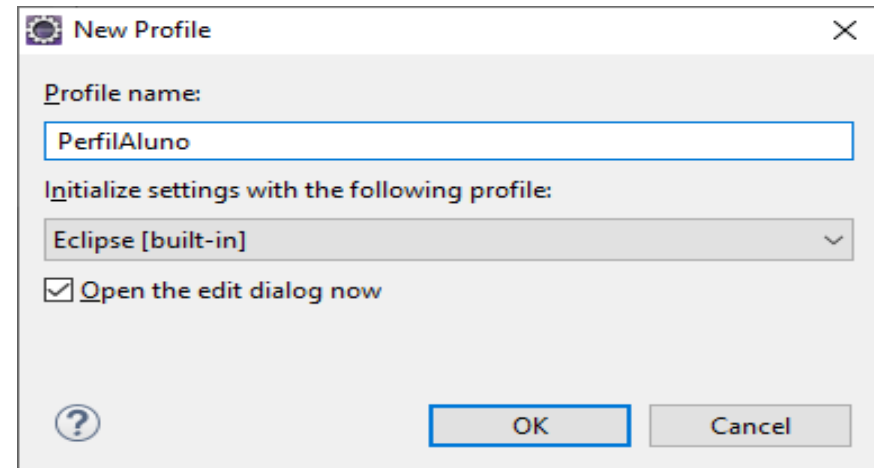
No exemplo abaixo quando fazemos a indentação do texto e o mesmo não fica na mesma linha. Para alterar a opção e aumentar o tamanho da coluna para que o texto seja exibido em uma única linha precisamos configurar o Eclipse.

```
*Exemplo.java ×
1 package teste;
2
3 public class Exemplo {
4     public static void main(String[] args) {
5         System.out.println("Curso de Programação Orientada a Objetos Java"
6             + "Módulo de Introdução");
7     }
8 }
```

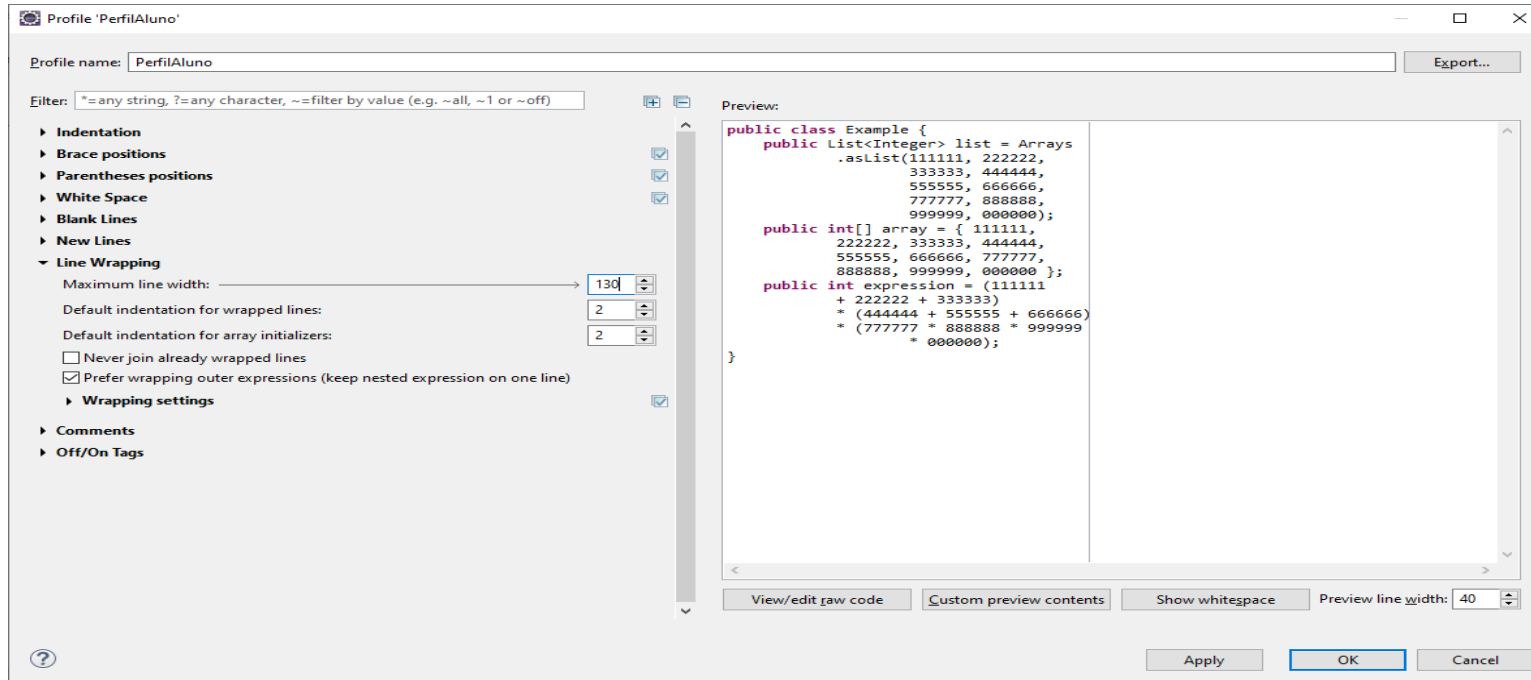
Menu **Window – Preferences** digite **Formatter**



