

PRÁCTICA 4:

Visual Studio Code

MIGUEL ANGEL GIRALDO POLANCO



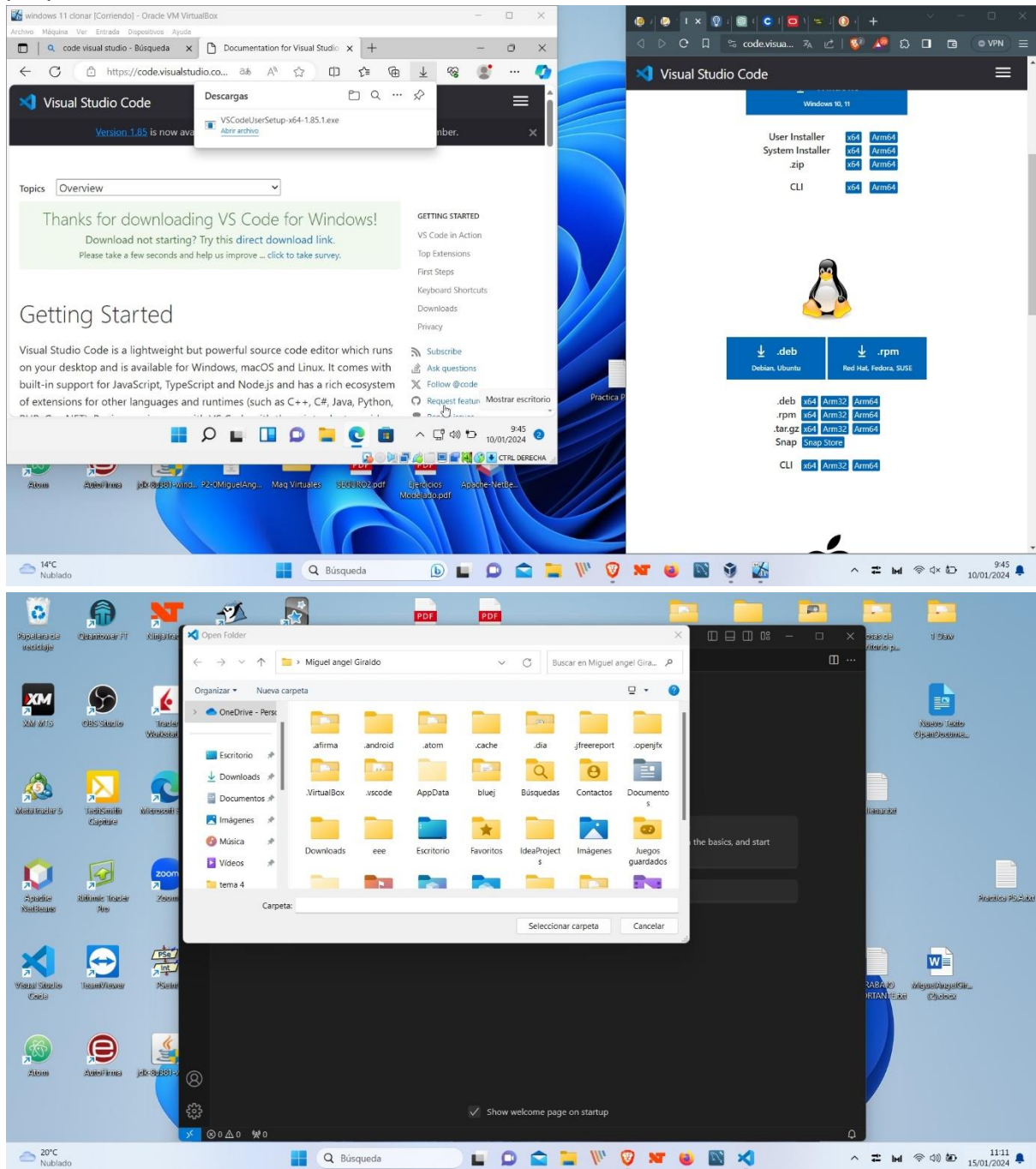
ENTORNOS DE DESARROLLO
GRADO SUPERIOR EN DESARROLLO DE APLICACIONES WEB
2023-2024

