

by Mari Chelo

Development Environments

Block 1

Unit 2: General purpose environments. Geany and Visual Studio Code

1.2.1. Introduction

A development environment is a set of processes and programming tools that we use to create software. An integrated development environment or **IDE** is an environment in which processes and tools are coordinated, and they provide an ordered interface, and some suitable views for the software development process (or, at least, for writing the code, test it and pack it).

The first IDEs were born at the beginning of the 70s, but they became really popular along the 90s. Their first aim is to save time and have more reliability in software processes. They provide the programmer some useful components that increase the efficiency and reduce the coding time.

Some examples of these IDEs are Geany, Visual Studio Code, NetBeans, Eclipse, Visual Studio... In this unit we are going to focus on the first two, since they are general purpose environments, this is, they can be used for many different programming languages.

1.2.2. Functions and components of a development environment

As we should know, development environments are composed by some useful programming tools that help us meet our goals. These tools can be:

- **Source code editor:** This is an essential part of every IDE, since it lets us write the source code of our program. It usually highlights the code syntax, and may also have some additional features, such as code autocompletion, auto indentation, context help, access to API documentation, auto insertion of parentheses, square brackets...
- **Compilers:** This tool is in charge of translating our source code into machine code. Some IDEs lets us add many different compilers for many different languages, as we will see later.
- **Interpreters:** It is also possible that our IDE uses some kind of virtual machine or interpreter for some languages, in case our programming language needs it. So, the executable will not be run from the processor, but from an intermediate virtual machine or interpreter.
- **Debugger:** It is in charge of debugging our source code. In other words, it lets us examine step by step the execution of our program, check the values of the different variables and stop the program at any point.
- **GUI builder:** It simplifies the creation of graphical user interfaces, so that we can place the different controls (buttons, lists, menus...) wherever we want.

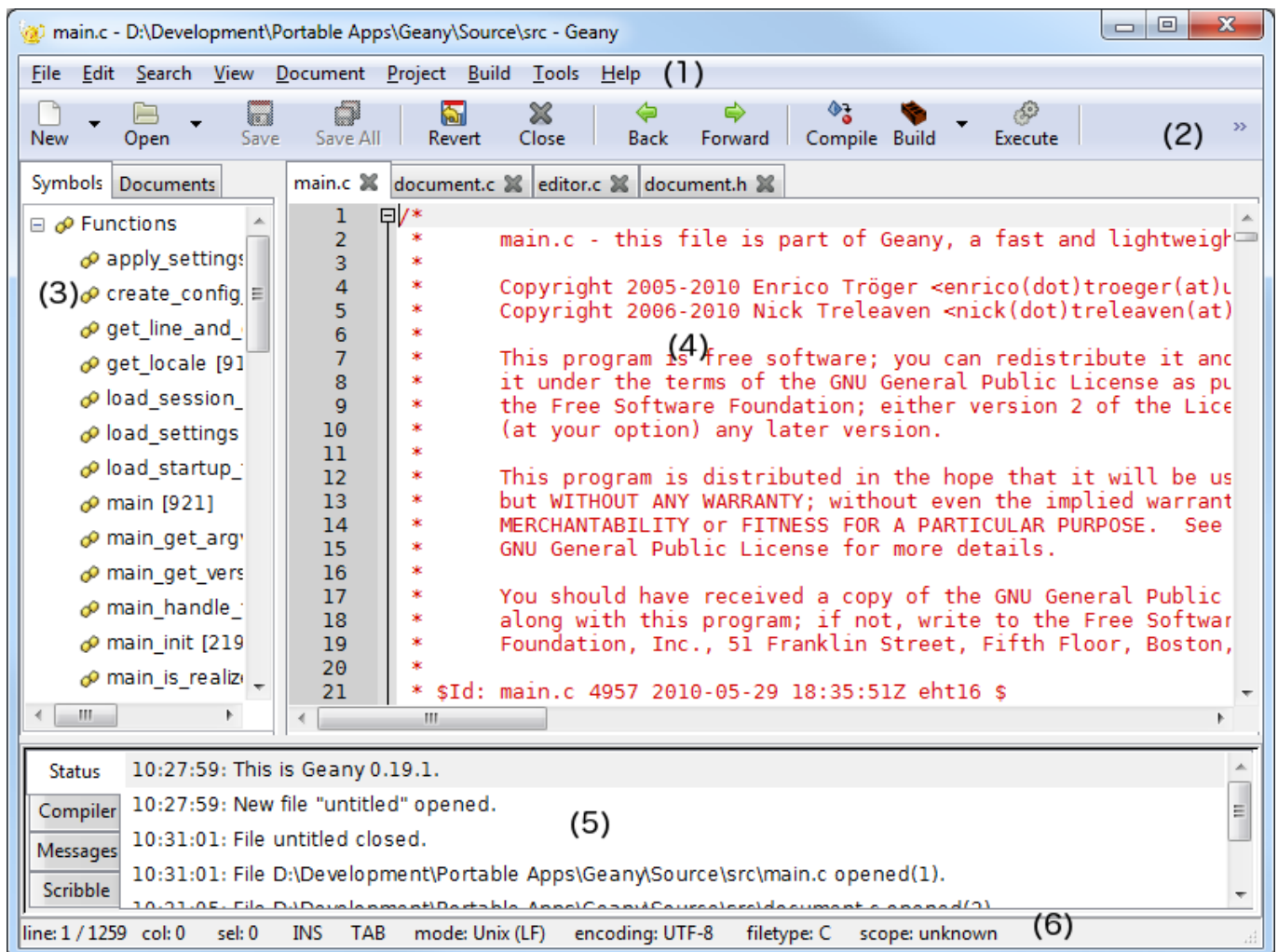
- **Version control:** With version control tools we can manage every change in the source code of our projects, so that we will have different versions of each source file, and we can go back and recover any older version at any time. It is an essential tool for teamwork.

1.2.3. Geany

Geany is an easy text editor with some basic features of an IDE. It can be downloaded for free from its official website www.geany.org, where we will find versions for Linux, Windows and Mac.

NOTE: regarding **Linux** (Ubuntu) users, it is better to install Geany from *Synaptic* package manager, which can be run from *System tools* section. Then, you can look for "geany" and install the corresponding package.

1.2.3.1 Work environment



Geany's work environment has the following sections:

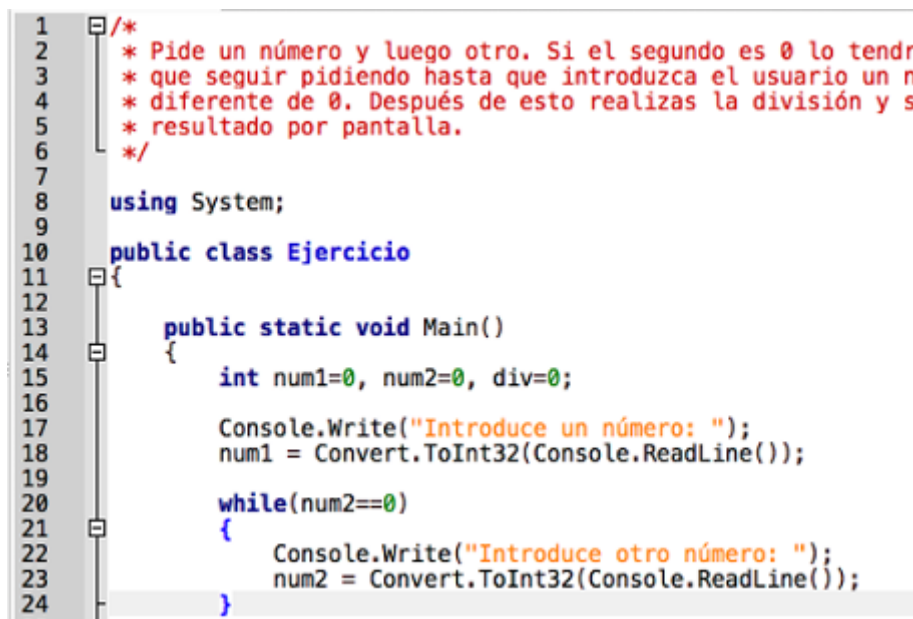
1. Menu.
2. Toolbar (optional)
3. Side bar with tabs:

1. Documents → List of documents.
2. Symbols → List of code symbols.
4. Edit window.
5. Message window, with the following tabs:
 1. Status → A list of status messages.
 2. Compiler → Compiler messages.
 3. Messages → General messages.
 4. Scribble → To make annotations.
6. Status bar, where we can see, among other things, the row and column we are editing, the file type, encoding...

1.2.3.2 Main features

1.2.3.2.1 Syntax highlighting

This is a very useful feature when we are typing code, since we can easily detect some basic typographic errors as we type the code. Besides, it helps us understand the whole code with a simple glance.

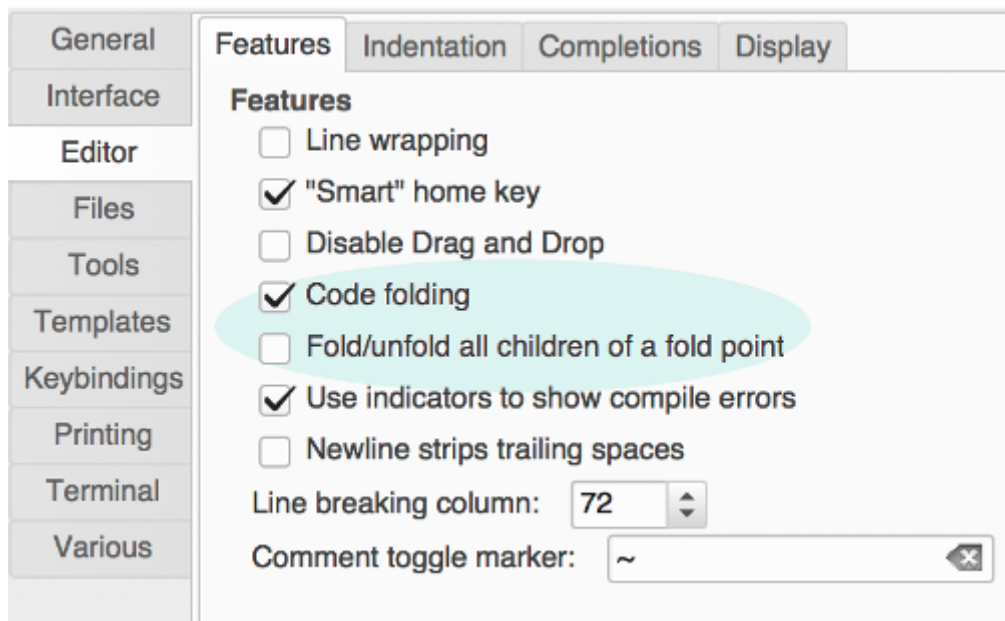


```
1  /*
2  * Pide un número y luego otro. Si el segundo es 0 lo tendri
3  * que seguir pidiendo hasta que introduzca el usuario un n
4  * diferente de 0. Después de esto realizas la división y si
5  * resultado por pantalla.
6  */
7
8  using System;
9
10 public class Ejercicio
11 {
12
13     public static void Main()
14     {
15         int num1=0, num2=0, div=0;
16
17         Console.Write("Introduce un número: ");
18         num1 = Convert.ToInt32(Console.ReadLine());
19
20         while(num2==0)
21         {
22             Console.Write("Introduce otro número: ");
23             num2 = Convert.ToInt32(Console.ReadLine());
24         }
25     }
26 }
```

As we can see in previous image, reserved words are written in a different color (blue). If we type a reserved word and it is not written in blue, then we may have not written it properly, so we can check it immediately, without waiting for the compilation error.