Clique em **New** e insira o nome do perfil



Na próxima tela insira o tamanho da coluna em Line Wrapping



Ao pressionar CTRL+SHIFT+F o código é colocado em uma única linha

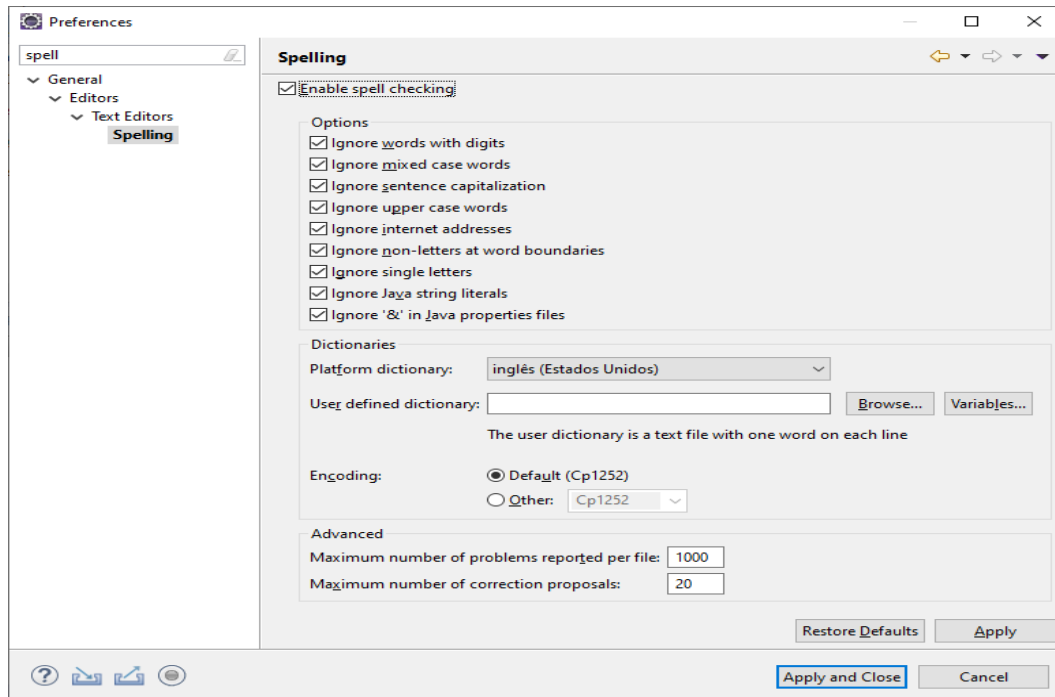
```
package teste;  
  
public class Exemplo {  
    public static void main(String[] args) {  
        System.out.println("Curso de Programação Orientada a Objetos Java Módulo de Introdução");  
    }  
}
```

