

Estructura General Programa Java

1. Comentarios de Inicio
2. Sentencia de paquete
3. Sentencia de importación
4. Declaraciones de clases:
 - Declaración de variables
 - Declaración de métodos

```
/*
 * modulo1
 */

// Sentencia de Paquete
Package modulos;

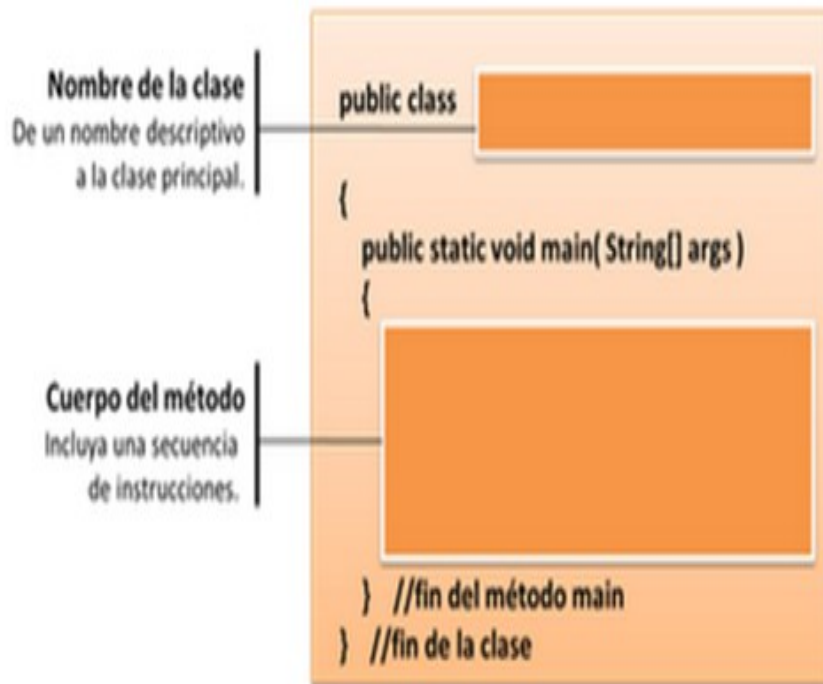
// Sentencia de Importación
import java.util.Scanner;

// Declaración de clase
public class Modulo1 {

    // Metodo principal
    public static void main(String[] args) {
        declararVariables();
    }

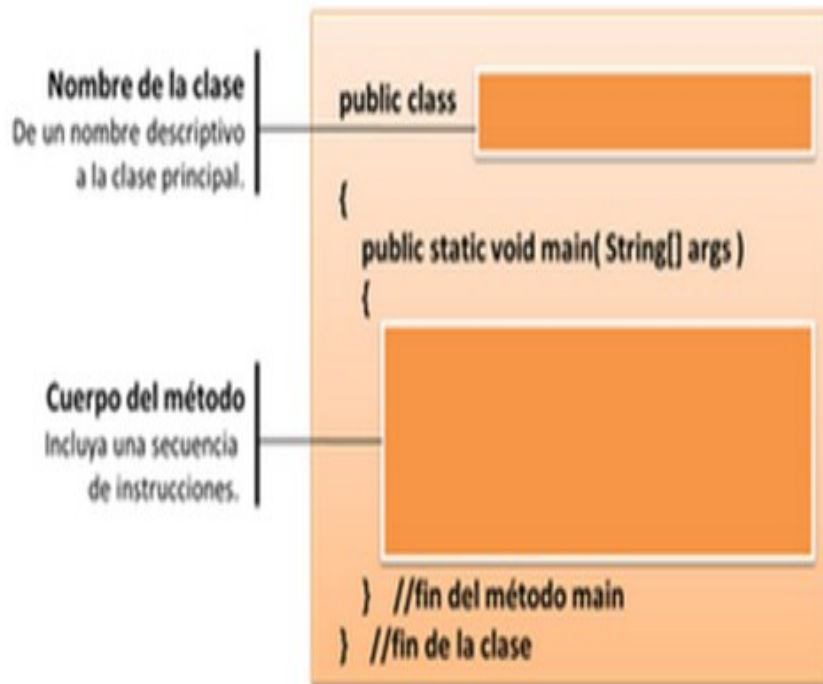
    // Otros métodos
    public static void declararVariables() {
        int    unEntero    = 30;
        System.out.printf("%d",unEntero);
    }
}
```