1.2.3.2.2 Code folding

Geany provides a basic code folding. This folding lets us show or hide some parts of the code, so that we can focus only in the parts that we are currently editing. This feature can be enabled or disabled from *Edit/Preferences* menu.



From this menu, we can also enable the possibility of folding and unfolding every nested foldings that are inside the code block that we are trying to fold/unfold. By default this option is disabled, so we have to fold/unfold every code block independently.

```

/*
 * Pide un número y luego otro. Si el segundo es 0 lo tendrás
 * que seguir pidiendo hasta que introduzca el usuario un número
 * diferente de 0. Después de esto realizas la división y sacas el
 * resultado por pantalla.
 */

using System;

public class Ejercicio
{
    public static void Main()
    {
        int num1=0, num2=0, div=0;

        Console.Write("Introduce un número: ");
        num1 = Convert.ToInt32(Console.ReadLine());

        while(num2==0)
        {
            Console.Write("Introduce otro número: ");
            num2 = Convert.ToInt32(Console.ReadLine());
        }

        div = num1 / num2;
        Console.WriteLine("El resultado es: {0}. ", div);
    }
}

```

```

/*
using System;

public class Ejercicio
{
    public static void Main()
    {
        int num1=0, num2=0, div=0;

        Console.Write("Introduce un número: ");
        num1 = Convert.ToInt32(Console.ReadLine());

        while(num2==0)
        {
            div = num1 / num2;
            Console.WriteLine("El resultado es: {0}. ", div);
        }
    }
}

/*
using System;

public class Ejercicio
{
    public static void Main()
    {
}

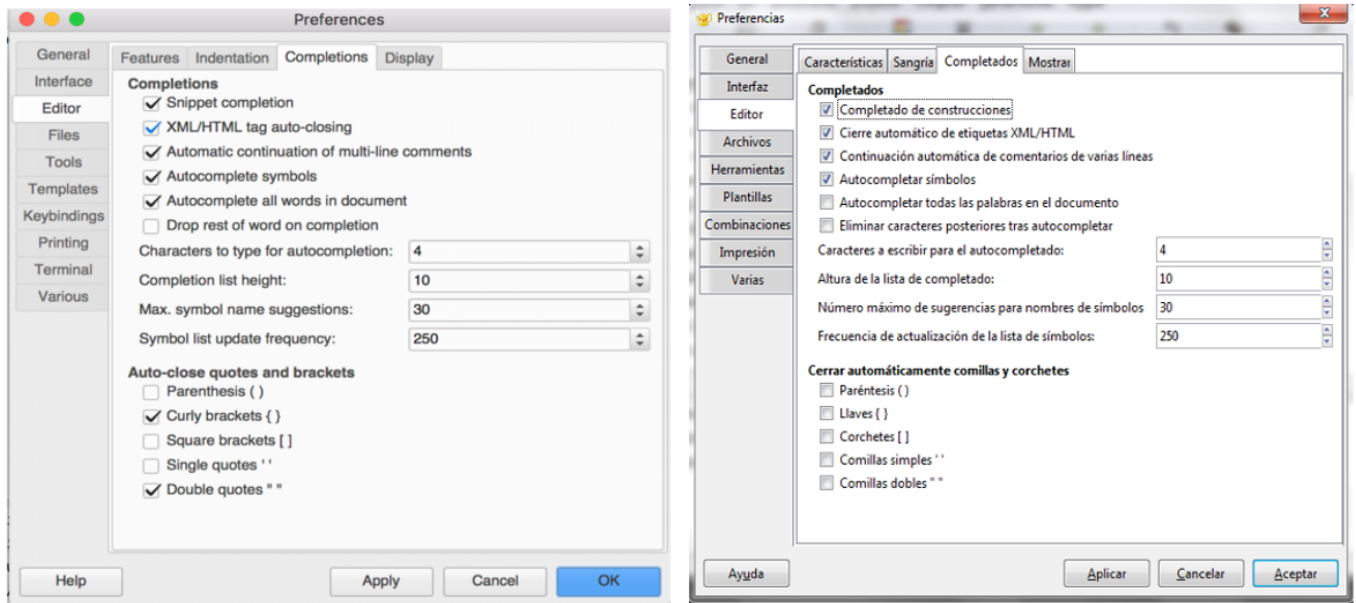
/*
using System;

public class Ejercicio
{

```

1.2.3.2.3 Word and symbol auto completion

This option can be configured in *Edit/Preferences* menu, under *Editor* option, in *Completions* tab.



With this feature enabled, we will see a list of suitable options whenever we type the beginning of a reserved word, so we will be able to complete it just typing Enter. By default it is enabled to show the list when we type the first 4 letters of the reserved word. Besides, we can also configure the automatic completion of parentheses, curly braces, square brackets and so on. This is really useful if we usually forget to close these symbols.

1.2.3.2.4 Auto closing XML/HTML tags

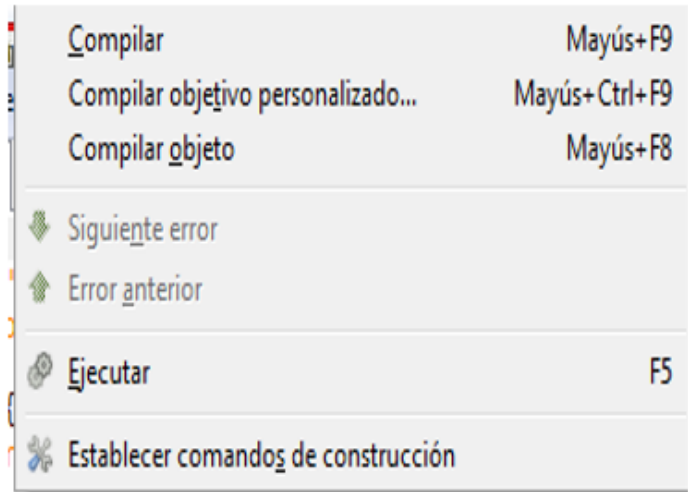
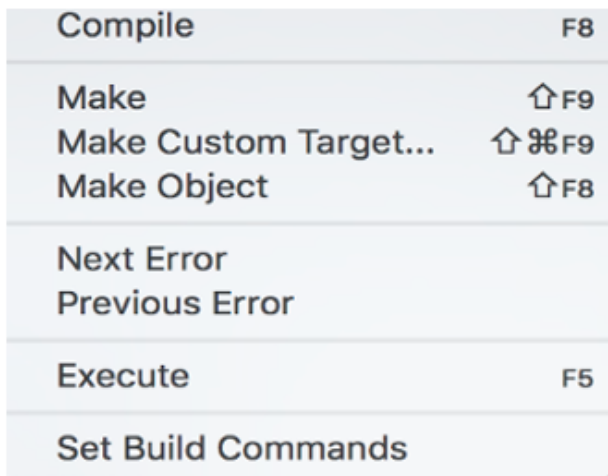
In the same configuration menu shown before, we can also activate the auto closing of XML/HTML tags when we type the corresponding opening tag.

1.2.3.2.5 Multi language support

Geany lets us choose among many different programming languages, such as C, Java, PHP, HTML, Python, C# and others, with their own auto completion and syntax highlighting.

1.2.3.2.6 Creating, compiling and testing our code

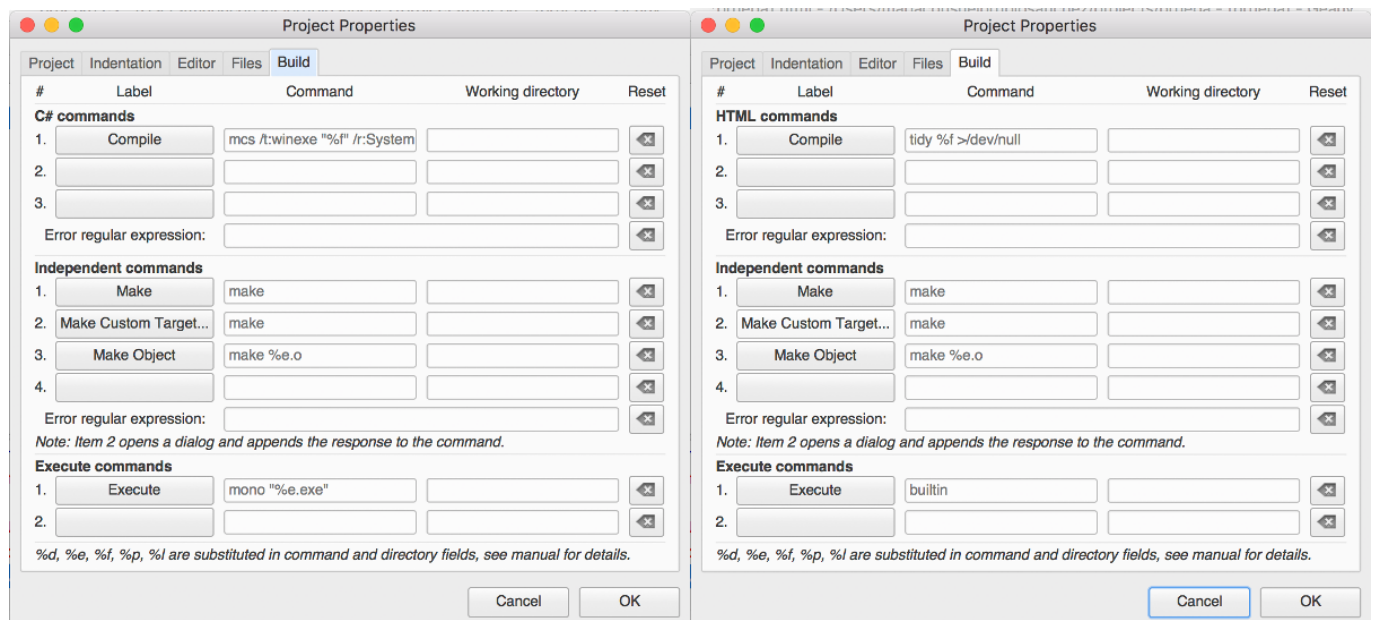
Once we have finished typing our source code, we can test it from Geany, by using the *Build* menu.