## Aumentando o desempenho desabilitando alguns recursos

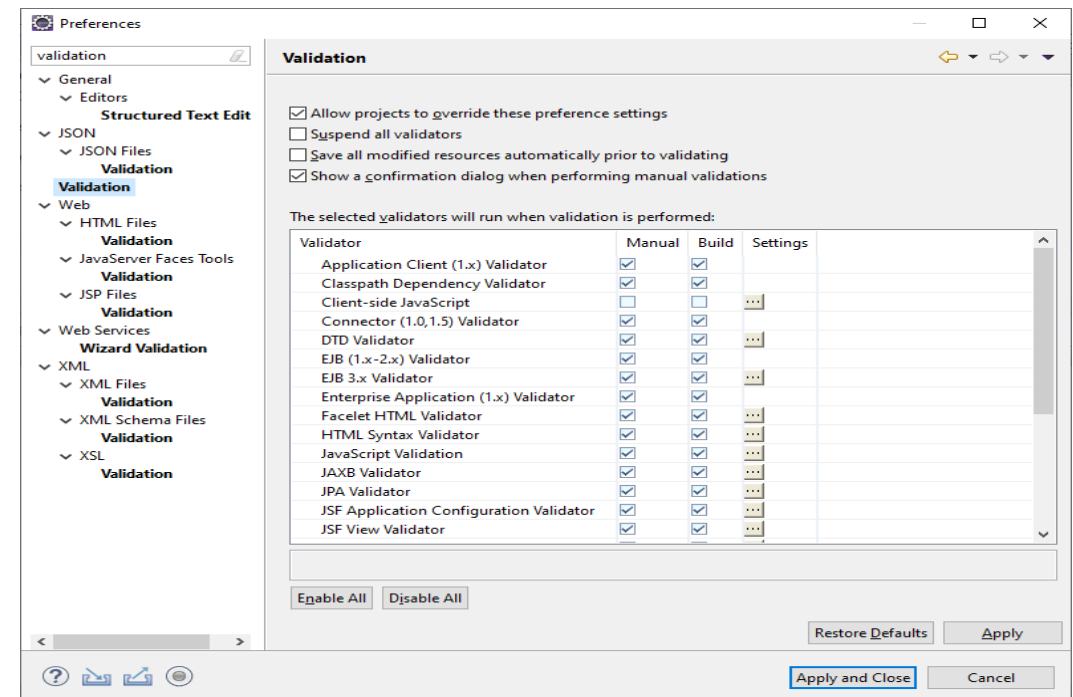
```
package teste;

public class Exemplo {
    public static void main(String[] args) {
        System.out.println("Curso de Programação Orientada a Objetos Java Módulo de Introdução");
    }
}
```