Programa básico en java



```
/**  
 * Hola  
 */  
public class Hola {  
  
    public static void main(String[] args) {  
        System.out.println("Hola Mundo");  
        System.out.println(args.length);  
        System.out.println(args[0]);  
  
    }  
  
}
```

Programa básico en java



```
/**
 * Hola
 */
public class Hola {

    public static void main(String[] args) {
        System.out.println("Hola Mundo");
        System.out.println(args.length);
        System.out.println(args[0]);

    }

}
```

Funciones

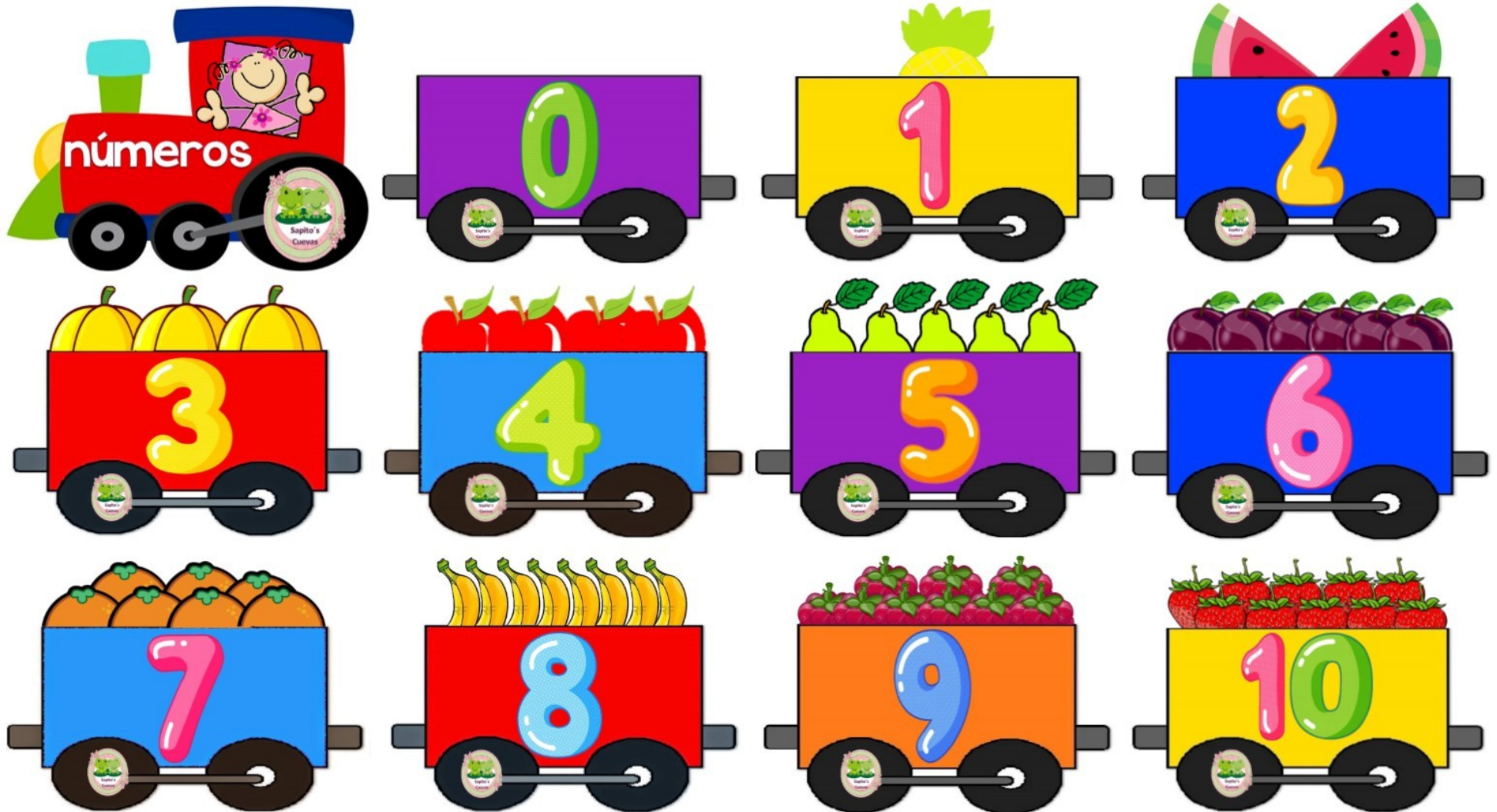
```
def validarTriangulo(ladoUno:int,ladoDos:int,ladoTres:int) -> bool:
    """
    Parámetros: Recibe los lados enteros de un triángulo
    Retorna : True: indicando que es posible construir el triángulo
            False: indicando que no es posible construir el triángulo
    """

    if (ladoUno + ladoDos > ladoTres) and (ladoUno + ladoTres > ladoDos) and (ladoDos + ladoTres > ladoUno):
        esPosibleTriangulo = True
    else:
        esPosibleTriangulo = False

    return esPosibleTriangulo
```

