

#### Sumário

- Introdução ao IDE NetBeans
- Exemplos básicos de Programação Sequencial
- Language Basics: <http://java.sun.com/docs/books/tutorial/java/nutsandbolts/index.html>
- Variables

#### Documentação complementar:

- [Aplicação Hello World - Windows, Mac e Linux](#)
- [NetBeans IDE Java Quick Start Tutorial](#)
- [Language Basics](#)

#### Observações

Se já possui o Netbeans IDE instalado e não consegue criar um projeto Java, deve proceder à instalação do plugin Java que se encontra em: **Tools -> Plugins -> Separador: Available Plugins**. Pesquise pelo plugin: **Java**.

## Introdução a ferramentas de desenvolvimento para a linguagem JAVA

### Compilar e executar um programa através de linha de comandos

1. Proceda à instalação do [JDK 8 ou superior](#) (Java Development Kit) de acordo com o seu sistema operativo.
2. Teste se o java já está funcional no sistema: Numa linha de comandos execute o comando:

```
java -version
```

Em caso de falha verificar se o *path* do sistema operativo inclui o caminho para as ferramentas do java:

```
SET PATH=JDK_INSTALL_DIR\bin;PATH (Windows)
export PATH=$PATH:JDK_INSTALL_DIR/bin (Unix)
```

Mais informação:

- <https://www.java.com/en/download/help/path.xml>
- <https://docs.oracle.com/javase/tutorial/getStarted/cupojava/index.html>

3. Crie um ficheiro java numa pasta à escolha (Sem utilizar um IDE!) chamado `Welcome.java` com o seguinte código:

```
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Saudações javanesas ");
    }
}
```

4. Nessa pasta executar: `javac Welcome.java`
5. Nessa pasta executar: `java Welcome`  
(em ambiente *Windows* terá de utilizar o comando: `java -cp Welcome.java`)

**Nota:** Se o ficheiro Welcome.java contiver a descrição do package, então deverá executar o comando java na raiz da diretoria que contém o nome do package e executar no seguinte formato: **java PACKAGE\_NAME.Welcome**

## Compilar e executar um programa através do IDE: [Netbeans](#)

O tutorial apresentado de seguida foi parcialmente retirado e adaptado de: [Link](#).

### Tutorial

Este tutorial proporciona uma introdução muito simples e rápida ao NetBeans IDE através da criação de uma aplicação de consola Java designada *Hello World*.

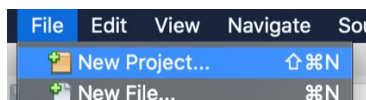
**Para a realização deste tutorial é necessário o seguinte software e recursos:**

Software or Recursos	Versão Necessária
NetBeans IDE	8.x ou posterior
Java Development Kit (JDK)	Versão 8+

### Preparação do Projeto

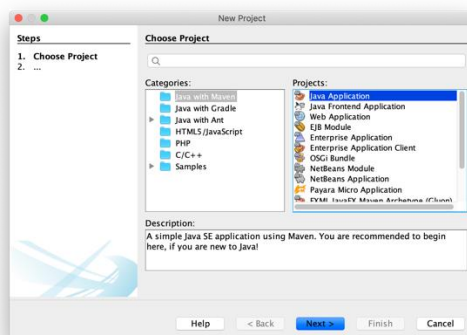
Para criar um projeto no NetBeans IDE:

