P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO	**LEI/LSIRC** **PP - Paradigmas de Programação** 2º Semestre ■ Docentes: RJS, BMO, FAS, TAC, JRMR Ficha Prática 1

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
- JDK 11

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 Developement 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 de: Link. No entanto, alguma secções foram atualizadas.

Tutorial

This tutorial provides a very simple and quick introduction to the NetBeans IDE workflow by walking you through the creation of a simple "Hello World" Java console application.
Source:

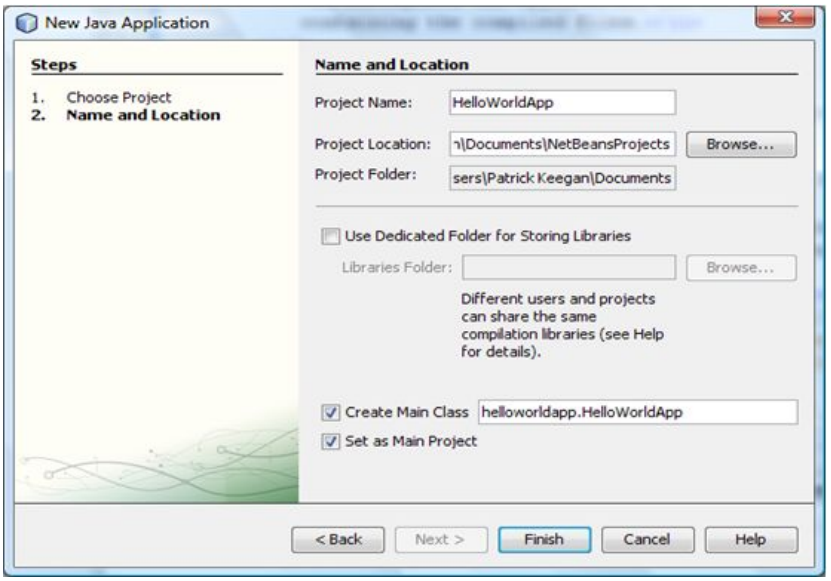
To complete this tutorial, you need the following software and resources.

Software or Resource	Version Required
NetBeans IDE	8.x or later
Java Development Kit (JDK)	Version 8+

Setting Up the Project

To create an IDE project:

1. Start NetBeans IDE.
2. In the IDE, choose File > New Project (Ctrl-Shift-N), as shown in the figure below.
3. In the New Project wizard, expand the Java category and select Java Application as shown in the figure below. Then click Next.
4. In the Name and Location page of the wizard, do the following (as shown in the figure below):
   - In the Project Name field, type `HelloWorldApp`.
   - Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
   - In the Create Main Class field, type `helloworldapp.HelloWorldApp`.
   - Leave the Set as Main Project checkbox selected.



Click **Finish**.

The project is created and opened in the IDE. You should see the following components:

- The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
- The Source Editor window with a file called HelloWorldApp open.
- The Navigator window, which you can use to quickly navigate between elements within the selected class.
- The Tasks window, which lists compilation errors as well as other tasks that are marked with keywords such as XXX and TODO.

Adding Code to the Generated Source File

Because you have left the Create Main Class checkbox selected in the New Project wizard, the IDE has created a skeleton class for you. You can add the "Hello World!" message to the skeleton code by replacing the line:

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

with the line:

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

Save the change by choosing File > Save.

```
The file should look something like the following code sample.

/*
 * 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!");
    }
}
```

Compiling and Running the Program

Because of the IDE's Compile on Save feature, you do not have to manually compile your project in order to run it in the IDE. When you save a Java source file, the IDE automatically compiles it.

To run the program:

- Choose Run > Run Main Project (F6).

If there are compilation errors, they are marked with red glyphs in the left and right margins of the Source Editor. The glyphs in the left margin indicate errors for the corresponding lines. The glyphs in the right margin show all of the areas of the file that have errors, including errors in lines that are not visible. You can mouse over an error mark to get a description of the error. You can click a glyph in the right margin to jump to the line with the error.