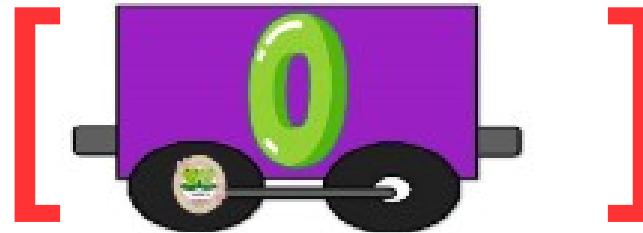
```
public static double calcularMenor(double valora, double valorb) {
    double menor;
    menor = valora < valorb ? valora : valorb;
    return menor;
}
```

Arreglos





5



```
Numeros = []
```

```
numeros[0]
```

```
public class Hola {  
    public static void main(String[] args)  
    {  
        System.out.println("Hola Mundo");  
        System.out.println(args.length);  
        System.out.println(args[0]);  
    }  
}
```

Estructuras de Decisión

if / if-else

```
if (ladoUno != ladoDos) and (ladoDos != ladoTres) and (ladoUno != ladoTres):  
    print ("El triángulo es escaleno")
```

```
if (entrada.equals("pepe")){  
    System.out.println("Hola: " + entrada);  
}
```

La evaluación de la condición debe devolver un valor booleano.

Estructuras de Decisión

if / if-else

```
if (ladoUno != ladoDos) and (ladoDos != ladoTres) and (ladoUno != ladoTres):  
    print ("El triángulo es escaleno")  
else:  
    print ("El triángulo no es escaleno")
```

```
if (entrada.equals("pepe")) {  
    System.out.println("Hola eres : " + entrada);  
}  
else{  
    System.out.println("No eres pepe eres: " + entrada);  
}
```

La evaluación de la condición debe devolver un valor booleano.

Clase Galleta



Propiedades de la Galleta

* forma = Corazón

Actividades de la Galleta

* Desboronarse()