Desabilitar verificação ortográfica  
Window - Preferences - Spelling

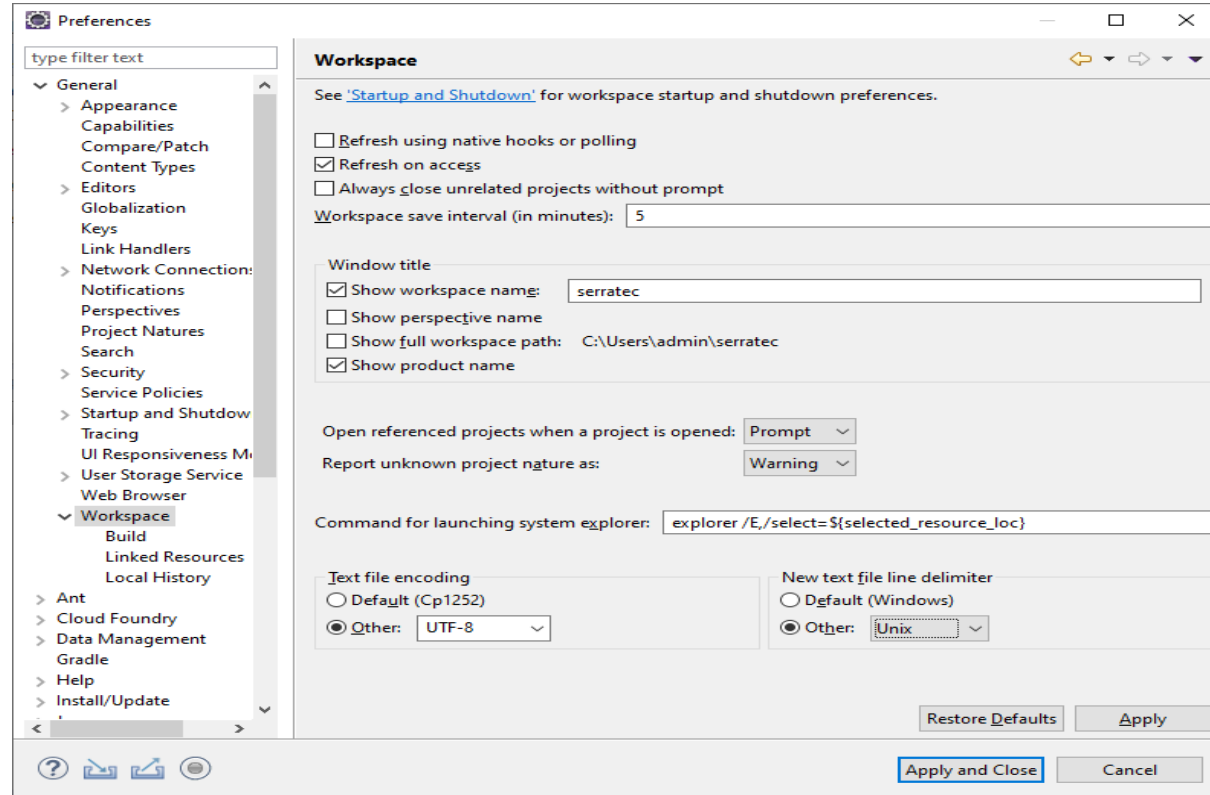


Desabilitar validações  
Window - Preferences - Validation



## Codificação

Configurar a codificação de caracteres é importante para ambientes de desenvolvimento com sistemas operacionais diferentes.