Building and Deploying the Application

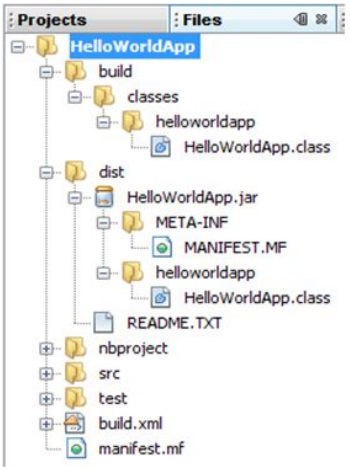
Once you have written and test run your application, you can use the Clean and Build command to build your application for deployment. When you use the Clean and Build command, the IDE runs a build script that performs the following tasks:

- Deletes any previously compiled files and other build outputs.
- Recompiles the application and builds a JAR file containing the compiled files.

To build your application:

- Choose Run > Clean and Build Main Project (Shift-F11).

You can view the build outputs by opening the Files window and expanding the HelloWorldApp node. The compiled bytecode file HelloWorldApp.class is within the build/classes/helloworldapp subnode. A deployable JAR file that contains the HelloWorldApp.class is within the dist node.



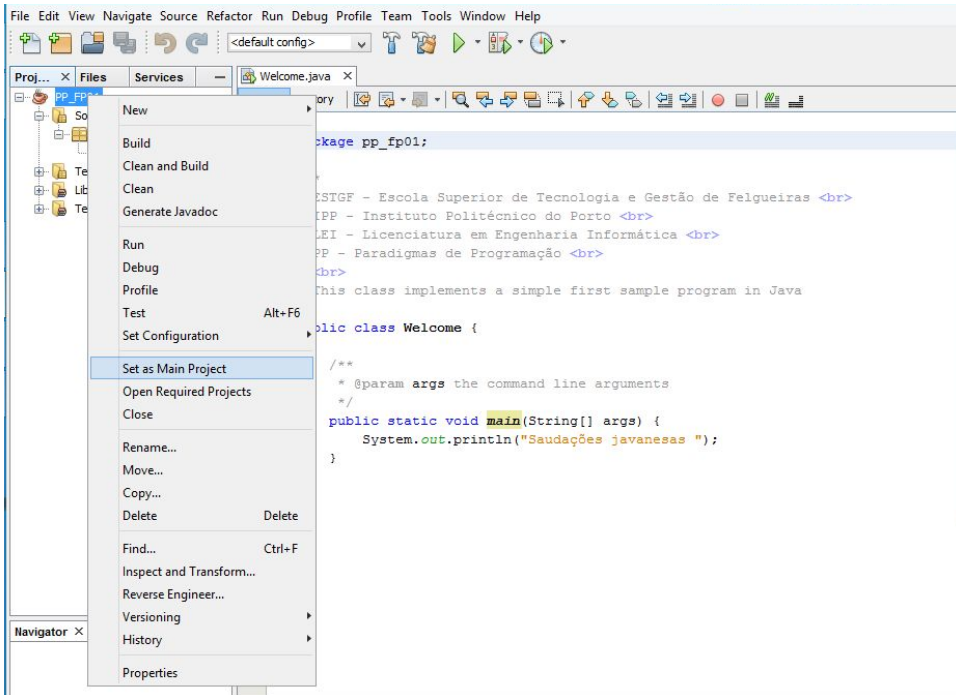
Exercício:

Experimente outras funcionalidades do IDE, tais como:

- **Tornar um projeto o projeto principal:**
  1. Selecione, no explorador de projetos (janela *Projects*), o projeto que pretende tornar o projeto

principal.