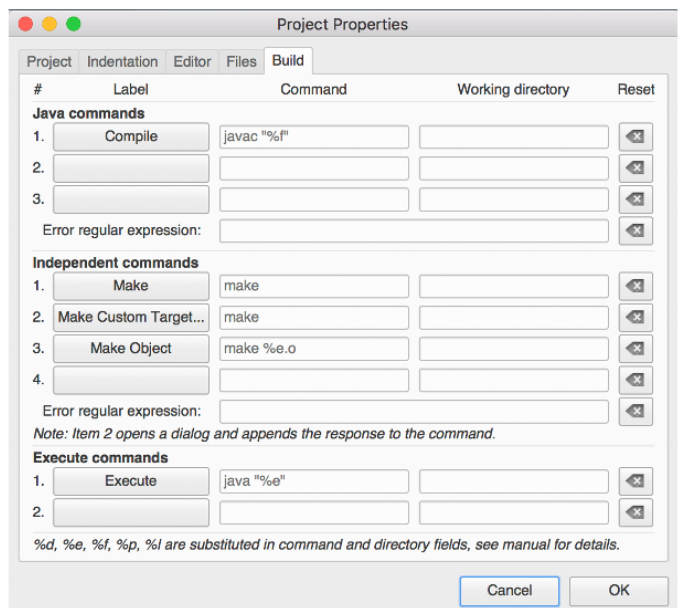
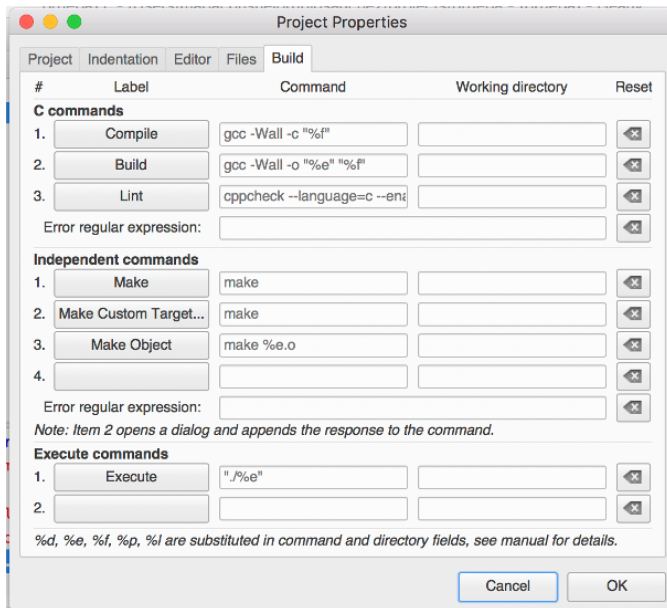
If we go to *Build/Set Build Commands* option, we can see the compiler that will be run when we try to compile the program.

Whenever we create a new document with Geany, it is important to save it with an appropriate name and extension before typing any code in it. This way, Geany will fill the corresponding compilation and execution commands automatically with the appropriate tools. Besides, only after detecting the type of source code that we are typing, code highlighting and auto completion will be activated.

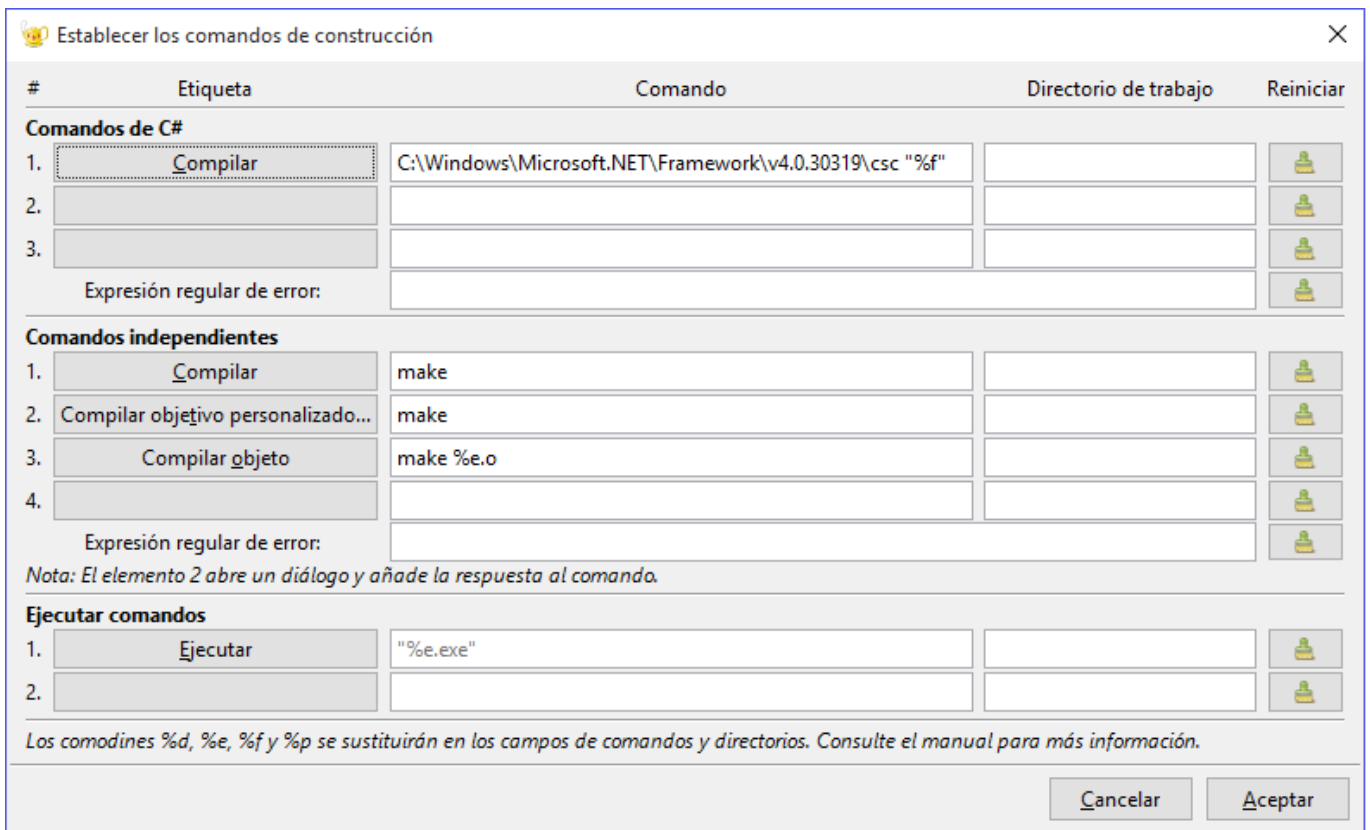
There are auto-detected compilers for C# and HTML (this last one is an interpreted language, so it has no compiler):



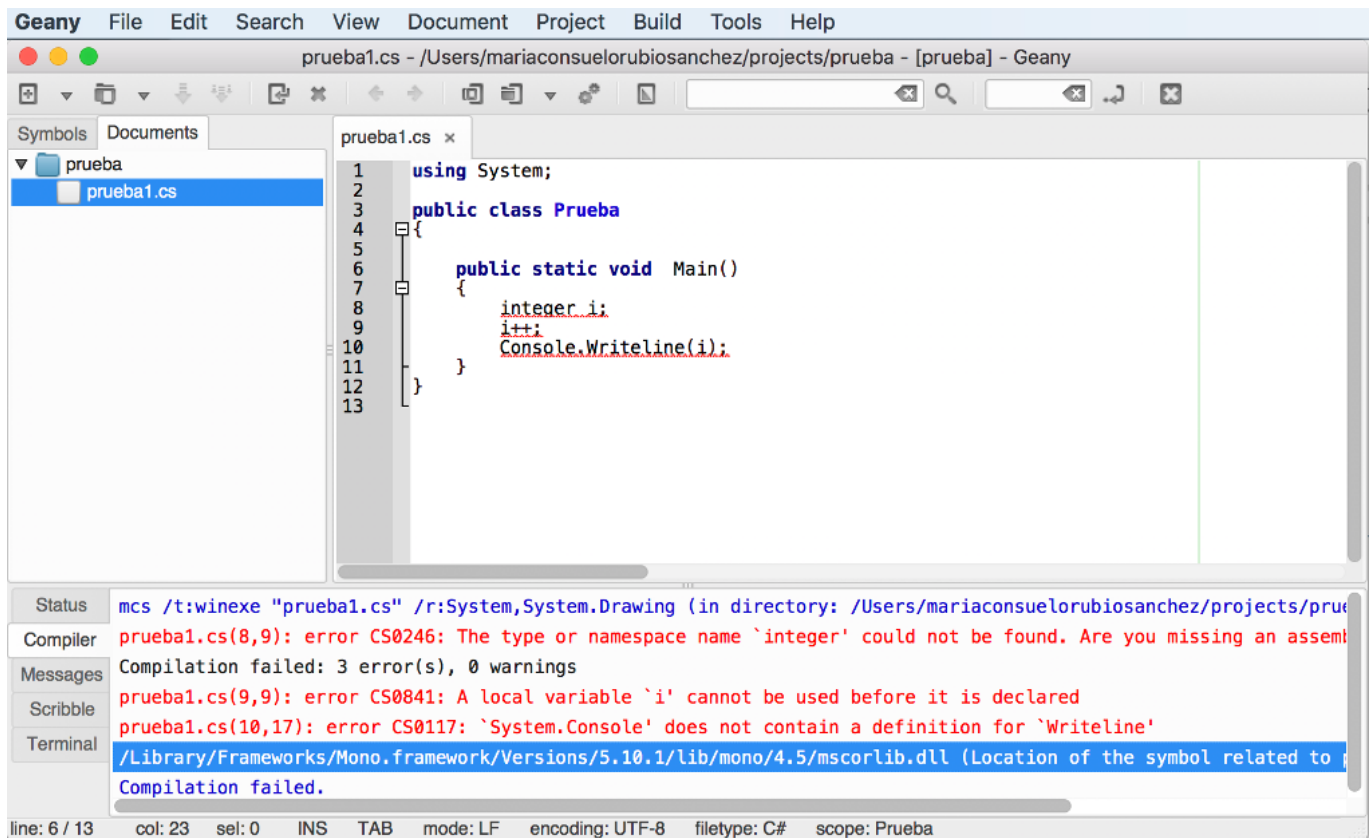
There are also auto-detected copilers for C and Java:



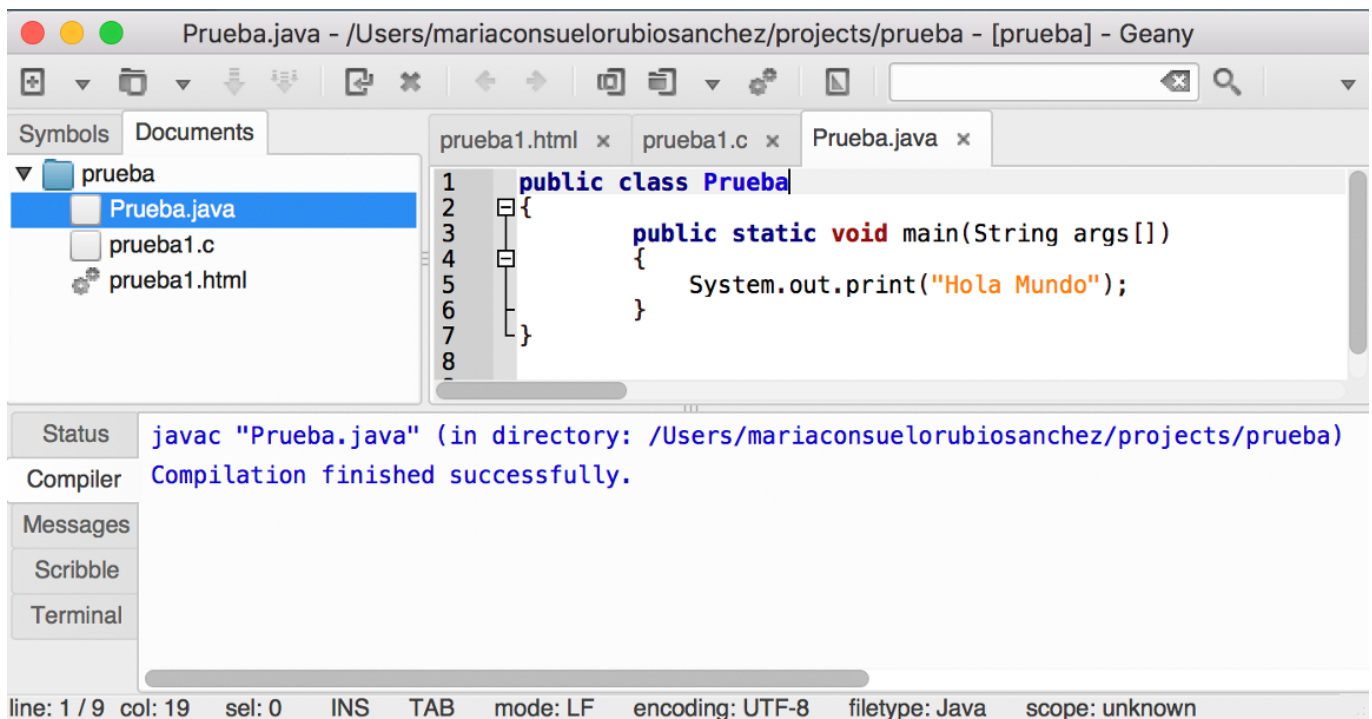
If we want to use Windows' C# compiler instead of *mono*, we must manually fill the corresponding fields as we can see in the following image. Previously, we must locate the folder in which this compiler called **csc** exists, and copy the full path in the *Compile* field.



We can also configure the editor to underline with red color every compilation error. This option is available under *Edit/Preferences* menu, inside *Editor* option, in *Features* tab (see image in subsection 1.2.3.2.2).



As we can see in the image above, we can check every compilation error in the compiler messages window at the bottom of the window.



In previous image we can see how to use different source code types, and each one has its own assigned compiler, so we can compile each document with its associated compiler.

Proposed exercises:

1.2.3.1. Open Geany with the source file "Test.java" created in previous unit, and see how it highlights the code. Also, check or enable the word auto completion and try it by adding a new line with the code to write "World" (just copy the line that says "Hello" and replace it with "World").

1.2.3.2. Compile the source file of previous exercise. If there has been no errors, run it.

1.2.3.3. Create a file called "test.c" with Geany. Remember: after choosing *File-New*, then choose *File-Save As* and save it with .c extension, so that code will be highlighted as we type it. Then, write the following code that you have also written in previous unit:

```
#include <stdio.h>

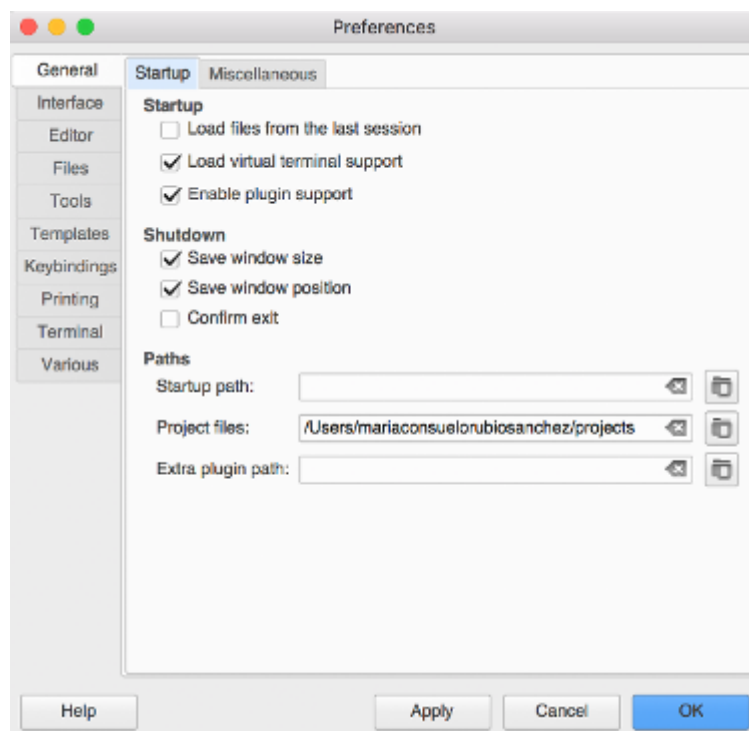
int main()
{
    printf("Hello");
    return 0;
}
```

Check the code auto completion, compile the program and run it.

1.2.3.3 Some additional useful features

1.2.3.3.1. Geany startup

When we run Geany, every open document from our previous session will be open again. This feature can be configured in *Edit/Preferences* menu, under *General* option, in *Startup* tab.



1.2.3.3.2 Clone documents

In the *Document* menu there is a *Clone* option that lets us not only copy the text of current document, but also its properties, such as cursor position. It can be really helpful for creating many source files with a similar structure.

1.2.3.3.3 Keybindings

There are many shortcuts or key bindings available in Geany, as we can also see in many other applications. In the following tables we can see some of the most common ones. These shortcuts can also be configured from *Edit-Preferences* menu, in the *Combinations* tab.

File shortcuts

Option	Shortcut	Action
New	Ctrl-N	Creates a new file
Open	Ctrl-O	Opens a file
Save	Ctrl-S	Saves current file
Save All	Ctrl-Shift-S	Saves every open file
Close All	Ctrl-Shift-W	Closes every open file
Close	Ctrl-W	Closes current file
Exit	Ctrl-Q	Exits Geany

Edit shortcuts

Option	Shortcut	Action
Undo	Ctrl-Z	Undoes last action
Redo	Ctrl-Y	Redoes last action
Delete current lines	Ctrl-K	Deletes currently selected line(s)
Delete until the end	Ctrl-Shift-Del	Deletes from current position until the end of the line
Delete from the beginning	Ctrl-Shift-BackSpace	Deletes from the beginning of the line until current cursor position
Duplicate line or selection	Ctrl-D	Duplicates current line or selection
Complete fragment	Tab	If we type some known structure such as <i>if</i> or <i>for</i> and press this key, it will complete the structure.