1. Inicie o NetBeans IDE
2. No IDE, selecione *File > New Project (Ctrl-Shift-N)* como apresentado na Figura 1



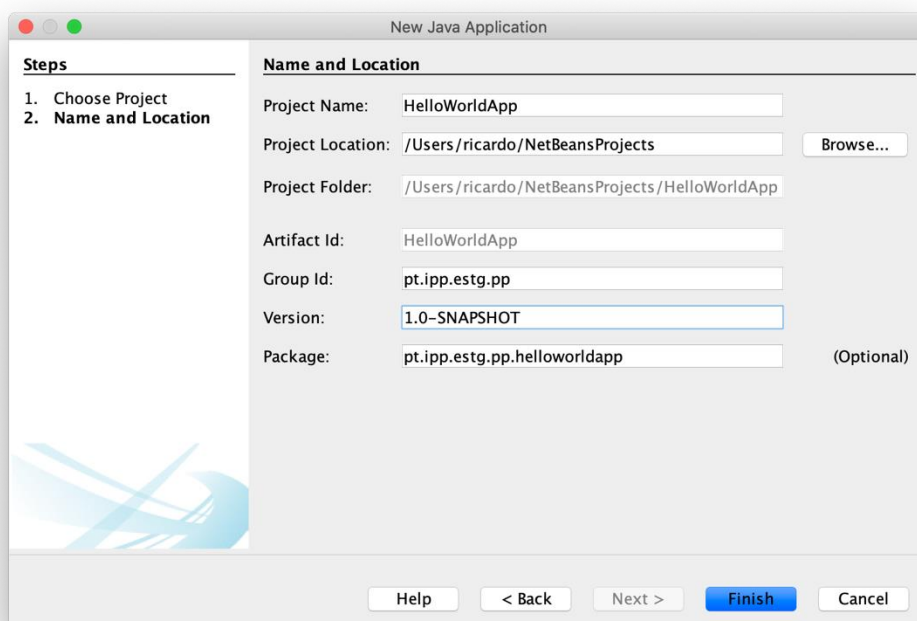
*Figura 1 – Exemplo de criação de um novo projeto em MacOS X*

3. Na janela de criação de novo projeto, expanda a categoria Java e selecione Java Application como apresentado na Figura 2. De seguida clique *Next*.



*Figura 2 - Seleção de um novo projeto Java*

4. Na página de nome e localização do projeto, insira a seguinte informação (Figura 3)
  - a. Project Name: HelloWorldApp
  - b. Deixe a localização do projeto com o valor apresentado por omissão
  - c. Group ID: pt.ipp.estg.pp
  - d. Version e Package: valores por omissão



*Figura 3 - Informação para criação de novo projeto*

#### 5. Clique *Finish*

O projeto é criado e aberto no IDE. Deverá visualizar os seguintes componentes:

- Janela dos projetos (*Project Window*), que contém uma vista em árvore dos componentes do projeto, incluindo *source files*, bibliotecas (*libraries*) cujo projeto esteja dependente, entre outros;
- Janela de edição (*Source Editor*)
- Janela de navegação (*Navigation Windows*), que pode utilizar para rapidamente navegar entre elementos da classe selecionada
- Janela de tarefas (*Tasks Window*), que lista erros de compilação assim como outras tarefas que se encontram marcadas com *keywords* como **TODO**

### Adicionar Código ao projeto criado

A criação do projeto recorrendo ao *Wizard* não gerou uma *Main Class*. Para a criação da *Main Class* execute os seguintes passos

1. Na janela de projetos expanda o projeto *HelloWorldApp* e, de seguida, expanda a pasta *Source Packages*
2. Selecione o package *pt.ipp.estg.pp.helloworldapp*
  - a. *File > New File*
  - b. Na categoria Java selecione *Java Main Class*
  - c. Insira *HelloWorld* como nome da classe (Figura 4)
  - d. Clique *Finish*

-- em alternativa --

  - i. Clique com o botão direito sobre o *package* (Figura 5)
  - ii. Selecione *New > Java Main Classe*
  - iii. Execute os passos descritos em c. e d.