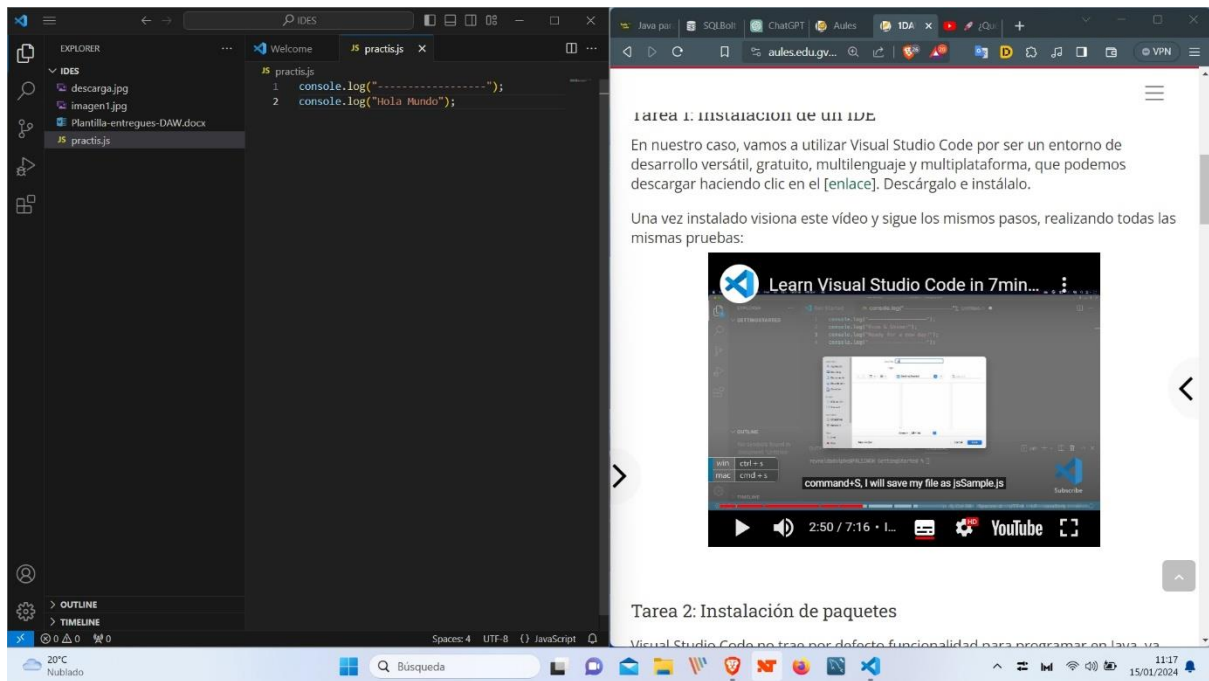
ÍNDICE

1.- INDICE	Pag 2.
2.- Instalación de un IDE	Pag 3.
3.- Instalación de paquetes	Pag 4.
4.- Get Started	Pag 5.
5.- Nuestro primer proyecto.....	Pag 6.
6.- Importando un proyecto	Pag 7.
7.- Trapecio	Pag 10.

Instalación de un IDE

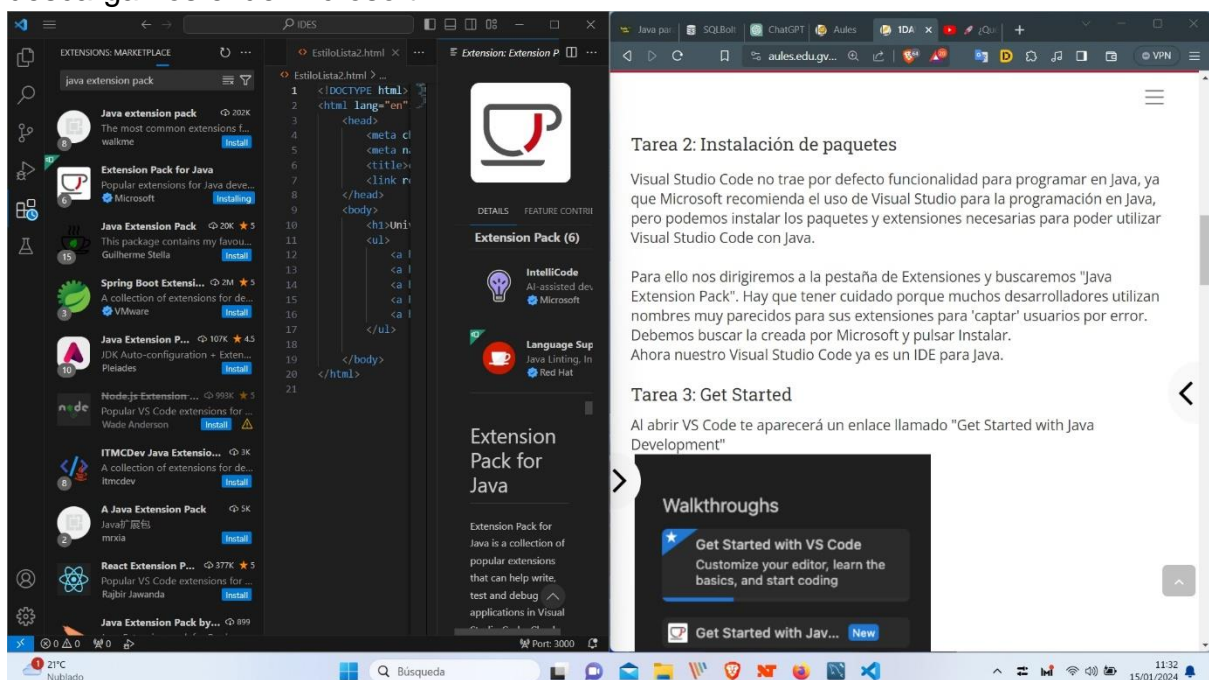
Siguiendo las instrucciones del video de AULES después de la instalación seguí todos los pasos para aprender un poco como crear proyectos, saber donde ver y detectar errores, ver que lenguaje esta siendo utilizado como descargar e instalar paquetes etc.