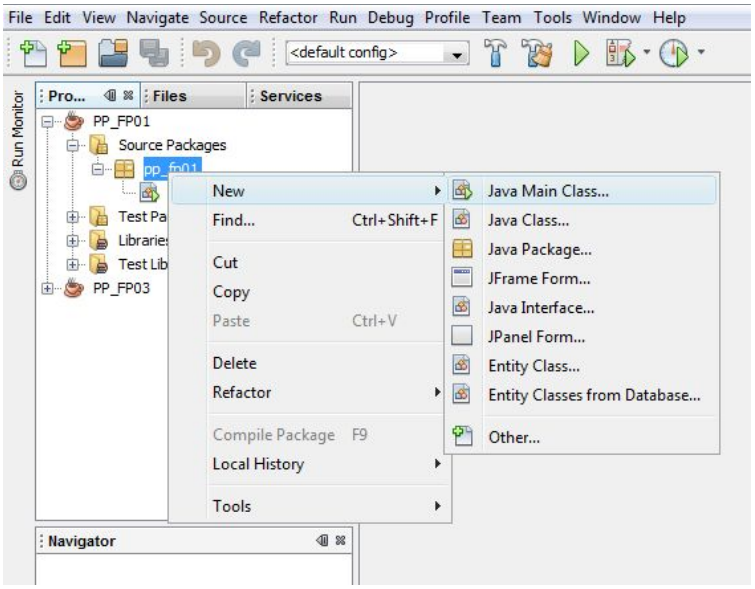
2. Sobre o projeto selecionado (estará *highlighted*, repare que na figura "PP_FP01" aparece com cor de fundo azul) pressione o botão do lado direito do rato.
3. No *pop-up menu* selecione a opção *Set as Main Project*.



4. Repare que o nome do projeto ficou com o nome a negrito (*bold*).

- **Acrescentar nova classe a executar (Main Class):**

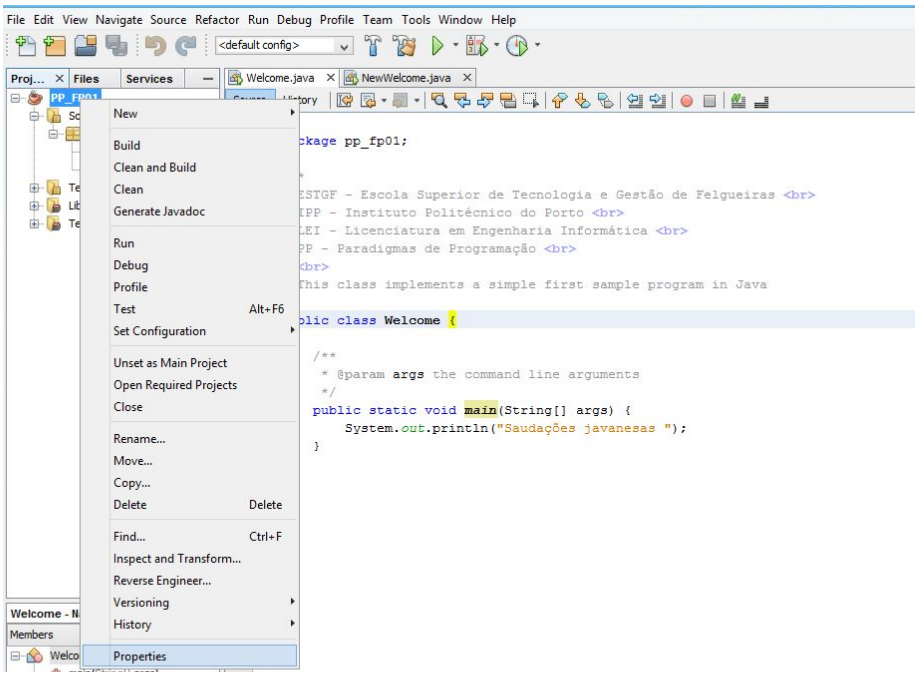
1. Selecione, no explorador de projetos (janela *Projects*), o *Source Package* ("pp_fp01") para o qual pretende acrescentar a classe a executar (*Main Class*)
2. Após pressionar o botão direito do rato sobre o *Source Package* ("pp_fp01"), selecione a opção **New > Java Main Class**
3. Na janela de criação da nova Main Class, mantenha os dados sugeridos e clique em *Finish*.