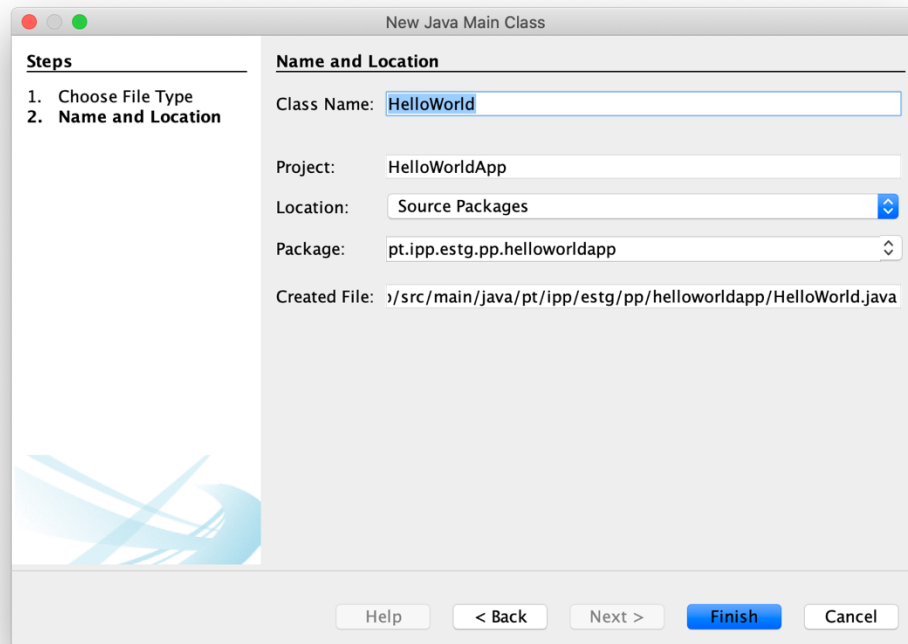


Figura 4 - Nome e localização da classe Main

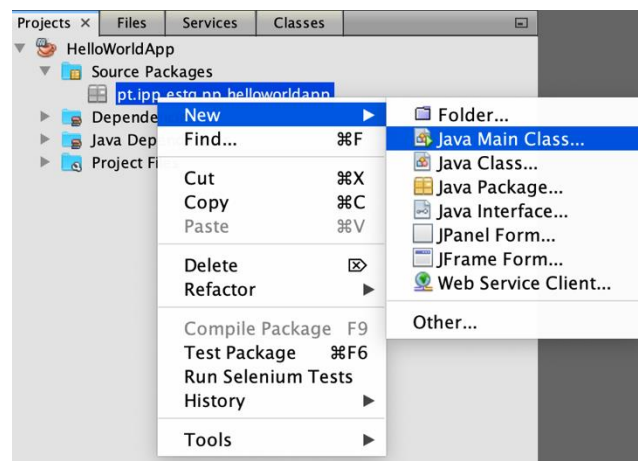


Figura 5 - Criação de Main Class

A classe *Main* recém-criada contém um *skeleton* por omissão. Adicione a mensagem “Hello World!” ao código substituindo a linha:

```
// TODO code application logic here
```

Pela seguinte linha:

```
System.out.println("Hello World!");
```

O resultado final deverá ser semelhante à Figura 6.

Guarde as alterações selecionando *File > Save*.

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package helloworldapp;