Clipboard shortcuts

Option	Shortcut	Action
Cut	Ctrl-X	Cuts current selection and copies it into the clipboard
Copy	Ctrl-C	Copies current selection into the clipboard
Paste	Ctrl-V	Pastes clipboard's content into current cursor position.
Cut line	Ctrl-Shift-X	Cuts currently selected line(s) and copies it/them into the clipboard
Copy line	Ctrl-Shift-C	Copies currently selected line(s) into the clipboard

Selection shortcuts

Option	Shortcut	Action
Select all	Ctrl-A	Selects all the text from current document
Select word	Ctrl-Shift-W	Selects the word in which cursor is currently placed
Select paragraph	Alt-Shift-P	Selects the paragraph in which cursor is currently placed
Select line	Alt-Shift-L	Selects the line in which cursor is currently placed

Format shortcuts

Option	Shortcut	Action
Swap Uppercase-Lowercase	Ctrl-Alt-U	Swaps uppercase and lowercase letters in current selection. If there are both uppercase and lowercase letters in the selection, then it turns everything into lowercase.
Comment- Uncomment line	Ctrl-E	Comments current line, or uncomments it if it is already commented.
Increase indentation	Ctrl-I	Adds a new indentation level to currently selected line(s)
Decrease indentation	Ctrl-U	Removes an indentation level from currently selected line(s)

Search shortcuts

Option	Shortcut	Action
Find	Ctrl-F	Opens the <i>find</i> dialog
Find next	Ctrl-G	Go to next result
Find previous	Ctrl-Shift-G	Go to previous result
Replace	Ctrl-H	Opens the <i>replace</i> dialog
Find in files	Ctrl-Shift-F	Opens the <i>find in files</i> dialog
Find usage	Ctrl-Shift-E	Finds every occurrence of current word or selection in every open document.
Find usage in current document	Ctrl-Shift-D	Same as before, but it only checks current document.
Highlight all	Ctrl-Shift-M	Highlights every occurrence of current word or selection in current document.

Besides these shortcuts, you can check all the possible ones [here](#)

Proposed exercises:

1.2.3.4. Try some of previous Geany shortcuts with any of the source files created so far.









1.2.4 Visual Studio Code

Visual Studio Code is a lightweight (but powerful) code editor available for Windows, MacOSX and Linux. It has native support for Javascript, TypeScript and Node.js, and a wide variety of extensions that let us work with (almost) any other programming language, such as C, C#, Java, Python, PHP, Go, etc.

You can download it from <https://code.visualstudio.com>, where you can also find some useful extensions for Java, C, C#...

Top Extensions

Enable additional languages, themes, debuggers, commands, and more. VS Code's growing community shares their secret sauce to improve your workflow.

 Python ms-python 13.1M Linting, Debugging (multi-threaded, remote), Inte...	 C/C++ ms-vscode 9.8M C/C++ IntelliSense, debugging, and code	 Debugger for Chrome msjsdiag 9.6M Debug your JavaScript code in the Chrome browser,...	 ESLint dbaeumer 9.5M Integrates ESLint JavaScript into VS Code.
 vscode-icons robertohuertasm 7.2M Icons for Visual Studio Code	 C# ms-vscode 7.1M C# for Visual Studio Code (powered by OmniSharp).	 TSLint eg2 6.1M TSLint for Visual Studio Code	 GitLens — Git superc... eamodio 6.0M Supercharge the Git capabilities built into Visua...

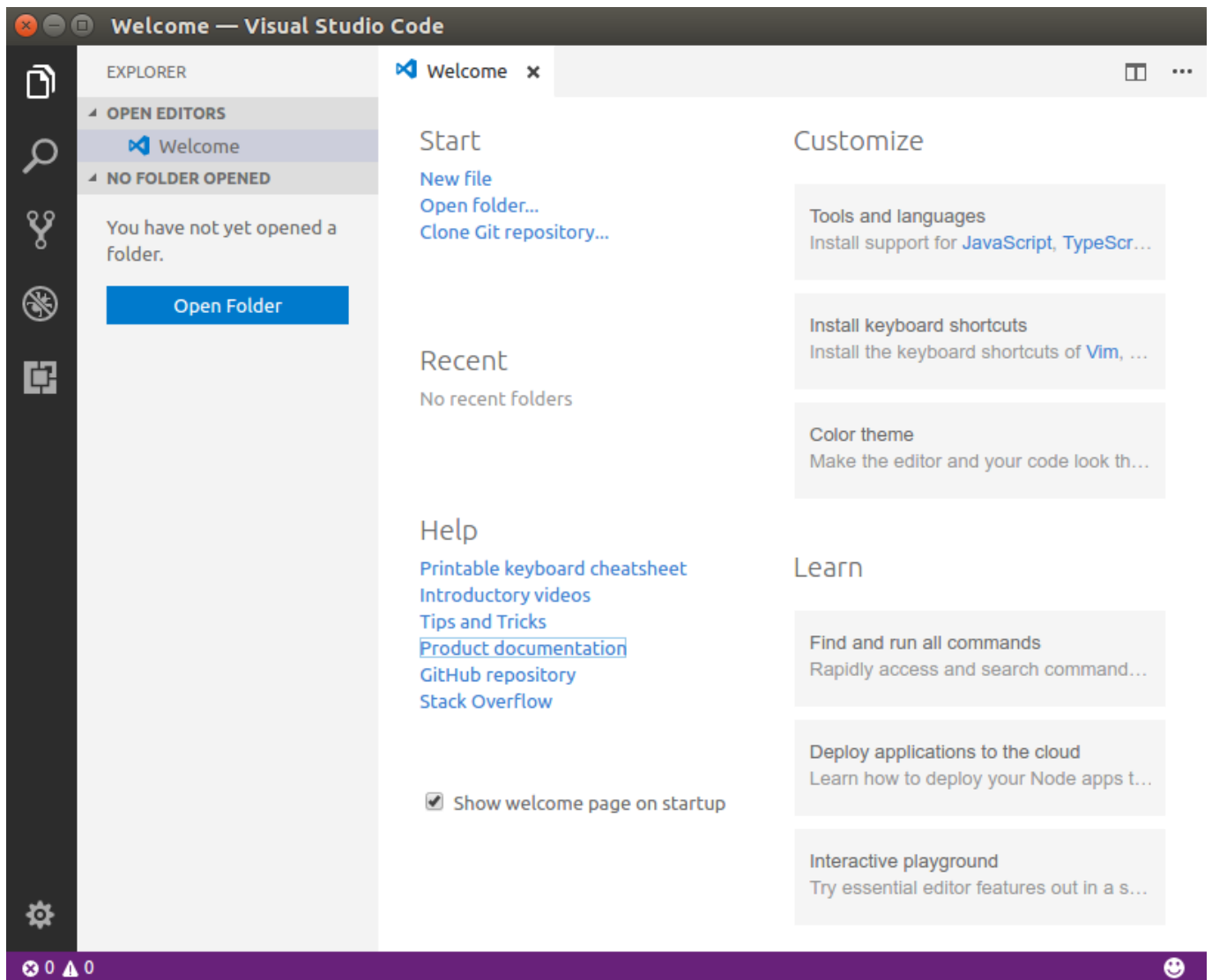
NOTE: again, for **Linux** (Ubuntu) users, once you download the *.deb* package from the official web site, you need to open a terminal and go to the download folder. Then, type this command and you will have Visual Studio Code available in the *Programming* section:

```
sudo dpkg -i <visual_studio_file_name>
```

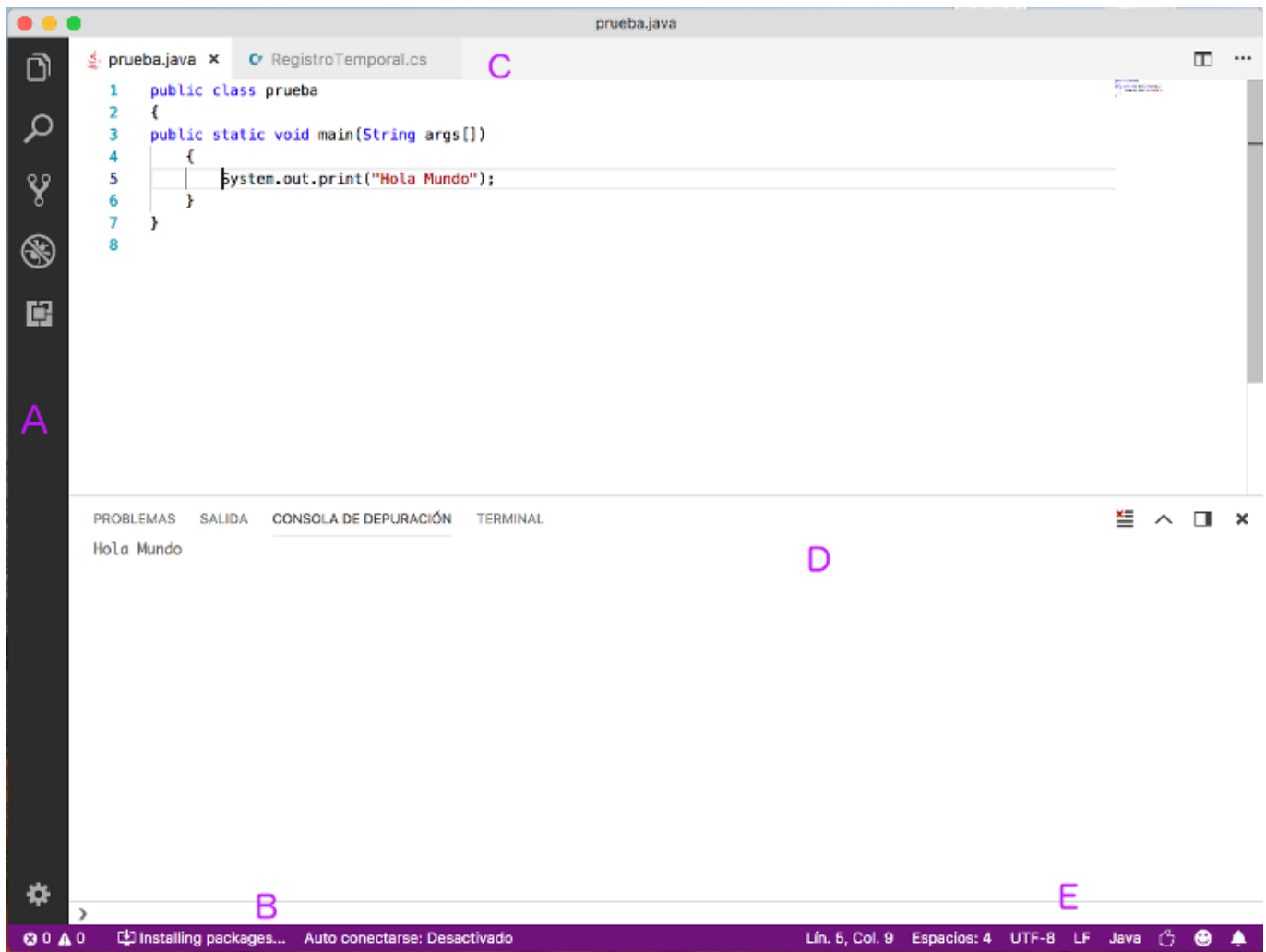
where `<visual_studio_file_name>` is the file name of the downloaded file.