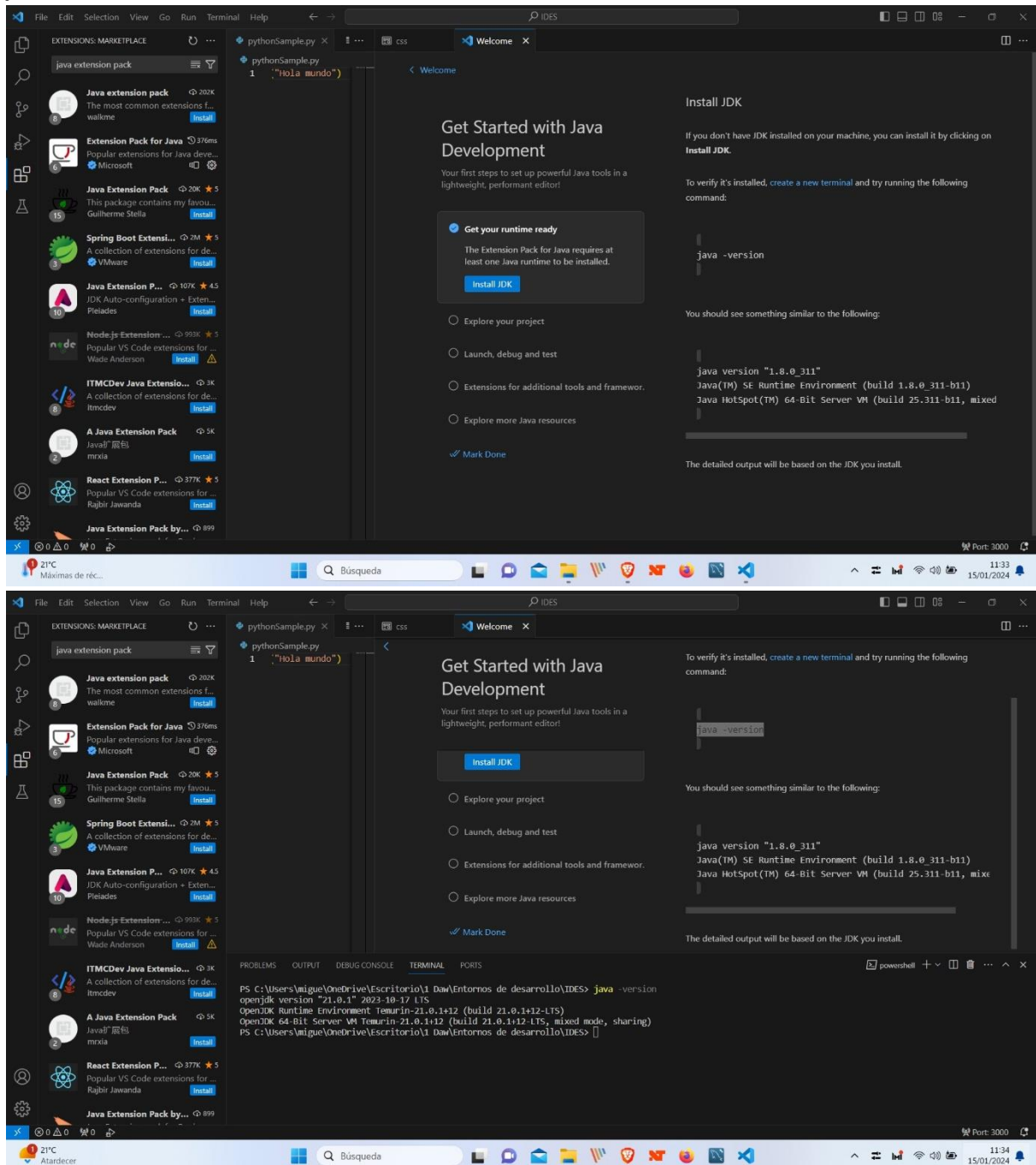
Instalación de paquetes

Para instalar los paquetes debemos ir a la pestaña Extensiones y buscar ahí los paquetes que deseamos descargar, en este caso fue el paquete de “Java Extensión Pack” teniendo cuidado porque hay muchos desarrolladores que buscan las descargas de sus paquetes y puede ser que no nos interese, en este caso descargamos el de Microsoft.