/**
 *
 * @author Patrick Keegan
 */
public class HelloWorldApp {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}
```

Figura 6 - Exemplo do código fonte da classe Main

## Compilar e Executar o Programa

Devido à funcionalidade do IDE de compilar ao guardar (Compile on Save), não é necessário proceder à compilação manual do projeto para poder executá-lo no IDE. Quando guardar um ficheiro com código Java, o IDE procede à sua compilação de forma automática.

### Para executar o programa:

- Selecione *Run > Run Main Project* (F6).

Na eventualidade de existirem erros, os mesmos serão marcados com um símbolo vermelho nas margens esquerda e direita do *Source Editor*. Os símbolos na margem esquerda indicam os erros nas linhas correspondentes. Na margem direita são identificadas todas as áreas do ficheiro que contêm erros, incluindo erros em linhas que não se encontram visíveis na janela de edição. Pode colocar o rato sobre a marca de erro para obter uma descrição do mesmo. Pode ainda clicar no símbolo vermelho na margem direita para ir diretamente para a linha com o erro.

## Building and Deploying the Application

Assim que escrever e testar a aplicação, pode utilizar o comando *Clean and Build* para criar a aplicação para distribuição. Quando utiliza este comando o IDE executa um *script* que realiza as seguintes tarefas:

- Elimina qualquer ficheiro anteriormente compilado e outros resultados do *Build*
- Recompila a aplicação e cria um ficheiro JAR que contém os ficheiros compilados.