Clase Pais



PAIS

Propiedades de Pais

- * nombre = Colombia
- * lengua = Español
- * poblacion = 50

Operaciones de Pais

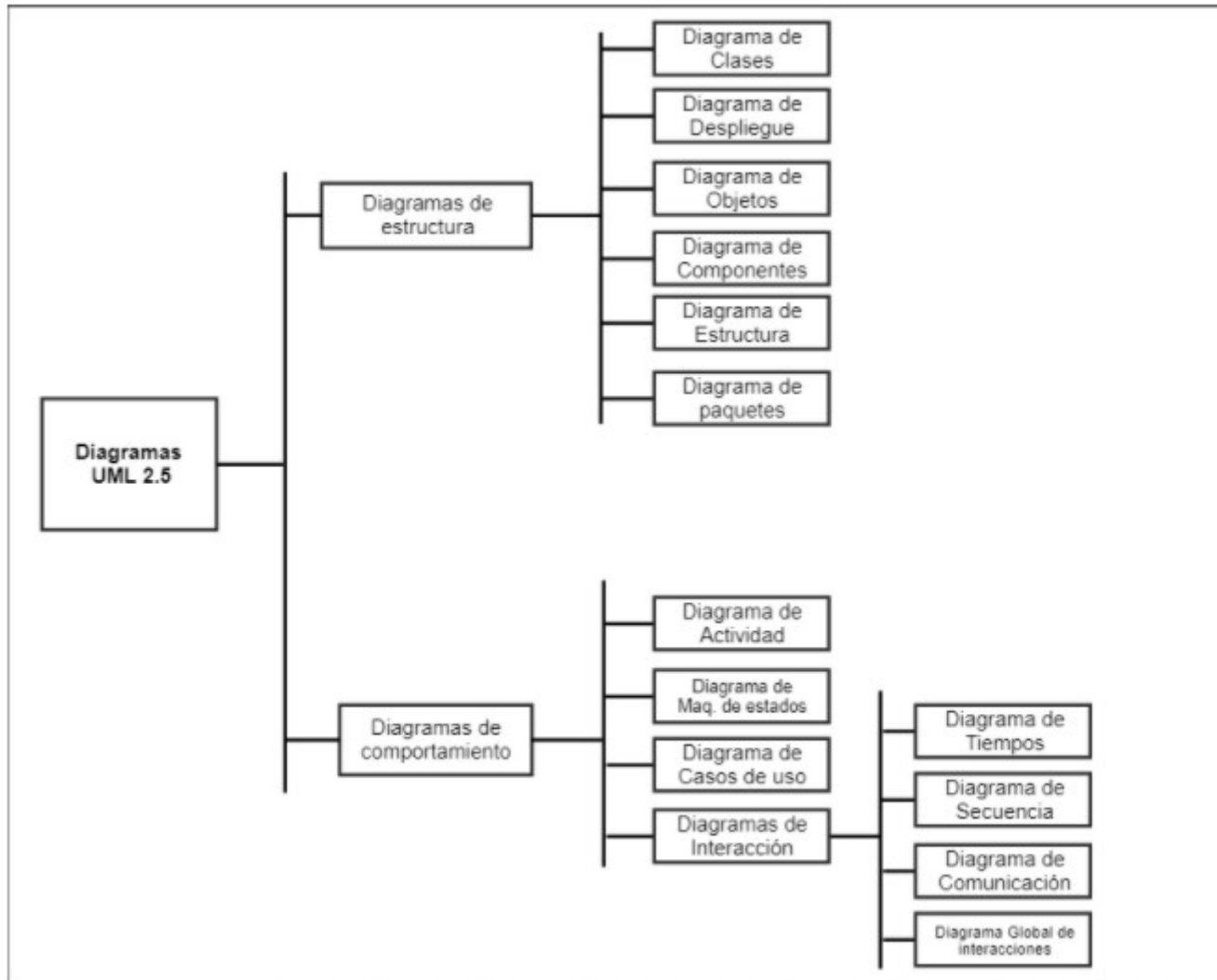
- * censaPoblacion();

Clase Pais

```
public class Pais{  
    private String nombre;  
    private String lengua;  
    private double poblacion;  
  
    public Pais(String nombre, String lengua, double poblacion) {  
        this.nombre = nombre;  
        this.lengua = lengua;  
        this.poblacion = poblacion;  
    }  
  
    public String getNombre() {  
        return this.nombre;  
    }  
  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public String getLengua() {  
        return this.lengua;  
    }  
  
    public void setLengua(String lengua) {  
        this.lengua = lengua;  
    }  
  
    public double getPoblacion() {  
        return this.poblacion;  
    }  
  
    public void setPoblacion(double poblacion) {  
        this.poblacion = poblacion;  
    }  
  
    public void censarPoblacion (String nombre){  
        System.out.printf("Vamos a censar la población de %s",nombre);  
    }  
}
```