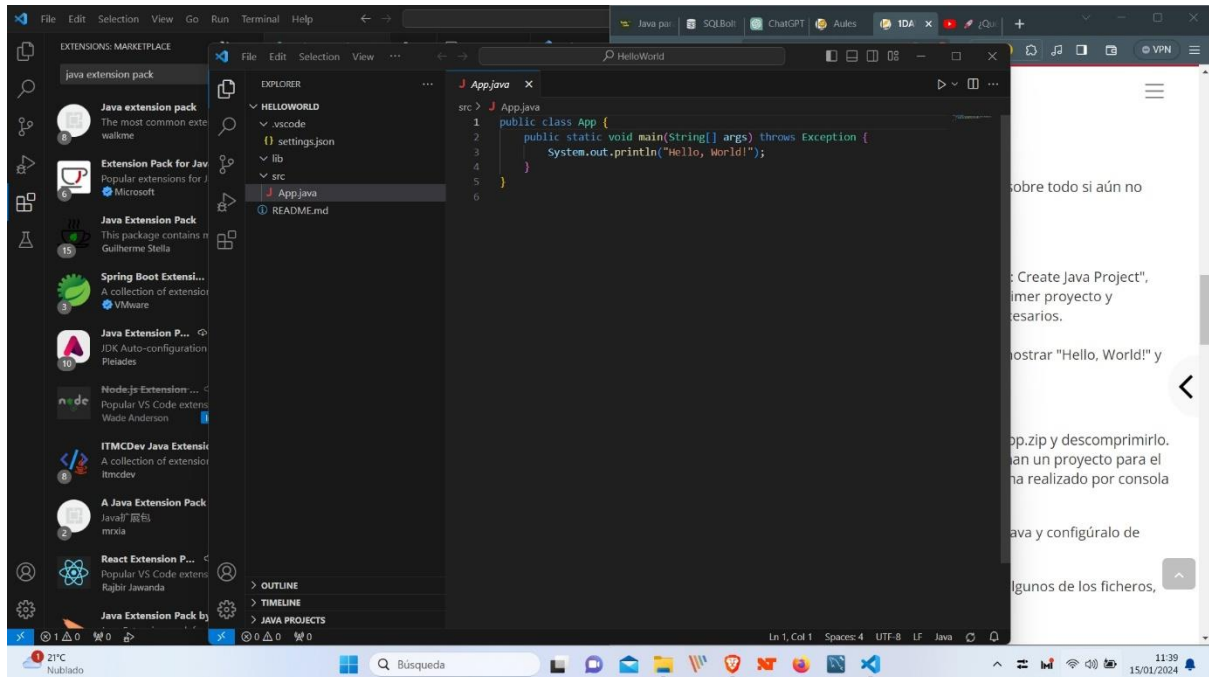
Get Started

Cuando iniciamos el VS Code tenemos una opción de “Get Started with java Development” aquí nos da el paso a paso para configurar el paquete de java pero también nos da el descargable del paquete de JDK necesario para la ejecución de java en el ordenador.