1.2.4.1 Work environment

Once it is installed, you can see the welcome screen with some of the basic options



If we create a new file we will see the usual work environment, which is divided into 5 sections.



- **Editor (C):** Main edition area. You can open as many editors as you need.
- **Margin bar (B):** It contains some useful information about the file explorer, code errors or warnings, etc.
- **Status bar (E):** It shows information about current project, or currently open files.
- **Activities bar (A):** It has some options:
 - File explorer, to locate the files to be edited. This option shows/hides a left panel to browse current folder
 - Search tool, to look for some words or regular expressions in our source files.
 - Source code control (to communicate with version control tools, such as Git repositories)
 - Debugger
 - Extension manager, from which we can install new extensions to this IDE, and check/config the ones that are already installed.
- **Panels (D):** Below the editor you can see some different panels, such as errors/warnings, or a terminal to type some commands from current folder. This panel can be moved to the right if we want to have more vertical space.

Every time we open Visual Studio Code, it shows the last snapshot before it was closed the last time, with the same open files and so on.

Changing the color theme

If you want to change the default color theme for Visual Studio Code, you must go to *File/Preferences* menu (or *Code/Preferences* menu if you are running it under Mac OSX), and then choose the *Color Theme* option. Then, you can choose among a wide variety of options. The most popular ones are the Visual Studio's Dark or Light modes.

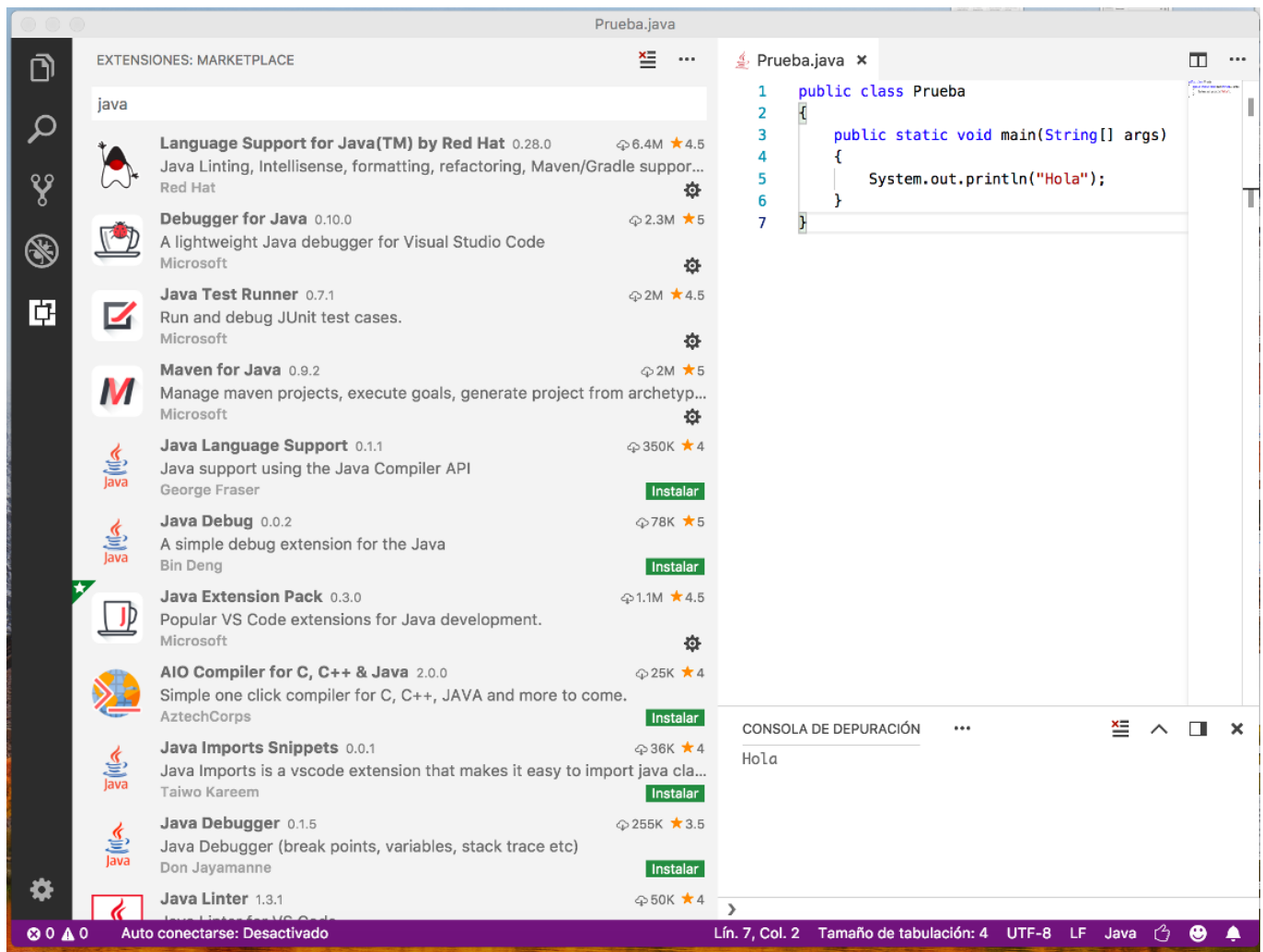
1.2.4.2 Installing extensions

1.2.4.2.1 Edit, compile and test Java programs

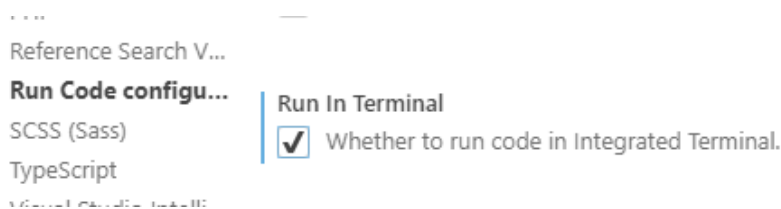
If we want to start coding, we need to install the appropriate extension(s) for the language we are going to work with. To do this, we click on the extension icon in the activities bar (left bar), and we look for the desired language in the text field. In our case, if we want to work with Java, we just type *Java* and then we will see a lot of extensions related with this language.

There is a useful extension called **Java Extension Pack**, from Microsoft, that contains the most common extensions to work with Java:

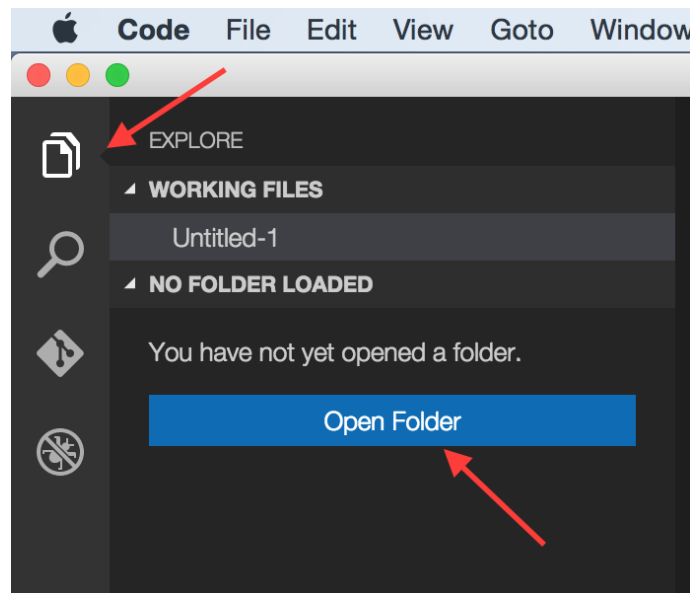
- Language Support for Java(TM) by Red Hat
- Debugger for Java
- Java Test Runner
- Maven for Java



Besides, you may need the **Code Runner** extension, a general extension that lets us run programs written in many different programming languages, such as C, Python, Java and so on. You should also go to *File/Preferences/Settings* menu, go to the *Extensions* section and choose *Run Code configuration*. Then, look for *Run in terminal* option and check it.



Next, we just need to open the folder in which we are going to store the source files. In the left panel, we will see the contents of this folder, and we will be able to open an existing file, or create a new one from the *File* menu.



In order to run a single file, we can just right click on its source code and choose *Run Code* option from the context menu (this option is provided by the *Code Runner* extension, so you must install this extension in order to use this option).

Proposed exercises:

1.2.4.1. Install the extensions needed to work with Java in Visual Studio Code.

1.2.4.2. Open the file *Test.java* from Visual Studio Code. Add a new line to write some more text in the screen, compile the program and test it. See the results in the bottom panel (debug console).

1.2.4.2.2 Edit, compile and test other programming languages

In the same way that we have tried our Java example, we can run programs in many other languages, such as C, Python and so on. We just need to download the appropriate extension(s) to work with this language, and run the source files with *Code Runner*.

However, there are some concrete languages whose configuration is quite more difficult. This is the case of C#, for instance. We need to install the appropriate compiler (either Mono or .NET), but we need to follow some additional steps in each case in order to set up everything properly. In this case, it may be a better choice to use Geany for single source files, or Visual Studio for complex projects.