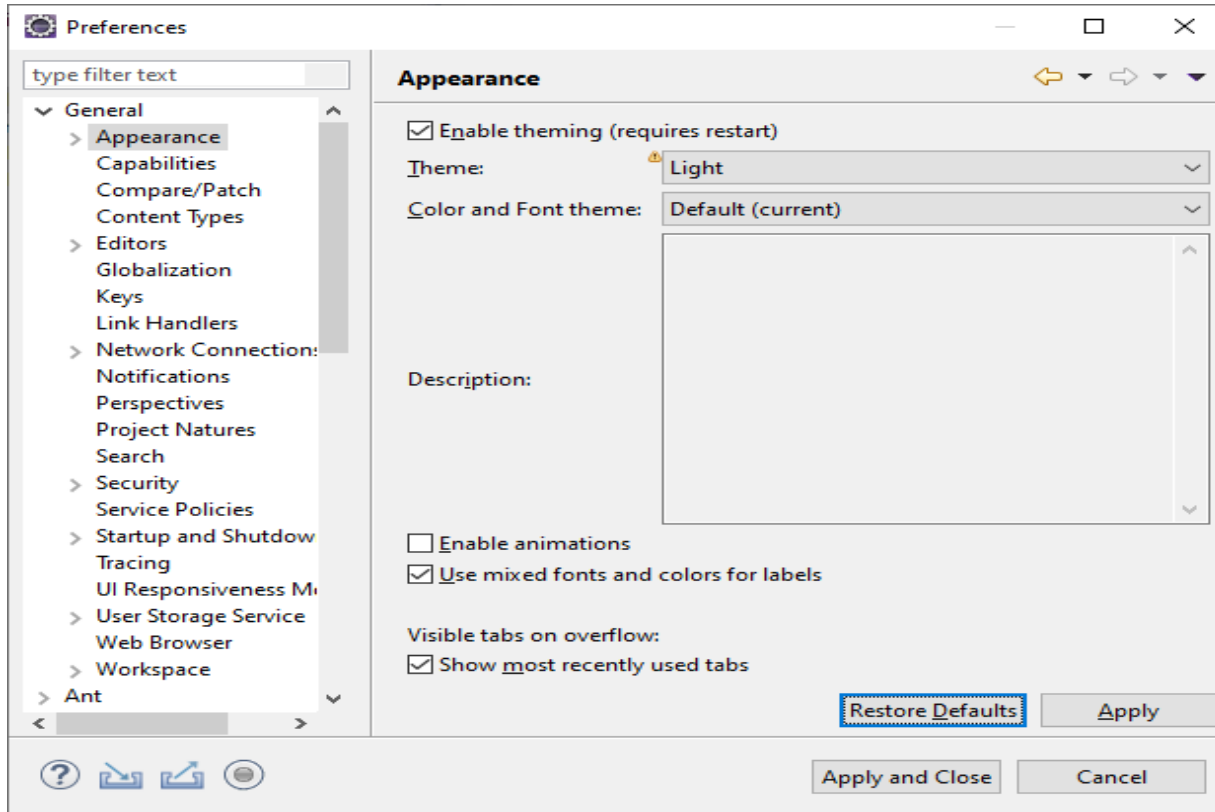
Text File Encoding

UTF-8

Delimitador Unix



## Temas

Clique no menu Window – Preferences – General - Appearance



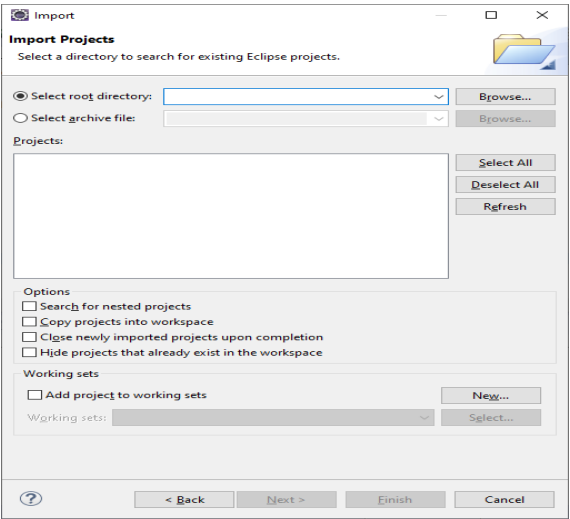
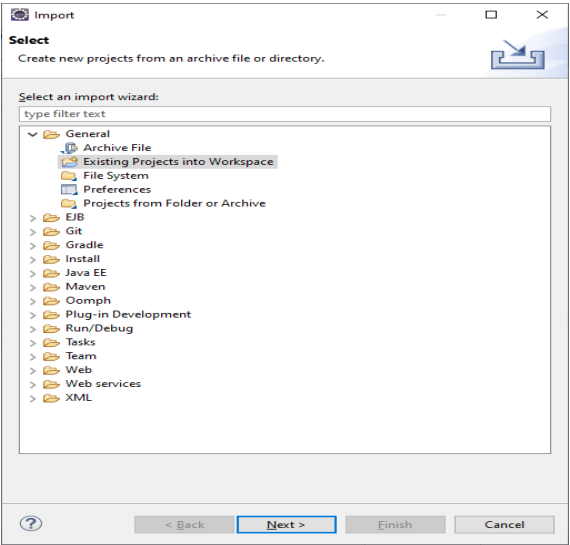
## Exportação Projeto Java