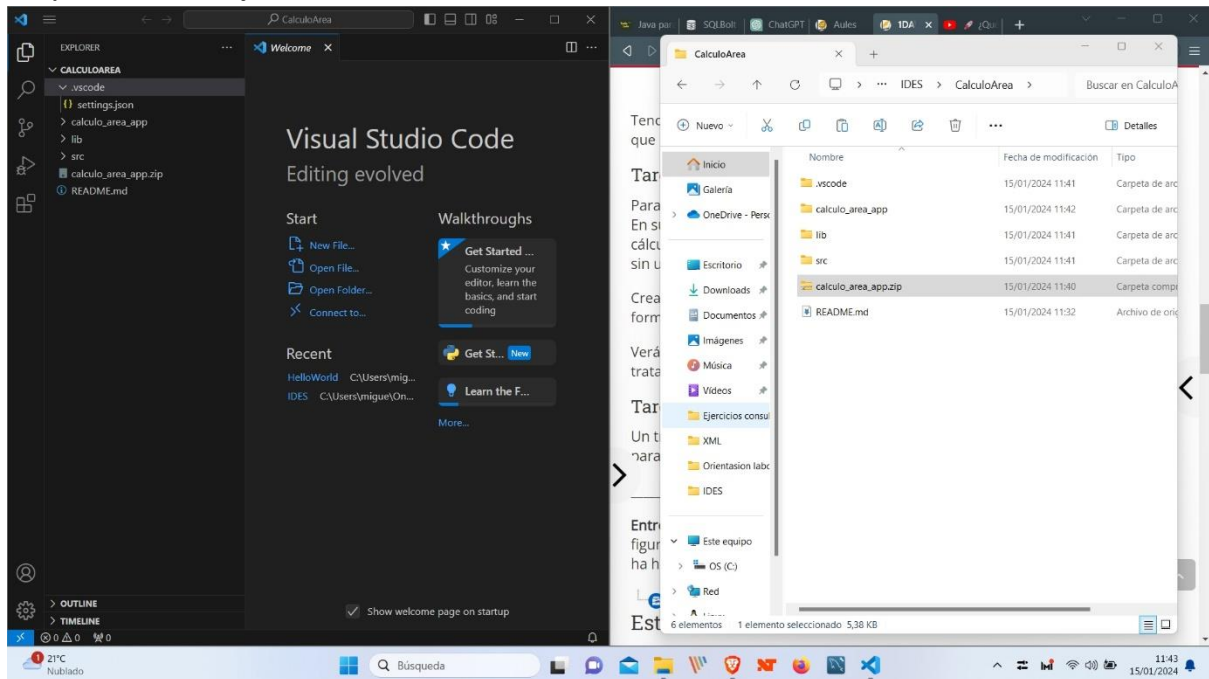
Nuestro Primer Proyecto

En la pestaña VIEW en la parte superior derecha encontramos Command Palette y buscamos java:Create Java Project” escogemos la carpeta donde queremos que este y creamos el proyecto, por defecto viene el hola mundo (lo normal en los lenguajes de programación) también existe un comando por teclado que explica en el video y es el Ctrl+shif+P.