### Para “construir” (*build*) a aplicação:

- Selecione *Run > Clean and Build Project* (Shift-F11).

Pode verificar os ficheiros criados abrindo a janela de ficheiros (*Files Window*) e expandindo o nó

*HelloWorldApp*. O ficheiro bytecode compilado (*HelloWorldApp.class*) encontra-se no nó *target* (Figura 7).

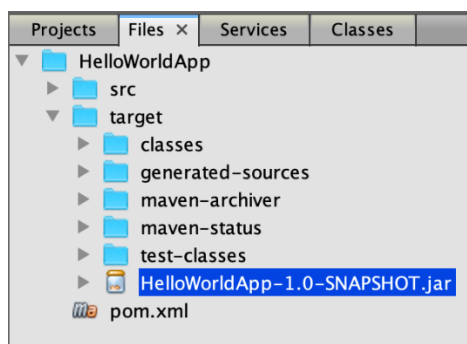
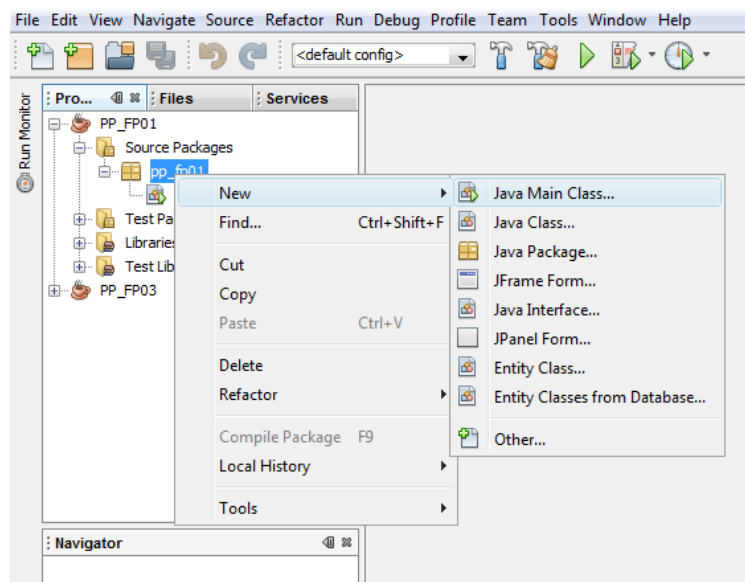


Figura 7 - Resultado da compilação

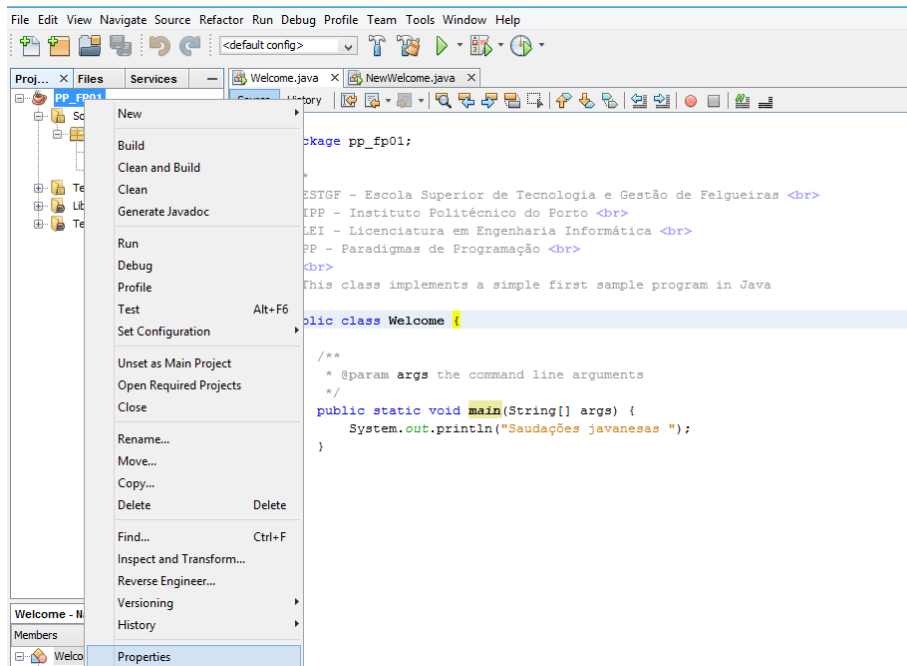
### Exercício:

Experimente outras funcionalidades do IDE, tais como:

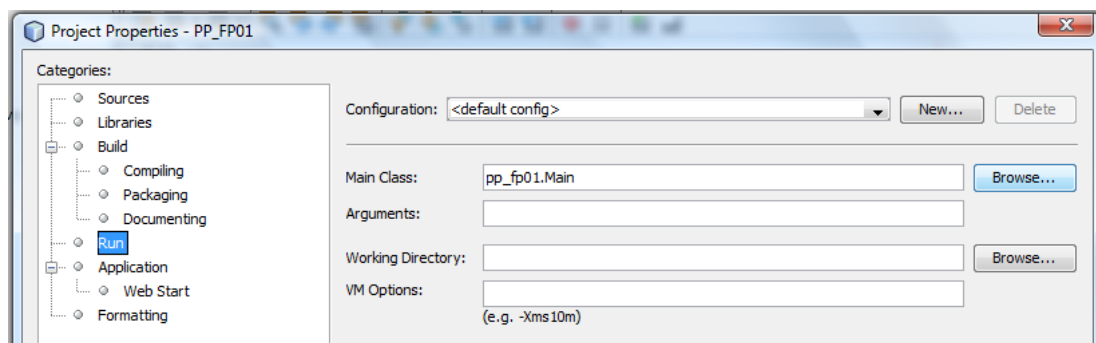
- **Acrescentar nova classe a executar (Main Class):**
  1. Selecione, no explorador de projetos (janela *Projects*), o *Source Package* (“pp\_fp01”) para o qual pretende acrescentar a classe a executar (*Main Class*)
  2. Após pressionar o botão direito do rato sobre o *Source Package* (“pp\_fp01”), selecione a opção **New > Java Main Class**
  3. Na janela de criação da nova Main Class, mantenha os dados sugeridos e clique em *Finish*.



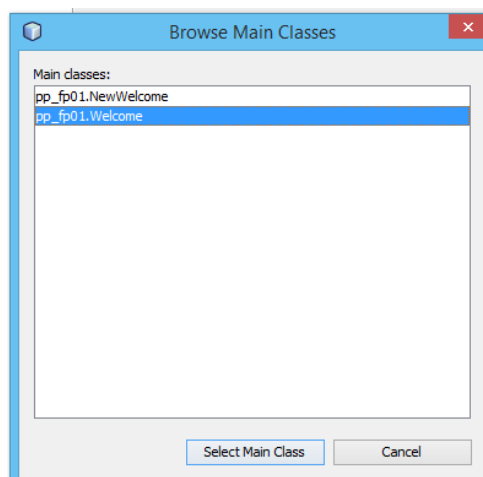
- **Alterar a classe a executar (Main Class):**
  1. Selecione, no explorador de projetos (janela *Projects*), o projeto para o qual pretende alterar a classe a executar (*Main Class*).
  2. Sobre o projeto selecionado (estará *highlighted*, repare que na figura “PP\_FP01” aparece com cor de fundo azul) pressione o botão do lado direito do rato.
  3. No *pop-up menu* selecione a opção *Properties*.



4. Na *dialog box* que aparece selecione *Run* em *Categories*.
5. Ainda na *dialog box* pressione o botão *Browse...*



