# Annex UD02: IDE's Useful tools

1. IDE's Useful tools
    1. 1. Netbeans
    1. 2. Eclipse
2. Information sources

# 1. IDE's Useful tools

## 1.1. Netbeans

- Code templates.
    - `psvm + TAB` for `public static void main()`
    - `sout + TAB` for `System.out.println("")`
    - Consult more or customize: `Tools/Options/Editor/Code Templates`
- `Fix imports` (right click `Fix imports`)
- `Fix package` (when the file is not in the correct package, we can modify the code or folder structure)
- If a word is underlined in red, we can press `ALT+ENTER` to see the suggestions, for example:
    - Rename file based on class
- `Refactor` names of variables, methods, classes... (We change the name of a variable and modifies all occurrences in the source code. This does it in a way smarter than "Replace All")
- `CONTROL+SPACE` view method parameters and/or complete names.
- `ALT+SHIFT+F` to format the code
- Insert Code:
    - `ALT+INSERT` o botón derecho `Insert Code...`
        - Constructor
        - Getters
        - Setters
        - Getters and setters
        - ...
- ```
  Date fecha = new ~~Date~~ (2021, 2, 19); //Date constructor is deprecated
  ```
- When the name of a class appears in italics, it is because it is (`static`)
- Run with parameters: `Run/Set Project Configuration/Customize.../Run` you can define which is the main class in the `Main Class:` field with the format `&lt;paquete&gt;.&lt;Clase&gt;` and the parameters that the main method of said class will receive in the field `Arguments`: separating the parameters by spaces.
- Debug:
    - `Debug file` (debug file, `CTRL+SHIFT+F5`)
    - `Togle Line breakpoint` ((turn a breakpoint on/off, red square, `CTRL+F8`, caution on linux)
    - `Step over` (Next step, without going into details, `F8`)
    - `Step into` (Next step, going into details, `F7`)
    - `Continue` (Continue, `F5`)
    - `New Breakpoint...` (customize the conditions and place of the breakpoint `CTRL+SHIFT+F8`)
    - `New Watch...` (inspect the content of the variable or expression while debug the code)
  ```

- `TODO`: When our programs get big we want to leave notes of what that we "leave pending", that's what the `TODO` (from the English "to be done") is for, if we add in a comment the words `@todo`, `TODO`, `FIXME` (and some more that you can customize in `Tools/Options/Team/Action Items`) they will appear available in the window actions (`Window/Action items`). My recommendation on Netbeans is to modify the template so that it only recognizes when you type "TODO:" (with colon), and thus avoid confusion with usual words in Spanish such as "*méTODO*".

## 1.2. Eclipse

- Code templates.
    - `main + CONTROL+SPACE` and select `main method` for `public static void main()`
    - `sysou + CONTROL+SPACE` for `System.out.println("")`
    - Consult more or customize: `Window/Preferences/Java/Editor/Templates`
- Organize imports (right click `Source/Organize imports` o `CONTROL+SHIFT+O`)
- Fix package (when the file is not in the correct package, we can modify the code or folder structure).
- If a word is underlined in red, we can press `F2` to see the suggestions, for example:
    - Rename file based on class
- Refactor names of variables, methods, classes... (We change the name of a variable and modifies all occurrences in the source code. This does it in a way smarter than "Replace All").
- The completion of method parameters and/or completion of names appears automatically, once we have chosen the desired method we can toggle between the parameters with the `TAB` key.
- `CONTROL+SHIFT+F` to format the code
- Generate code automatically:
    - Inside the `Source` menu
        - Generate Getters and Setters...
        - Generate toString()...
        - Generate Constructor using fields...
        - Generate Constructor from Superclass...
- When the name of a class appears crossed out, it is because it is better not to use it and there are to look for an alternative (`dreprecated`)

```
Date fecha = new ~~Date~~ (2021, 2, 19); //Date constructor is deprecated
```

- When the name of a class appears in italics (in the invocations) it is because it is static (`static`)
- Run with parameters: `Run/Run Configurations` on the left you must choose your class within the `Java Application` section and on the right side you can define which is the main class in the `Main` tab and the parameters that the `main` method will receive from said class in the a `Arguments` tab, separating the parameters by spaces.
- Debug:
    - `Debug` (debug file, F11)
    - `Togle Line breakpoint` (activate/deactivate a breakpoint, blue circle, `CTRL+SHIFT+B`).
    - `Step over` (Next step, without going into details, `F6`).
    - `Step into` (Next step, going into details, `F5`).

    - `Continue` (Continue, `F8`)
    - I have not found a way to condition a `breakpoint` as it does Netbeans.
    - `Run/Watch` menu (inspect the content of the variable or expression while code is debugged).
- `TODO`: When our programs get big we want to leave notes of what that we "leave pending", that's what the `TODO` (from the English "to be done") is for, if we add in a comment the words `@todo`, `TODO`, `FIXME` (and some more that you can customize `Window/Preferences/General/Editors/Task tags`) will appear available in the actions window (`Window/Show view/Tasks`). Eclipse only recognizes the `TODO` that goes along with the single line comment `//TODO`. In addition, eclipse sets priorities and you can customize the presentation and priorities.

# 2. Information sources

- Wikipedia
- Code&Coke (Fernando Valdeón)
- Notes IES El Grao (Mª Isabel Barquilla?)
- Notes IOC (Marcel García)
- Notes José Luis Comesaña
- Notes IES Luis Vélez de Guevara 17-18 (José Antonio Muñoz Jiménez)