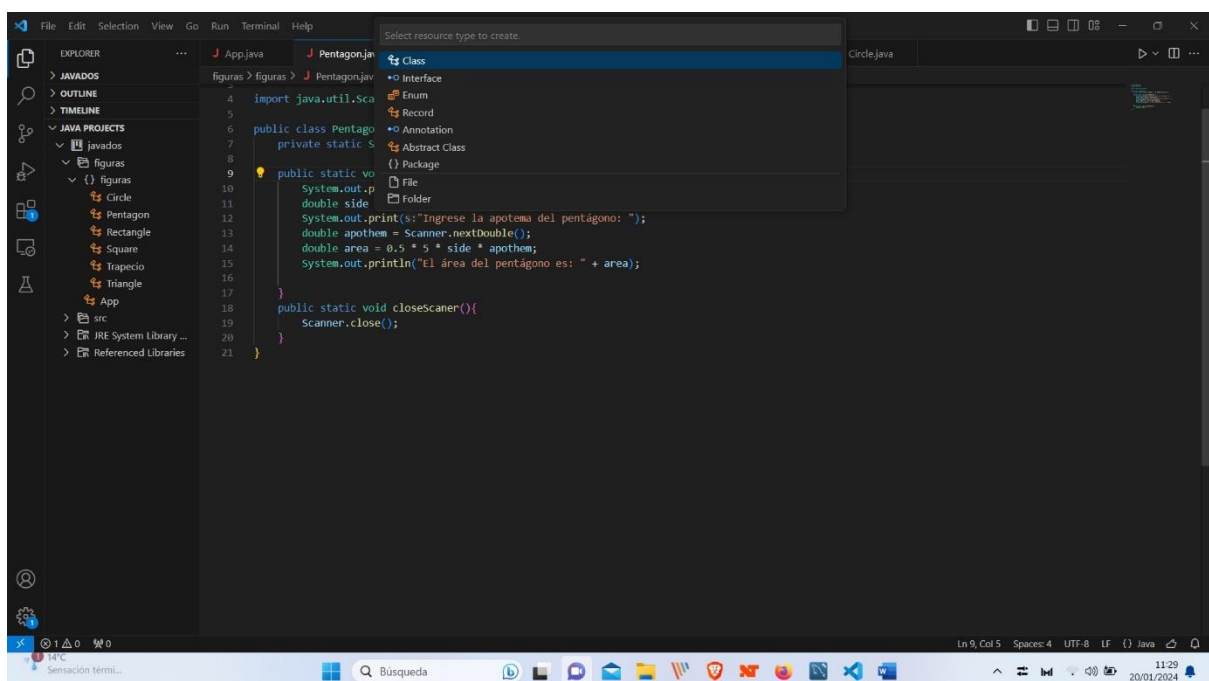


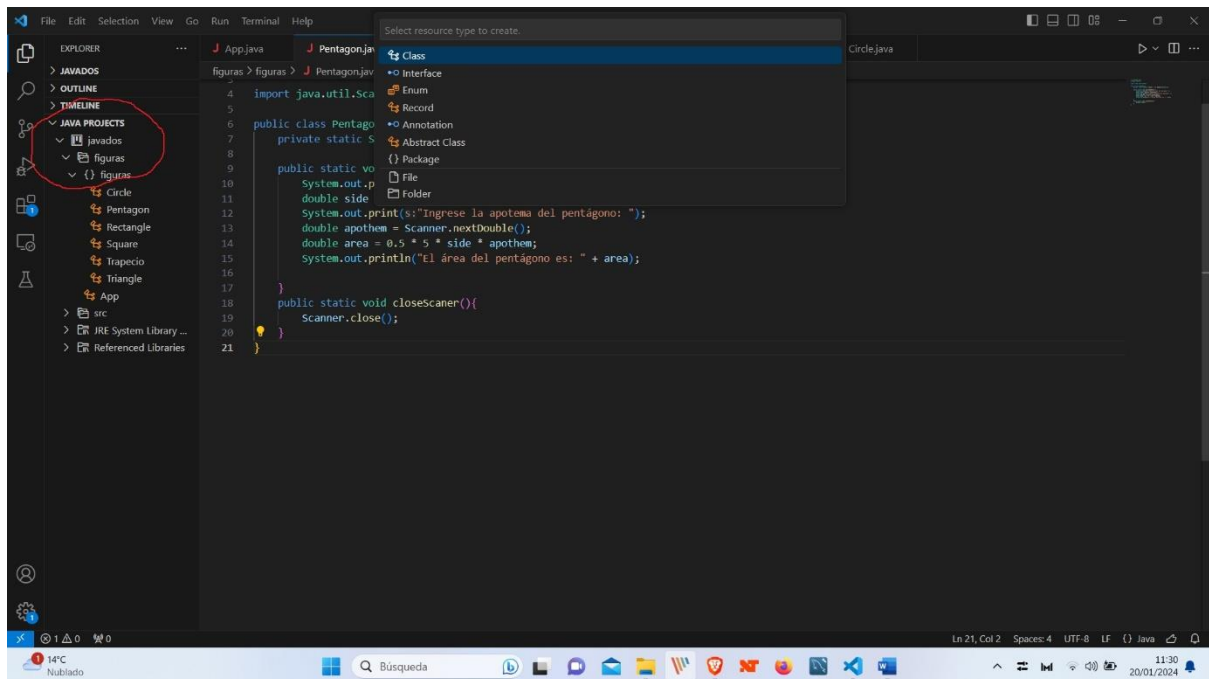
Importando un Proyecto

Empezamos esta tarea con la descarga del fichero en Aules y posteriormente empezamos la ejecución en VS Code.

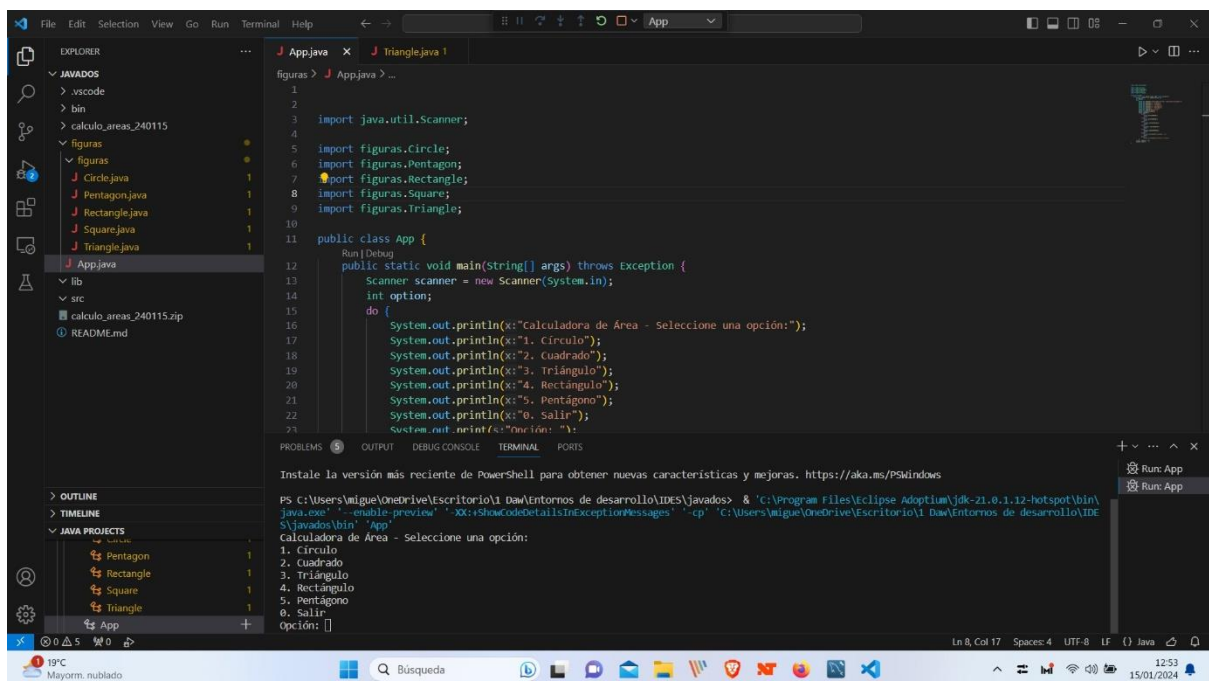


Empezamos a ejecutarlo pero primero tuvimos que crear un package de nombre figuras para meter todo el proyecto java en el y que funcionara para que el método main llamado App pudiera llamar las clases y hacer cada una de las operaciones.