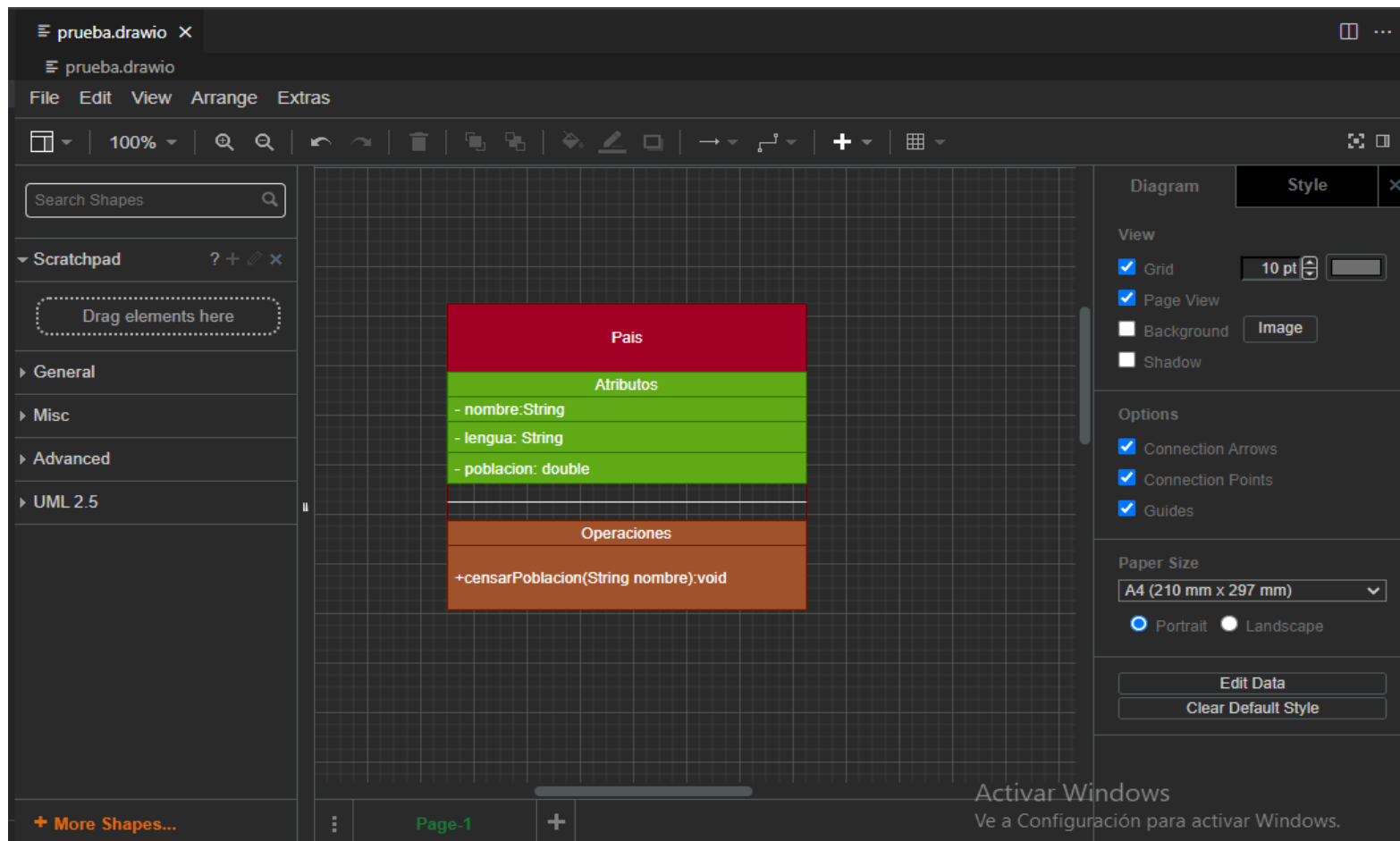


UML (OMG)



Clasificación de los diagramas UML

UML



Draw.io Integration v1.6.1

Henning Dieterichs | 393.878 | ★★★★★ (70)

This unofficial extension integrates Draw.io into VS Code.

Disable Uninstall ⚙️

This extension is enabled globally.