Abra a pasta do seu **workspace** e copie a pasta para o local de destino

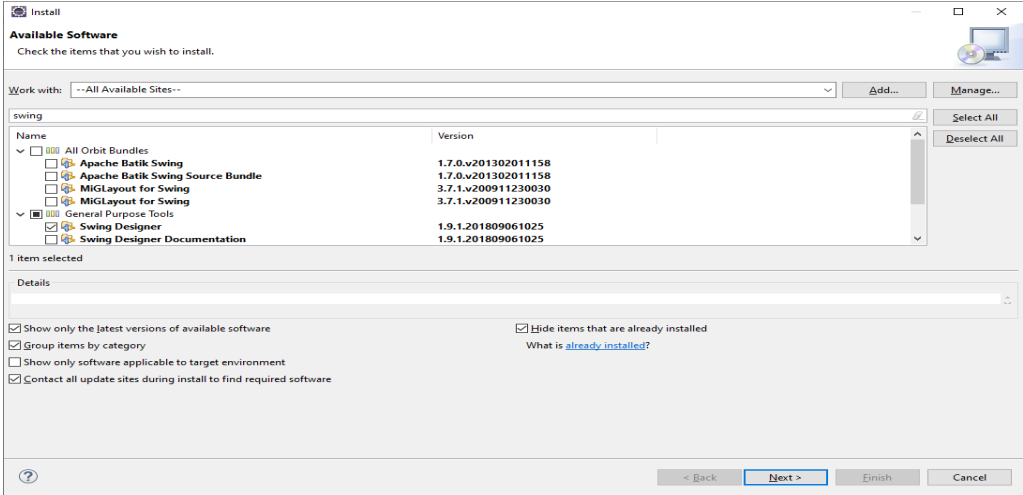
 .metadata	24/07/2020 10:18	Pasta de arquivos
 aula1	24/07/2020 10:20	Pasta de arquivos

## Importação Projeto Java

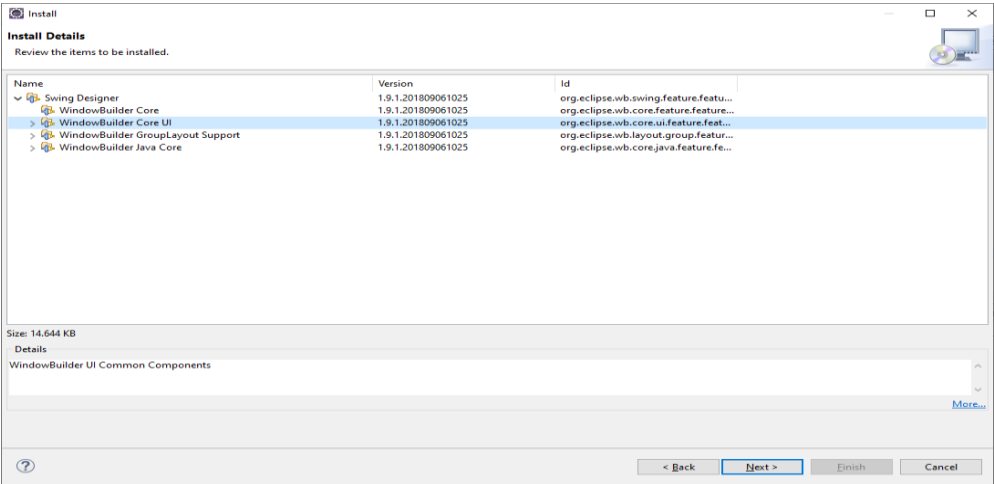
Para importar copie a pasta para o workspace da máquina de destino e abra o Eclipse e clique no menu File - Import - General – Existing project into Workspace. Selecione o diretório da pasta.