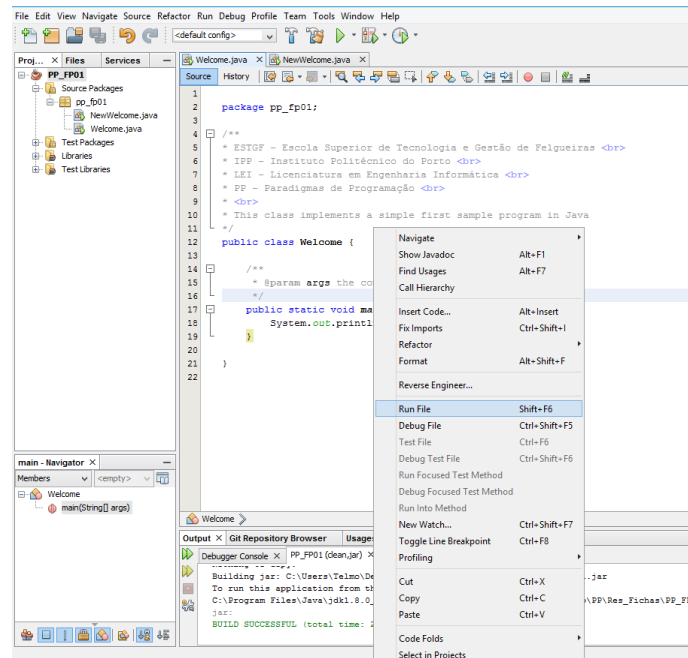
6. Na nova *dialog box* que aparece selecione a classe desejada.



7. Uma vez seleccionada a classe desejada pressione o botão *Select Main Class*.

- **Executar um programa de forma rápida:**

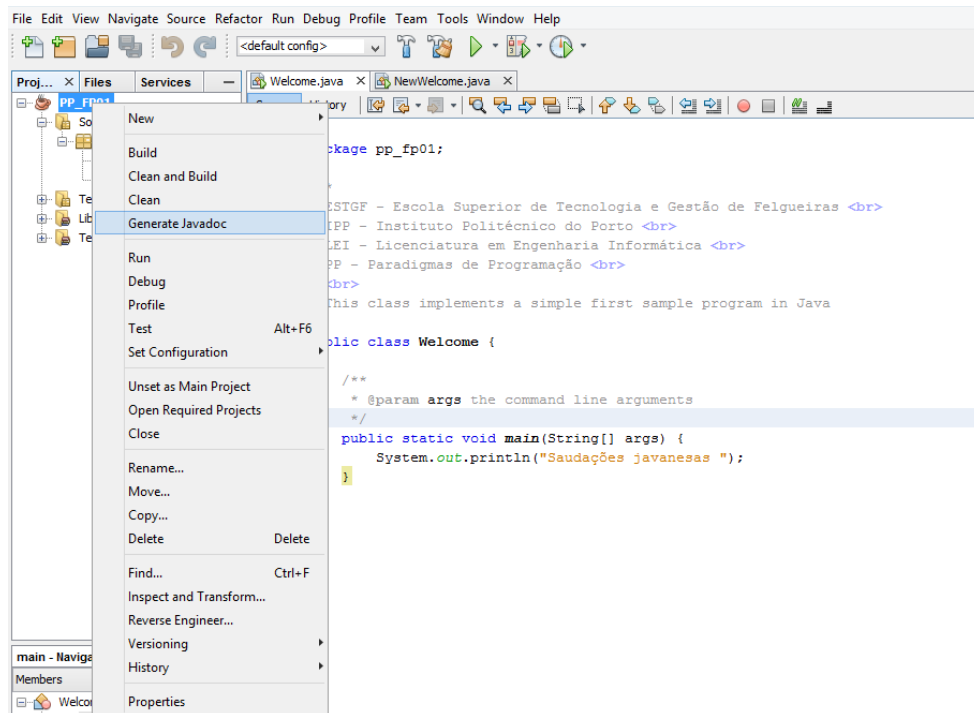
1. Sempre que for necessário executar um programa, pode fazê-lo a partir do botão direito do rato sob a classe (obrigatoriamente com um método **main()**) ao seleccionar a opção *Run File*.



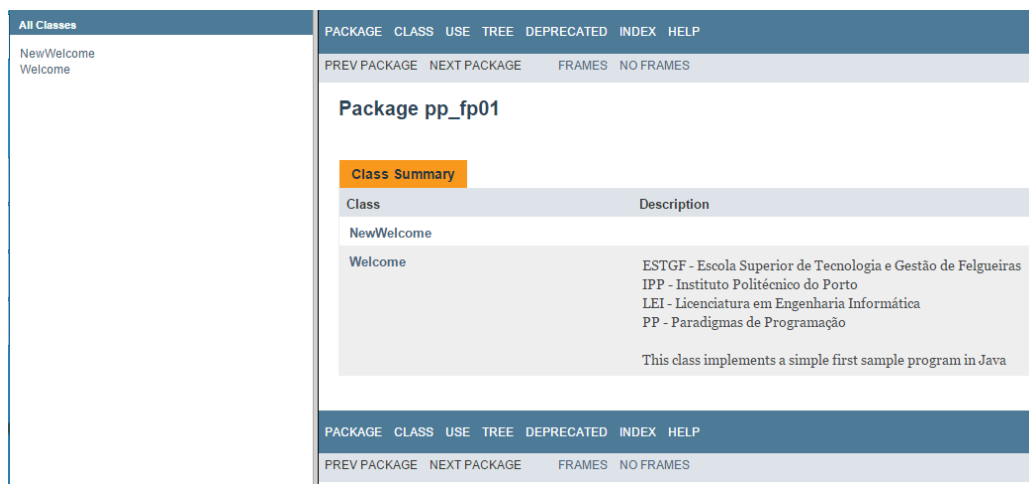
- **Gerar o Javadoc:**

1. Selecione, no explorador de projetos (janela *Projects*), o projeto para o qual pretende gerar o Javadoc.
2. Sobre o projeto seleccionado (estará *highlighted*, repare que na figura “PP\_FP01” aparece com cor de fundo azul) pressione o botão do lado direito do rato.
3. No *pop-up menu* selecione a opção *Generate Javadoc*.



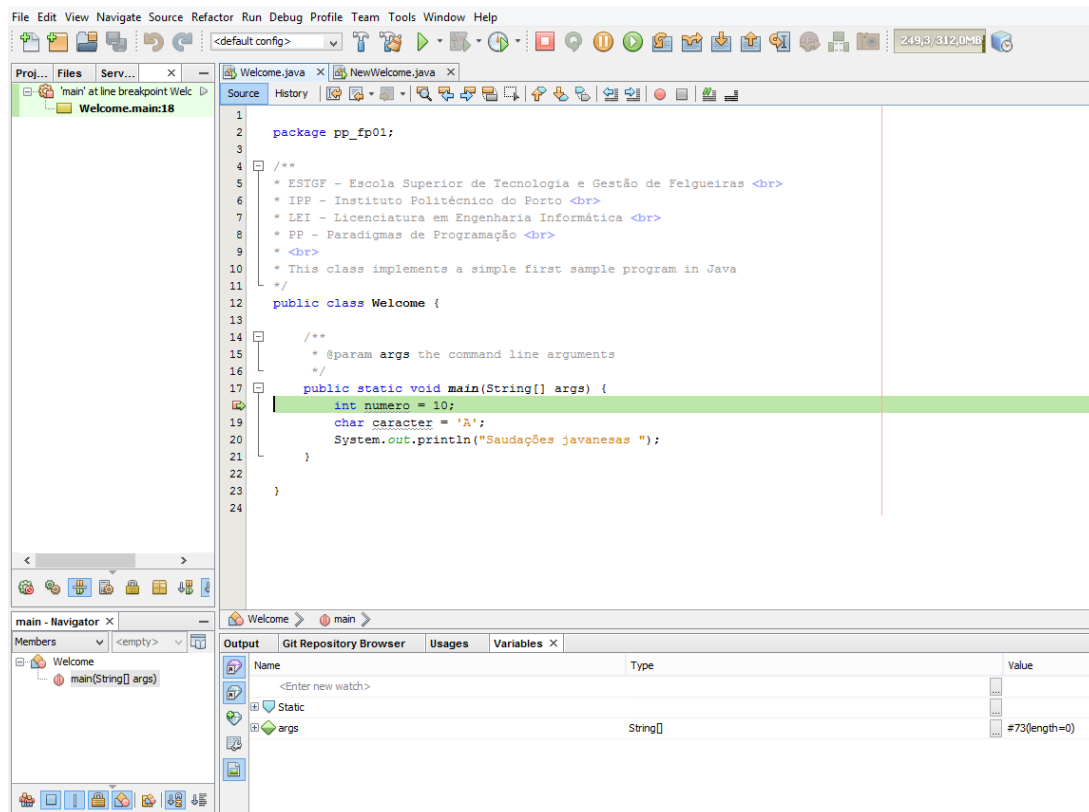


4. O Javadoc é criado em formato html e pode ser consultado em “PP\_FP01/dist/javadoc/index.html”. Para projeto apresentado, o Javadoc gerado é o seguinte:




- **Utilizar o debugger:**

1. Na classe main do projeto anterior e dentro do método main declare uma variável do tipo int atribuindo-lhe o valor 10 e o variável do tipo char atribuindo o valor 'A'
2. Agora clique com o botão direito do rato sobre o número da linha onde declara a variável do tipo inteiro e selecione a opção Breakpoint >> Toggle Line Breakpoint
3. Agora execute este ficheiro em modo de debug (botão direito sobre o nome do ficheiro e seleccionar a opção “Debug File”)