Proposed exercises:

1.2.4.3. Open the file *test.c* in VS Code and try to run it and see the "Hello" message in the terminal.

1.2.4.3 Main features

Visual Studio Code has the same features seen before with Geany regarding:

- Syntax highlighting
- Code folding

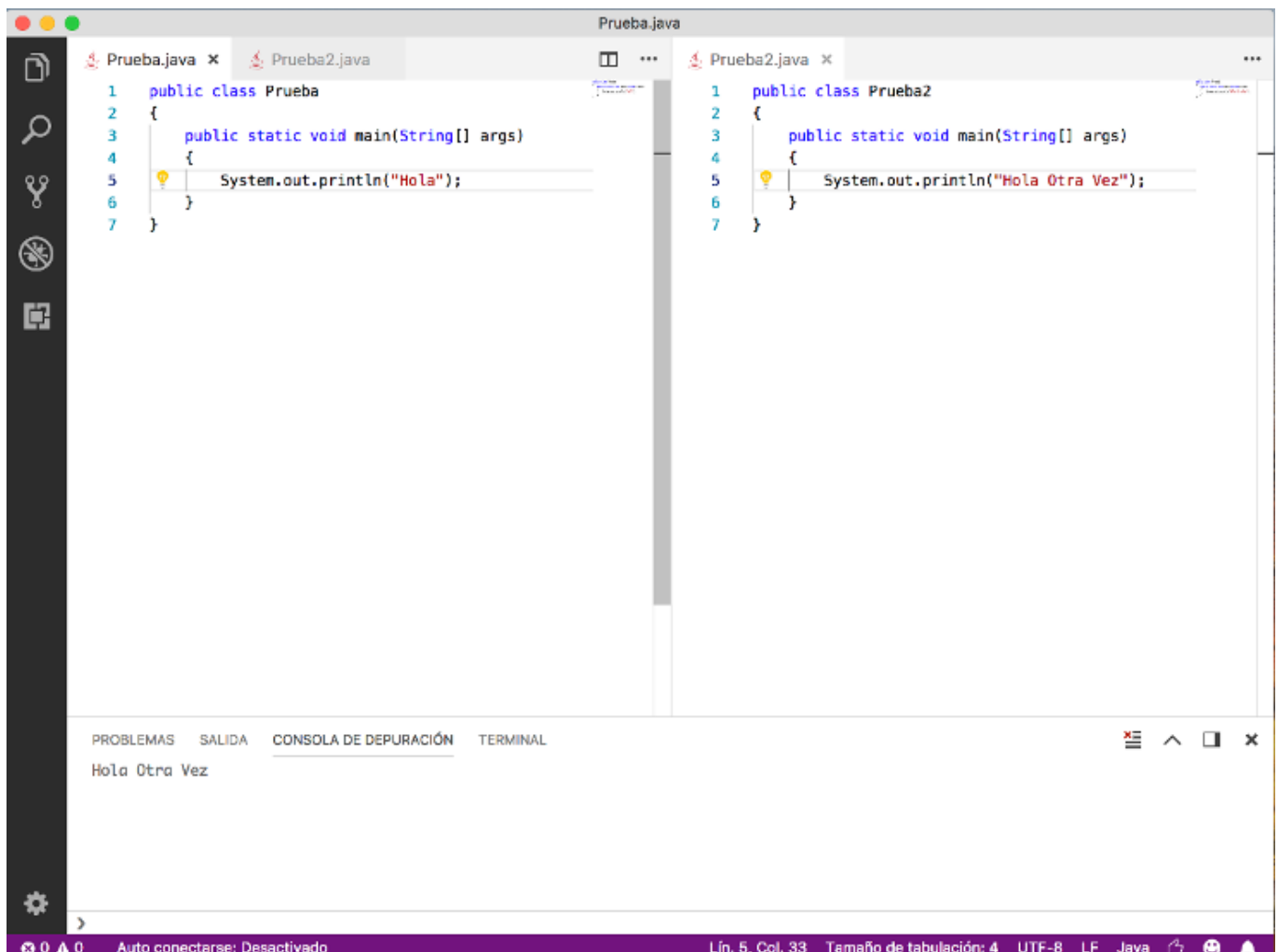
- Word auto completion
- XML/HTML tags auto closing
- etc.

The configuration can be changed in *File/Preferences/Configuration* (in MacOSX, this menu turns into *Code/Preferences/Configuration*).

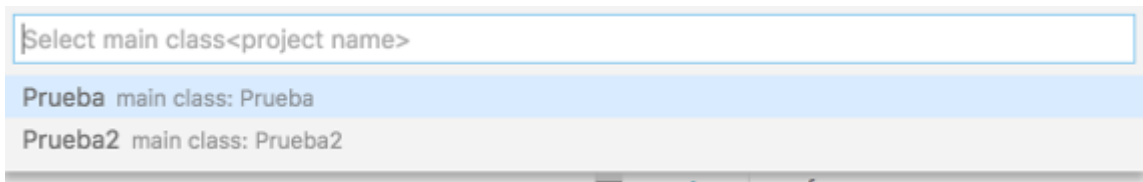
1.2.4.4 Additional features

1.2.4.4.1 Side by side editing

This feature lets us edit two or more source files at the same time, having each one in a different column. This way, we can edit and compile them without having to close any of them. This option is activated by clicking on the 2 column icon in the upper right bar.

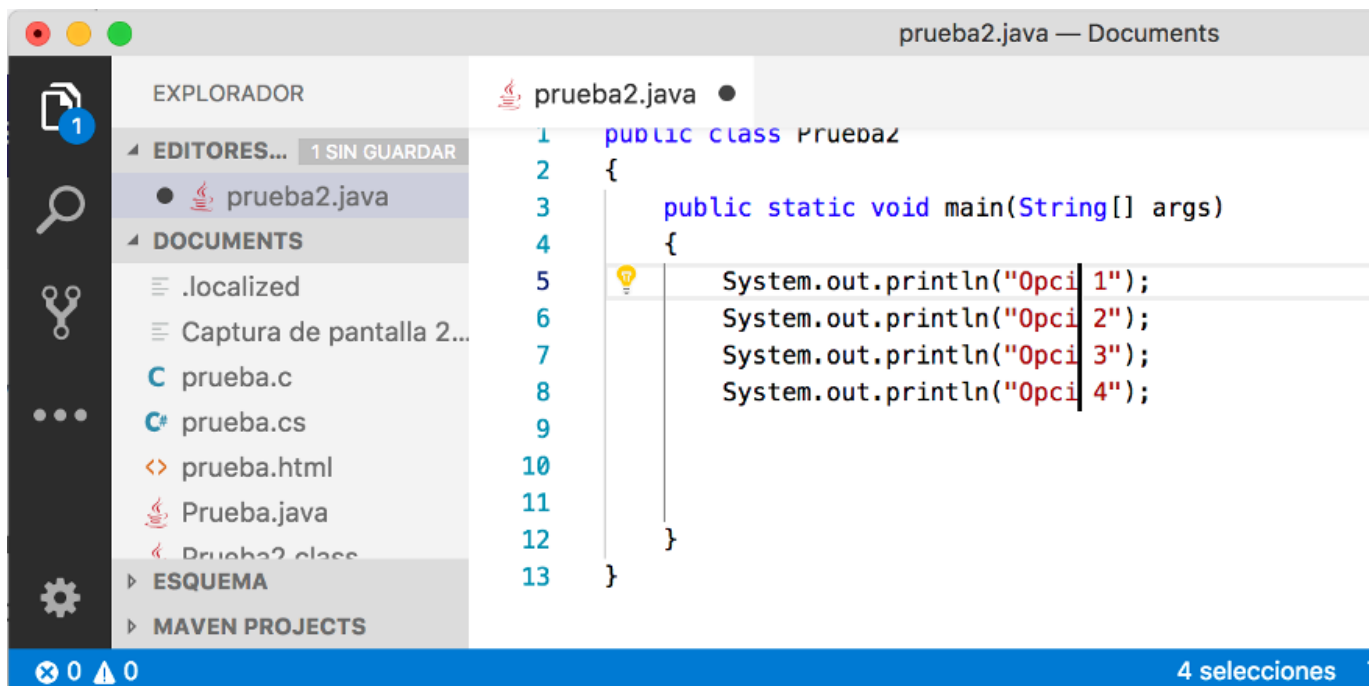


If we want to compile any of these files, as we have more than one active document, a popup menu will be shown to choose the file to be compiled.



1.2.4.4.2 Multi-cursor

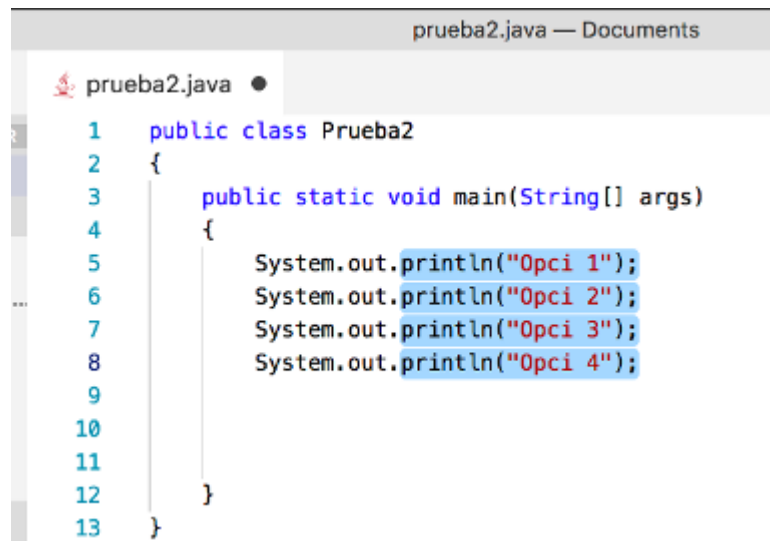
We can also activate more than one cursor at the same time, by typing **Alt + Clic** in the position in which we want to add a new cursor. Once we have all the cursors set, everything that we type or delete will be added/removed at the same time to/from every cursor position.



Whenever we want to remove all the additional cursors and leave just one of them, we must type *Escape*.