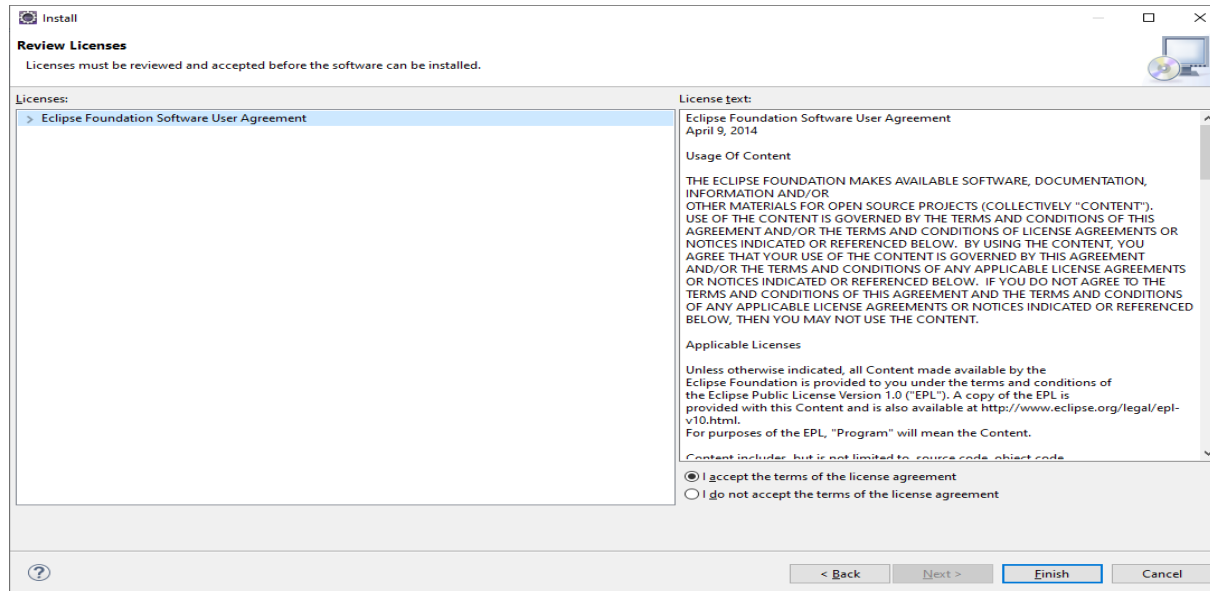
Para instalação de um novo plugin, selecione o menu help install new software. Selecione All Available Sites  
Como exemplo faremos a instalação do Swing. Digite Swing na caixa de texto, selecione a opção Swing Design



Selecione Window Builder Core UI



Marque a primeira opção e clique em Finish.



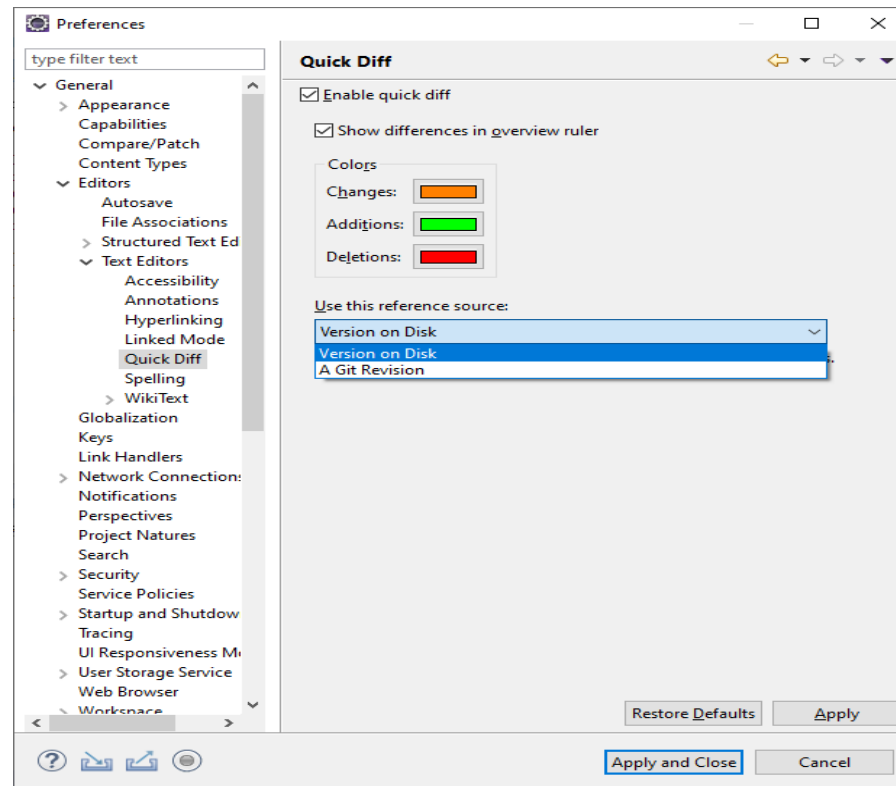
Existem outros plugins que podem ajudar o desenvolvedor como o FindBugs e o CheckStyle por exemplo.