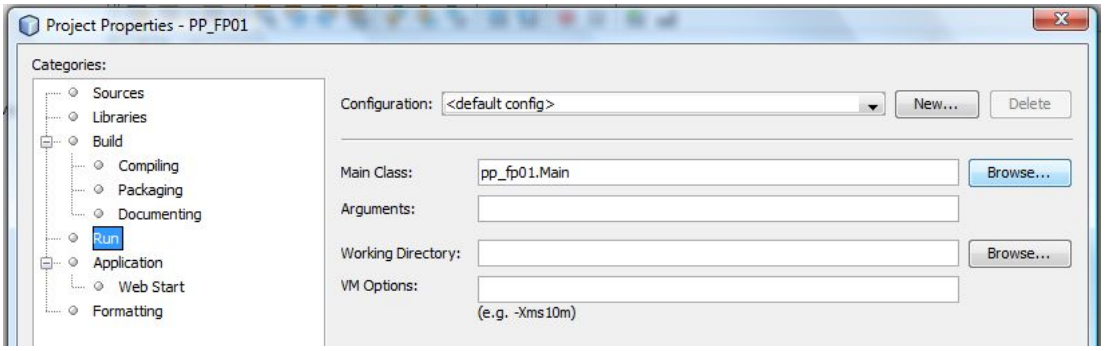
- **Alterar a classe a executar (*Main Class*):**

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.

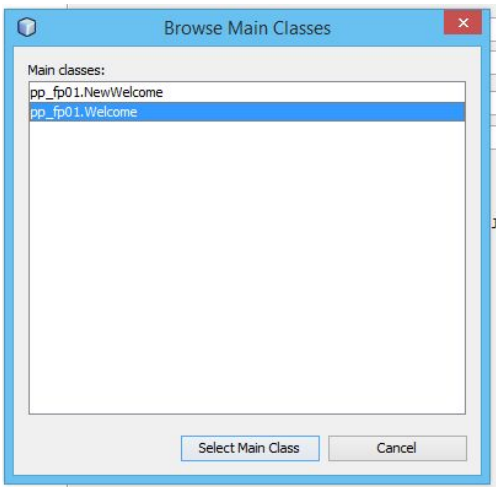
3. No *pop-up menu* selecione a opção *Properties*.