1.2.4.4.3 Box selection

If we want to make this type of selection, we must hold **Shift+Alt** as we select the desired column(s) with the mouse.



```

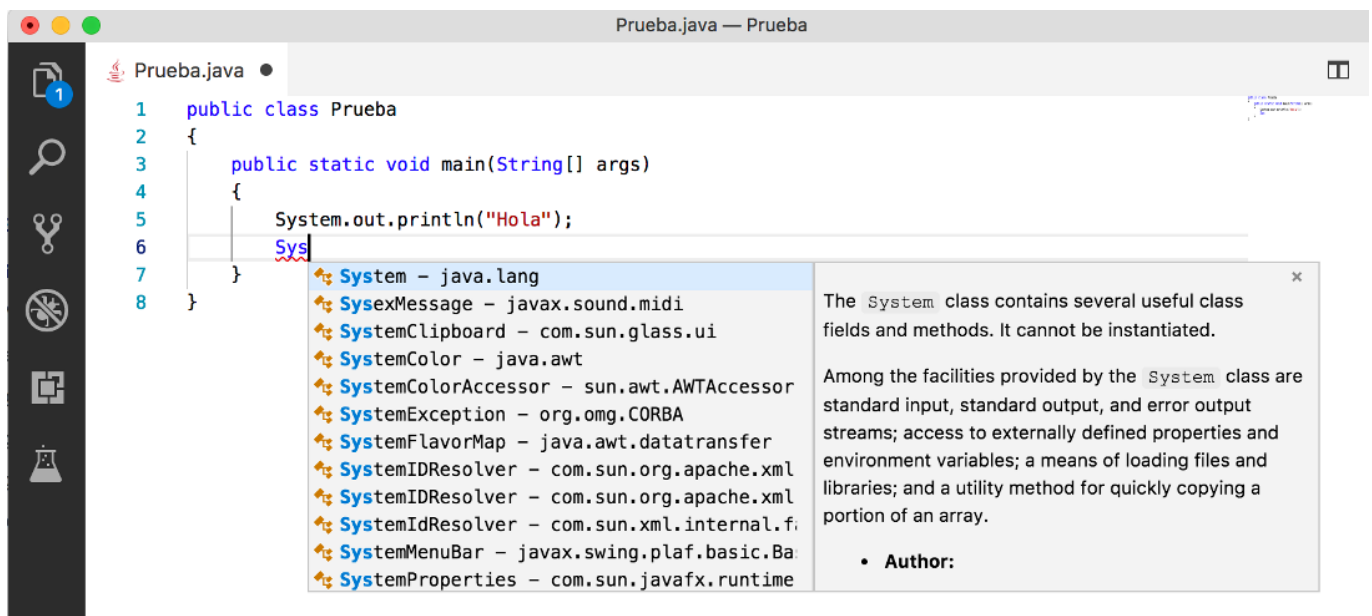
prueba2.java — Documents

prueba2.java •
1  public class Prueba2
2  {
3      public static void main(String[] args)
4      {
5          System.out.println("Opci 1");
6          System.out.println("Opci 2");
7          System.out.println("Opci 3");
8          System.out.println("Opci 4");
9
10
11
12      }
13  }

```

1.2.4.4.4 IntelliSense

Besides word auto completion, VS Code provides an additional structure list for languages such as Java or C# (among others).



1.2.4.4.5 Code formatting

There is an option to format the currently selected code (or the whole file). This option is in the context menu of the editor (by right clicking on the editor panel), with the name *Format Document*. It formats the code according to the language we are using. For instance, for a Java program we will have this format:

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

We can see that the opening braces have been placed next to the line that creates the block (instead of putting them in the next line). This is the format that Java source files usually have.

Proposed exercises:

1.2.4.5. Open the source file *Test.java* and format the document. Check the resulting source code.

1.2.4.4.6 Auto save

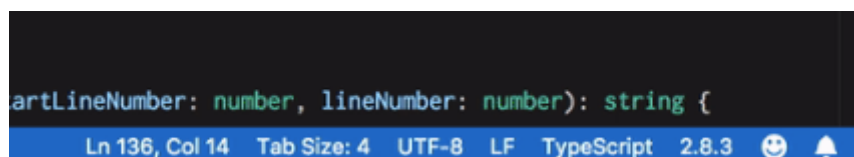
By default, VS Code asks us to manually save our changes, but there is an option in the *File* menu called *Auto Save*. If it is enabled, the IDE will periodically save our work (or every time the editor loses the focus).

1.2.4.4.7 Hot Exit

Every time we exit VS Code without saving changes, this state is automatically saved, and when we open the IDE again, we will keep all our unsaved changes.

1.2.4.4.8 Auto detecting indentation

Whenever we open a source file, its own indentation is automatically detected and used, instead of the one set by default in VS Code. This option is shown (and can be modified) in the status bar (*Tab size*).



Seleccionar acción

Aplicar sangría con espacios
Indent Using Spaces

Aplicar sangría con tabulaciones
Indent Using Tabs


Detectar sangría del contenido
Detect Indentation from Content

Convertir sangría en espacios
Convert Indentation to Spaces

Convertir sangría en tabulaciones
Convert Indentation to Tabs

1.2.4.5 Keybindings

In the welcome screen we have a link with a complete list of shortcuts for VS Code.



Visual Studio Code

Keyboard shortcuts for Windows

General

Ctrl+Shift+P, F1	Show Command Palette
Ctrl+P	Quick Open, Go to File...
Ctrl+Shift+N	New window/instance
Ctrl+Shift+W	Close window/instance
Ctrl+,	User Settings
Ctrl+K Ctrl+S	Keyboard Shortcuts

Basic editing

Ctrl+X	Cut line (empty selection)
Ctrl+C	Copy line (empty selection)
Alt+ ↑ / ↓	Move line up/down
Shift+Alt+ ↑ / ↓	Copy line up/down
Ctrl+Shift+K	Delete line
Ctrl+Enter	Insert line below
Ctrl+Shift+Enter	Insert line above
Ctrl+Shift+\	Jump to matching bracket
Ctrl+] / [Indent/outdent line
Home / End	Go to beginning/end of line
Ctrl+Home	Go to beginning of file
Ctrl+End	Go to end of file
Ctrl+↑ / ↓	Scroll line up/down
Alt+PgUp / PgDn	Scroll page up/down
Ctrl+Shift+[Fold (collapse) region
Ctrl+Shift+]	Unfold (uncollapse) region
Ctrl+K Ctrl+[Fold (collapse) all subregions
Ctrl+K Ctrl+]	Unfold (uncollapse) all subregions
Ctrl+K Ctrl+0	Fold (collapse) all regions
Ctrl+K Ctrl+J	Unfold (uncollapse) all regions
Ctrl+K Ctrl+C	Add line comment
Ctrl+K Ctrl+U	Remove line comment
Ctrl+/	Toggle line comment
Shift+Alt+A	Toggle block comment
Alt+Z	Toggle word wrap

Navigation

Ctrl+T	Show all Symbols
Ctrl+G	Go to Line...
Ctrl+P	Go to File...
Ctrl+Shift+O	Go to Symbol...
Ctrl+Shift+M	Show Problems panel
F8	Go to next error or warning
Shift+F8	Go to previous error or warning
Ctrl+Shift+Tab	Navigate editor group history
Alt+ ← / →	Go back / forward
Ctrl+M	Toggle Tab moves focus

Search and replace

Ctrl+F	Find
Ctrl+H	Replace
F3 / Shift+F3	Find next/previous
Alt+Enter	Select all occurrences of Find match
Ctrl+D	Add selection to next Find match
Ctrl+K Ctrl+D	Move last selection to next Find match
Alt+C / R / W	Toggle case-sensitive / regex / whole word

Multi-cursor and selection

Alt+Click	Insert cursor
Ctrl+Alt+ ↑ / ↓	Insert cursor above / below
Ctrl+U	Undo last cursor operation
Shift+Alt+I	Insert cursor at end of each line selected
Ctrl+I	Select current line
Ctrl+Shift+L	Select all occurrences of current selection
Ctrl+F2	Select all occurrences of current word
Shift+Alt+→	Expand selection
Shift+Alt+←	Shrink selection
Shift+Alt+ (drag mouse)	Column (box) selection
Ctrl+Shift+Alt+ (arrow key)	Column (box) selection
Ctrl+Shift+Alt+PgUp/PgDn	Column (box) selection page up/down

Rich languages editing

Ctrl+Space	Trigger suggestion
Ctrl+Shift+Space	Trigger parameter hints
Shift+Alt+F	Format document
Ctrl+K Ctrl+F	Format selection
F12	Go to Definition
Alt+F12	Peek Definition
Ctrl+K F12	Open Definition to the side
Ctrl+.	Quick Fix
Shift+F12	Show References
F2	Rename Symbol
Ctrl+K Ctrl+X	Trim trailing whitespace
Ctrl+K M	Change file language

Editor management

Ctrl+F4, Ctrl+W	Close editor
Ctrl+K F	Close folder
Ctrl+↖	Split editor
Ctrl+ 1 / 2 / 3	Focus into 1 st , 2 nd or 3 rd editor group
Ctrl+K Ctrl+ ↖ / →	Focus into previous/next editor group
Ctrl+Shift+PgUp / PgDn	Move editor left/right
Ctrl+K ← / →	Move active editor group

File management

Ctrl+N	New File
Ctrl+O	Open File...
Ctrl+S	Save
Ctrl+Shift+S	Save As...
Ctrl+K S	Save All
Ctrl+F4	Close
Ctrl+K Ctrl+W	Close All
Ctrl+Shift+T	Reopen closed editor
Ctrl+K Enter	Keep preview mode editor open
Ctrl+Tab	Open next
Ctrl+Shift+Tab	Open previous
Ctrl+K P	Copy path of active file
Ctrl+K R	Reveal active file in Explorer
Ctrl+K O	Show active file in new window/instance

Display

F11	Toggle full screen
Shift+Alt+0	Toggle editor layout (horizontal/vertical)
Ctrl+ = / -	Zoom in/out
Ctrl+B	Toggle Sidebar visibility
Ctrl+Shift+E	Show Explorer / Toggle focus
Ctrl+Shift+F	Show Search
Ctrl+Shift+G	Show Source Control
Ctrl+Shift+D	Show Debug
Ctrl+Shift+X	Show Extensions
Ctrl+Shift+H	Replace in files
Ctrl+Shift+J	Toggle Search details
Ctrl+Shift+U	Show Output panel
Ctrl+Shift+V	Open Markdown preview
Ctrl+K V	Open Markdown preview to the side
Ctrl+K Z	Zen Mode (Esc Esc to exit)

Debug

F9	Toggle breakpoint
F5	Start/Continue
Shift+F5	Stop
F11 / Shift+F11	Step into/out
F10	Step over
Ctrl+K Ctrl+I	Show hover

Integrated terminal

Ctrl+`	Show integrated terminal
Ctrl+Shift+`	Create new terminal
Ctrl+C	Copy selection
Ctrl+V	Paste into active terminal
Ctrl+↑ / ↓	Scroll up/down
Shift+PgUp / PgDn	Scroll page up/down
Ctrl+Home / End	Scroll to top/bottom

Other operating systems' keyboard shortcuts and additional unassigned shortcuts available at aka.ms/vscodekeybindings

1.2.5 Bibliography

<https://www.geany.org/manual/current/index.html>

<https://code.visualstudio.com/>