O Eclipse suporta plugins para controle de versão como git e svn por exemplo. Podemos comparar as linhas que foram modificadas em nosso projeto e não foram feitas commit. Esta configuração pode habilitada com o atalho CTRL+SHIFT+Q ou clicando com o botão direito do mouse em um arquivo do projeto próximo ao identificador de linha.



Podemos também alterar as cores de destaque em Windows Preferences

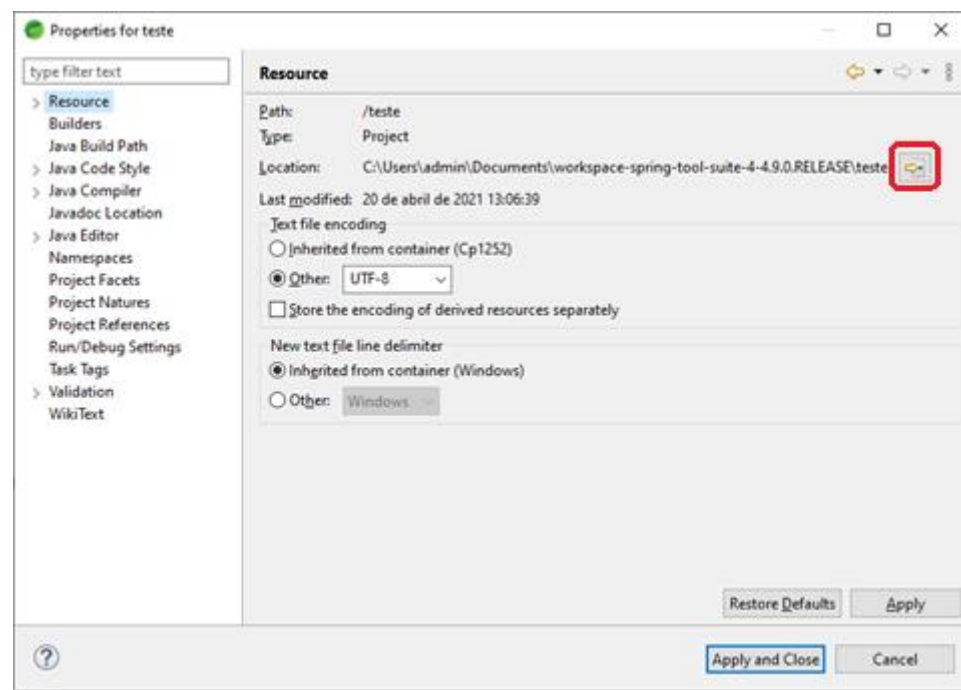


As configurações efetuadas em um determinado workspace não são automaticamente replicadas para novos workspaces.

Para efetuar a cópia seguir os passos abaixo:

Dentro do workspace configurado, localize e copie a pasta **.settings** para o novo workspace.

Obs: Só copiar esta pasta.



Nome	Data de modificação	Tipo	Tamanho
org.eclipse.core.resources.prefs	20/04/2021 14:55	Arquivo PREFS	1 KB
org.eclipse.jdt.core.prefs	20/04/2021 15:27	Arquivo PREFS	32 KB
org.eclipse.jdt.ui.prefs	20/04/2021 15:27	Arquivo PREFS	1 KB
org.springframework.ide.eclipse.xml.namespaces.prefs	20/04/2021 15:27	Arquivo PREFS	1 KB

- 1-Abra o Eclipse
- 2-Crie seu workspace
- 3-Adicione as perspectivas: **Java e Debug**
- 4-Criar um novo projeto com nome “**introducao**”
- 5-Criar o pacote “**aulas**”
- 6-Criar uma classe “**Exercicio.java**” dentro do pacote e criar o método “main”
- 7-Imprimir na tela em cada linha a frase “**Curso Técnico de Programação**”
- 8 – Imprimir na tela “Hello World !!” usando o código **Sytem.err.println** e pesquisar a diferença Entre o atributo out e err.