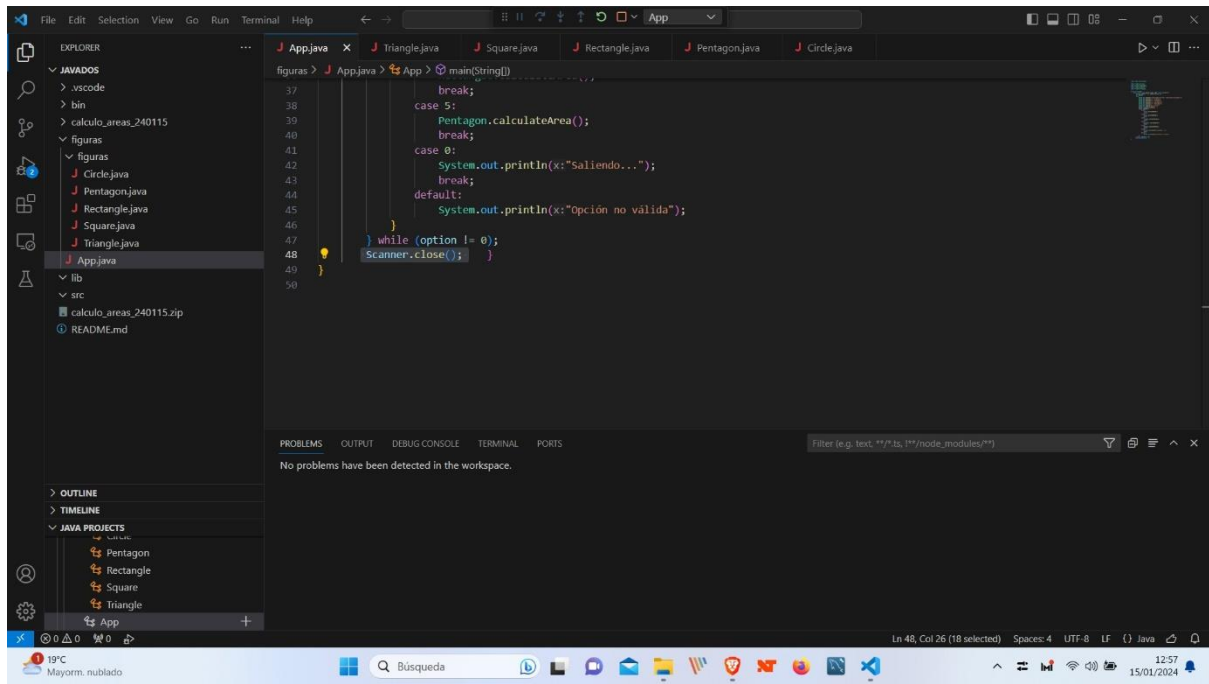




Luego que el programa ya encontró todas las clases y métodos que se llamaban al ejecutar el main(App) pudimos ejecutar el programa y que funcionase