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

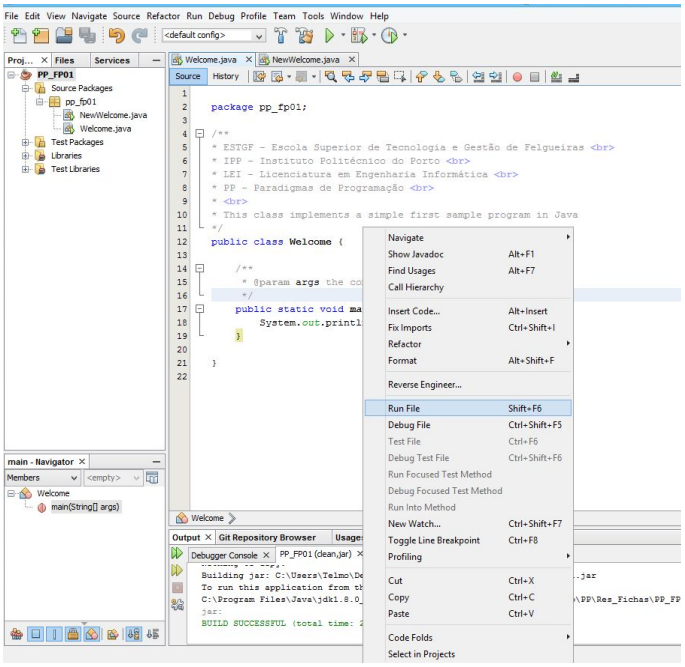


6. Na nova *dialog box* que aparece selecione a classe desejada.

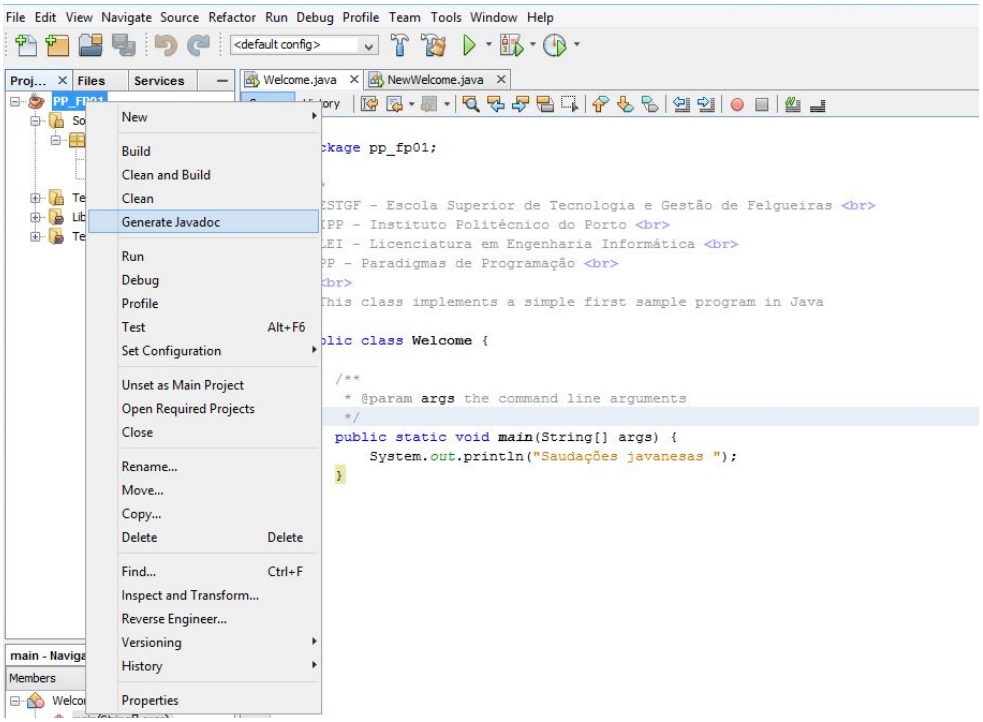


7. Uma vez seleccionada a classe desejada pressione o botão *Select Main Class*.

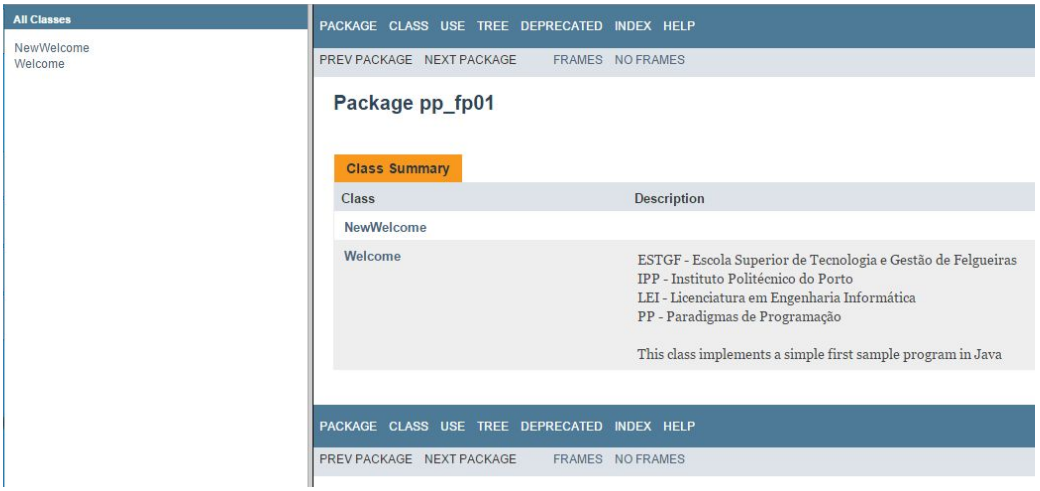
- **Executar um programa de forma rápida:**
    1. Sempre que for necessário executar um programa, pode fazê-lo a partir do botão direito do rato sob a classe (obrigatoriamente com um método **main()**) ao selecionar a opção *Run File*.