4. Quando a linha onde seleccionou o Breakpoint ficar a verde significa que o programa está a ser executado e está parado nesse ponto



5. Agora pode perceber qual é o estado do programa num determinado momento de execução.
6. O painel “Variables” mostra qual o estado de todas as variáveis existentes até ao momento
7. Agora pode avançar passo a passo de várias maneiras:
  - Step Over – Executa a próxima instrução
  - Step Over Expression - Permite percorrer cada invocação de métodos e visualizar os parâmetros de *input* e o *output* de cada método. Este comando pode ser invocado como qualquer outro mesmo que a instrução selecionada não possua uma invocação de um método. Neste caso, o comportamento será similar ao comando *Step Over*.
  - Step Into – Se a próxima instrução for um método o *debugger* irá mostrar o que acontece dentro dele.
  - Step Out - Se estiver a executar uma instrução dentro de um método, o *debugger* sai desse método e continua a execução no método invocador.
  - Run to Cursor - Avança entre instruções em que foram definidos *breakpoints*.

 <p>ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO</p>	<p><b>LEI/LSIRC</b> <b>PP - Paradigmas de Programação</b> 2º Semestre ■ Docentes: RJS, BMO, CDF, MFG, OAO Ficha Prática 1 – 2023/2024</p>
---------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

## Exercícios

1. Declare no método *main()* variáveis de diversos tipos primitivos e de seguida imprima os valores que essas variáveis armazenam.

### Exemplo:

```
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    //local variables
    char l = 'l', p = 'p';
    int q = 4, d = 2;
    System.out.println(l);
    System.out.println(p);
    System.out.println(d);
    System.out.println(q);
    System.out.println(l + p + 2);
    System.out.println(" " + l + p + 2);
    System.out.println(q + d);
}
```

2. Crie no método *main()* variáveis unidimensionais (vulgarmente conhecidas como arrays) de diversos tipos primitivos, de seguida imprima valores de posições dessas variáveis.

3. Qual o output do seguinte programa?

```
boolean canITakeHisMoney;
int hisBalance = 5;
long myBalance = 4;
hisBalance += 8;
canITakeHisMoney = hisBalance > myBalance;
canITakeHisMoney = canITakeHisMoney & (hisBalance >= 3);
System.out.println(canITakeHisMoney);
```

4. Qual o output do seguinte programa?

```
int v = 0;
v++;
int amount = v++;
System.out.println(++v + " " + amount);
System.out.println(v);
```

5. Escreva um programa que siga as seguintes instruções:

1. Declarar um *long* com valor inicial 0;
2. Imprimir esse valor para a consola;
3. Mudar esse valor para 3;
4. Imprimir novamente a variável;
5. Declarar um *boolean* com valor *false*;
6. Imprimir o valor do *boolean*.

6. Execute o programa e agora acrescente:

1. Uma variável do tipo *double* e um inteiro sem valor inicial;
2. Imprimir os seus valores para a consola.

Por que razão o compilador apresenta avisos (*warning's*)?