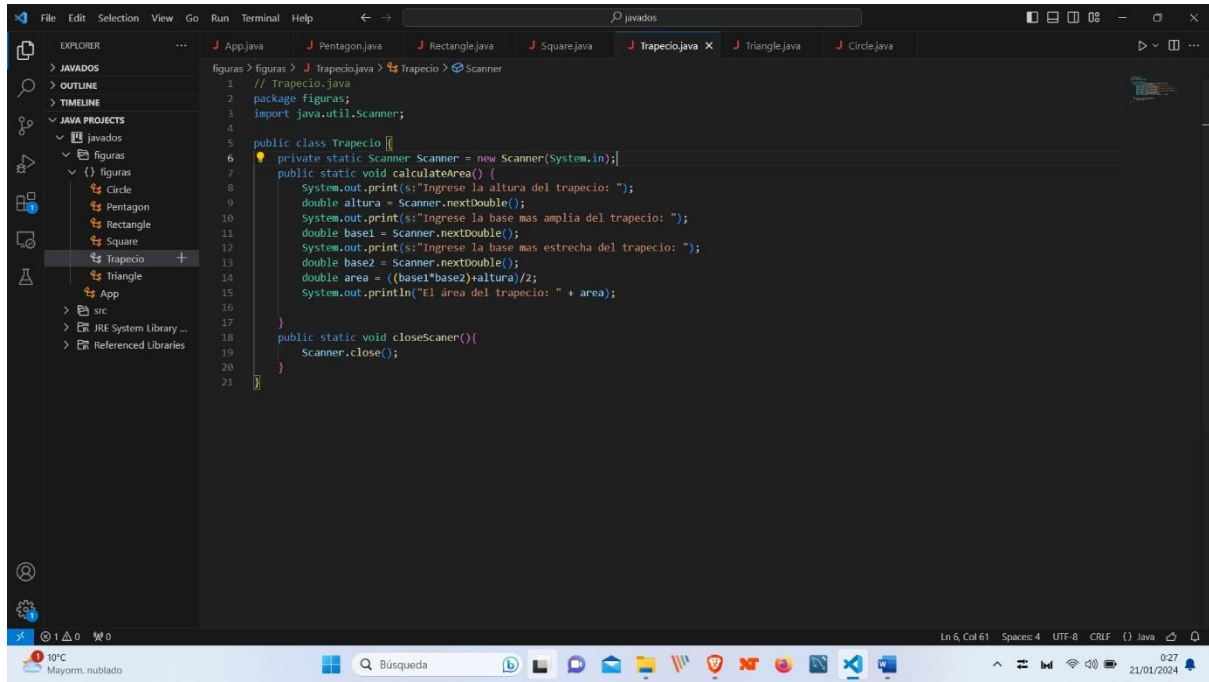
También tuvimos que agregar un scanner close al final de cada metodo para que ya no diera el aviso en problemas, aunque esto no es 100% necesario para que funciones el programa, es mejor cerrarlo ya que dejarlo abierto solo consume memoria y recursos del pc ya que se sigue ejecutando por que el programa no sabe si hay que cerrar el metodo scanner.



Aquí se muestra que ya no hay errores y que el programa se ejecuto sin problemas.

Trapecio

En esta tarea teníamos que agregar una clase nueva con su método llamada Trapecio.



```
1 // Trapecio.java
2 package figuras;
3 import java.util.Scanner;
4
5 public class Trapecio {
6     private static Scanner Scanner = new Scanner(System.in);
7     public static void calculateArea() {
8         System.out.print(s:"Ingrese la altura del trapecio: ");
9         double altura = Scanner.nextDouble();
10        System.out.print(s:"Ingrese la base mas amplia del trapecio: ");
11        double base1 = Scanner.nextDouble();
12        System.out.print(s:"Ingrese la base mas estrecha del trapecio: ");
13        double base2 = Scanner.nextDouble();
14        double area = ((base1+base2)*altura)/2;
15        System.out.println("El área del trapecio: " + area);
16    }
17    public static void closeScanner(){
18        Scanner.close();
19    }
20 }
21 }
```

al escribir el código tuvimos un inconveniente y es que el programa cuando se llamaba el método no mostraba por pantalla el resultado. Finalizaba el programa y lo cerraba, al final con la ayuda de los compañeros se encontró la solución y era dividir el scanner del método calculoArea().

```

1 // Trapecio.java
2 package figuras;
3 import java.util.Scanner;
4
5 public class Trapecio {
6     private static Scanner Scanner = new Scanner(System.in);
7     public static void calculateArea() {
8         System.out.print(s:"Ingrese la altura del trapecio: ");
9         double altura = Scanner.nextDouble();
10        System.out.print(s:"Ingrese la base mas amplia del trapecio: ");
11        double base1 = Scanner.nextDouble();
12        System.out.print(s:"Ingrese la base mas estrecha del trapecio: ");
13        double base2 = Scanner.nextDouble();
14        double area = ((base1+base2)*altura)/2;
15        System.out.println("El área del trapecio: " + area);
16    }
17 }
18 public static void closeScanner(){
19     Scanner.close();
20 }
21 }

```

Con esto, el programa en el método main ya funciona, en mi caso lo hice private y en el main tuvimos que poner todos los closeScanner como métodos propios de cada clase para que el main no cerrara el programa cuando hacia el calculo de cualquier clase, si no que siguiera en bucle hasta que el usuario lo quisiera apagar así:

```

47 case 0:
48     System.out.println(x:"Saliendo...");
49     break;
50 default:
51     System.out.println(x:"Opción no válida");
52 }
53 while (option != 0);
54 Circle.closeScanner();
55 Pentagon.closeScanner();
56 Rectangle.closeScanner();
57 Square.closeScanner();
58 Trapecio.closeScanner();
59 Triangle.closeScanner();
60 Scanner.close();
61 }
62 }
63 }
64 }
65 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Ingrese la altura del trapecio: 12
Ingrese la base mas amplia del trapecio: 5
Ingrese la base mas estrecha del trapecio: 5
El área del trapecio: 18.5
Calculadora de Área - Seleccione una opción:
1. Circulo
2. Cuadrado
3. Triángulo
4. Rectángulo
5. Pentágono
6. Trapecio
0. Salir
Opción: 0

```

Y así quedaría ya corregido y funcionando de forma adecuada.