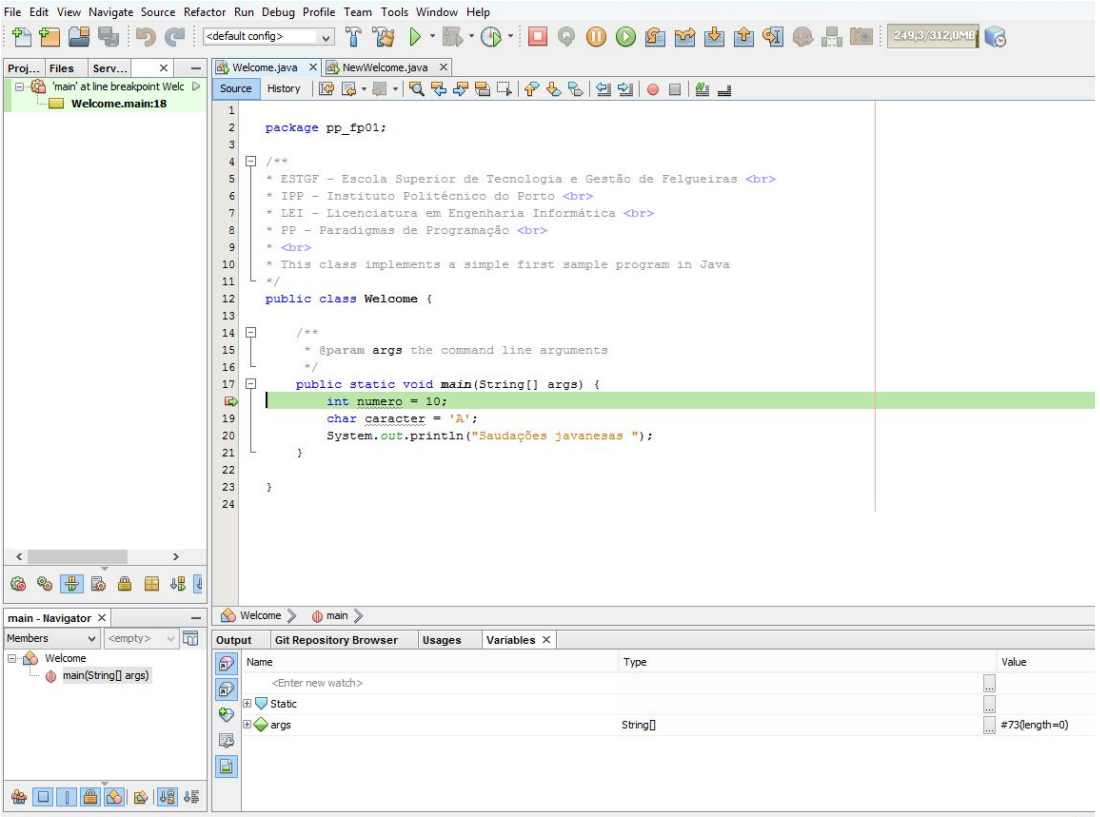
- **Gerar o Javadoc:**
    1. Selecione, no explorador de projetos (janela *Projects*), o projeto para o qual pretende gerar o Javadoc.
    2. Sobre o projeto selecionado (estará *highlighted*, repare que na figura "PP_FP01" aparece com cor de fundo azul) pressione o botão do lado direito do rato.
    3. No *pop-up menu* selecione a opção *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 selecionar 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 contínua a execução no método invocador.
- Run to Cursor - Avança entre instruções em que foram definidos *breakpoints*.

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.

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