Clase Pais



```
public class Pais{  
    private String nombre;  
    private String lengua;  
    private double poblacion;  
  
    public Pais(String nombre, String lengua, double poblacion){  
        this.nombre = nombre;  
        this.lengua = lengua;  
        this.poblacion = poblacion;  
    }  
  
    public String getNombre() {  
        return this.nombre;  
    }  
  
    public void setNombre(String nombre) {  
        this.nombre = nombre;  
    }  
  
    public String getLengua() {  
        return this.lengua;  
    }  
  
    public void setLengua(String lengua) {  
        this.lengua = lengua;  
    }  
  
    public double getPoblacion() {  
        return this.poblacion;  
    }  
  
    public void setPoblacion(double poblacion) {  
        this.poblacion = poblacion;  
    }  
  
    public void censarPoblacion (String nombre){  
        System.out.printf("Vamos a censar la población de %s", nombre);  
    }  
}
```



Generar código automáticamente

```
public class Pais{
    private String nombre;
    private String lengua;
    private double poblacion;

    public Pais(String nombre, String lengua, double poblacion) {
        this.nombre = nombre;
        this.lengua = lengua;
        this.poblacion = poblacion;
    }

    public String getNombre() {
        return this.nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

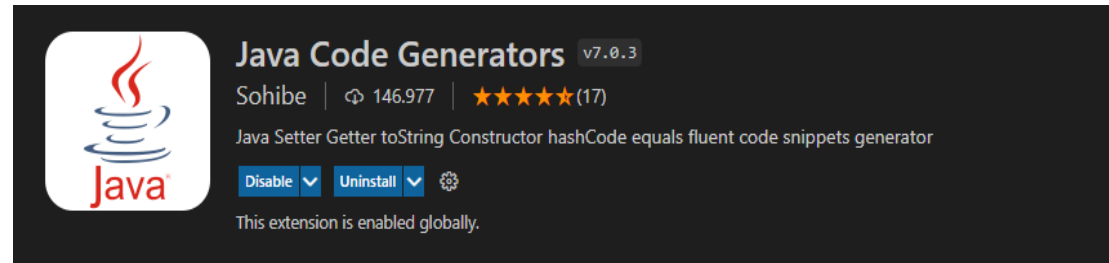
    public String getLengua() {
        return this.lengua;
    }

    public void setLengua(String lengua) {
        this.lengua = lengua;
    }

    public double getPoblacion() {
        return this.poblacion;
    }

    public void setPoblacion(double poblacion) {
        this.poblacion = poblacion;
    }

    public void censarPoblacion (String nombre){
        System.out.printf("Vamos a censar la población de %s",nombre);
    }
}
```



Tipos de Modificadores

- **Acceso**
- **No acceso**

Acceder a una clase

- Una clase A accede a otra clase B cuando:

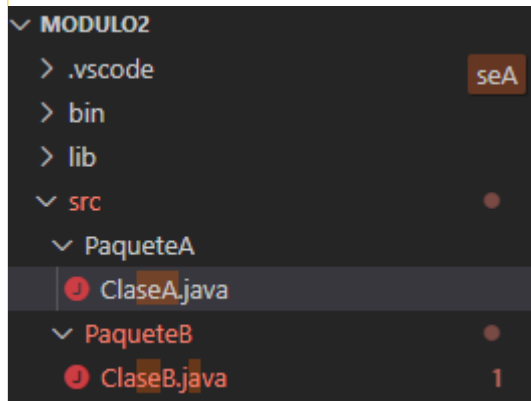
Crea una instancia de la clase B.

Extiende de una clase B.

Accede a metodos y variables.

Modificador de Acceso Default clase

- Una clase con acceso default solo puede ser vista por clases dentro del mismo paquete.



```
src > PaqueteA > ClaseA.java > ClaseA
1  package PaqueteA;
2
3  /**
4   * ClaseA
5   */
6  class ClaseA {
7
8
9  }
```

```
src > PaqueteB > ClaseC.java > ClaseC
1  package PaqueteB;
2
3  class ClaseC {
4
5
6  }
```

VS Code editor showing an error in ClaseB.java. The error message is "The type ClaseA is not visible java(167777219)". The code in ClaseB.java is:

```
src > PaqueteB > ClaseB.java > ...
1  package PaqueteB;
2
3  import PaqueteA.*;
4
5  public class ClaseB {
6
7      public static void main(String[] args) {
8
9          ClaseA objetoA;
10
11      }
12
13  }
```

```
PaqueteB > ClaseB.java > ClaseB > main(String[])
package PaqueteB;

import PaqueteA.*;

public class ClaseB {

    Run | Debug
    public static void main(String[] args) {

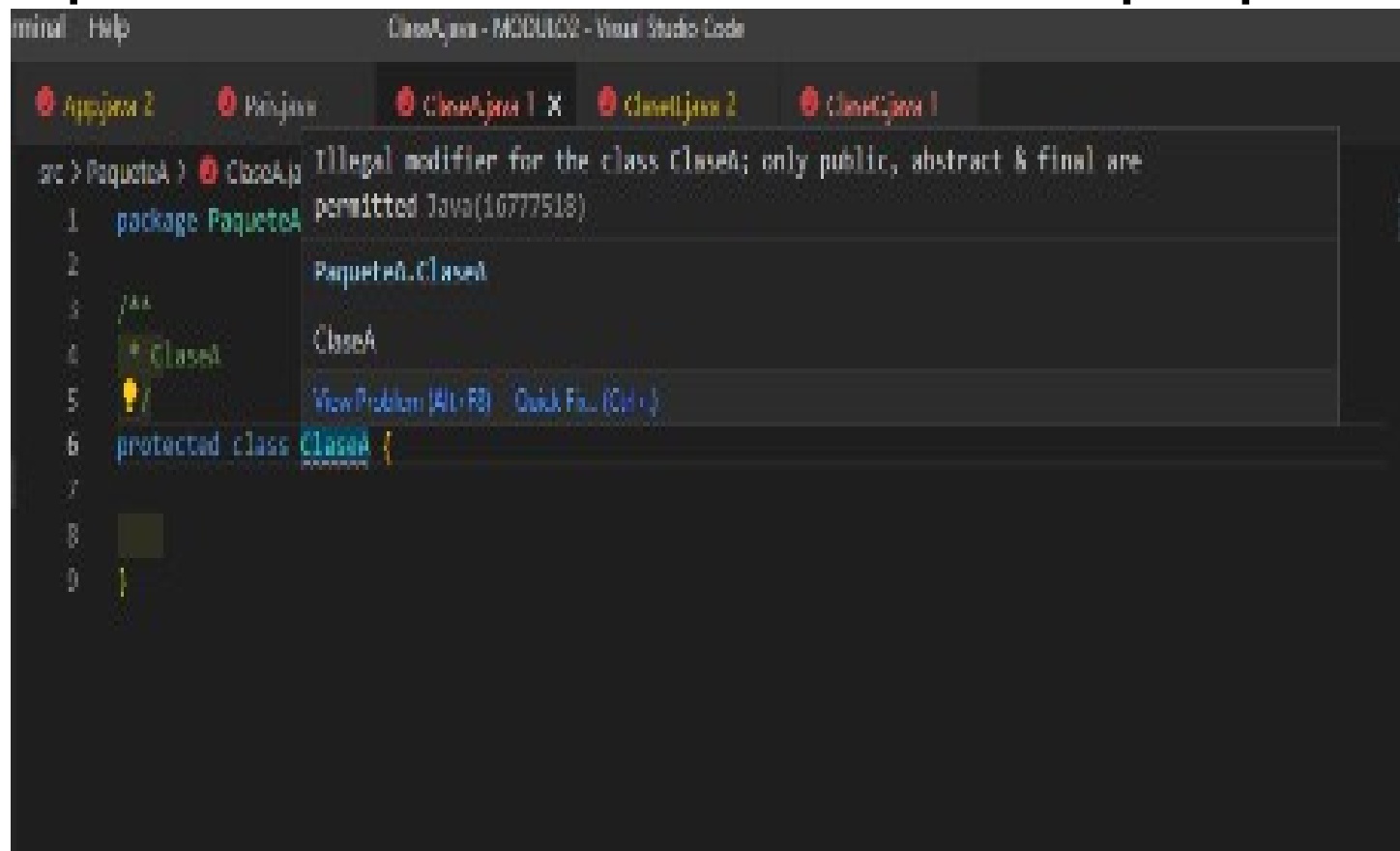
        ClaseA objetoA;
        ClaseC objectC;

        objectC = new ClaseC();

    }
```

Modificador de Acceso protected clase

- Una clase con acceso protected solo puede ser vista por clases dentro del mismo paquete.

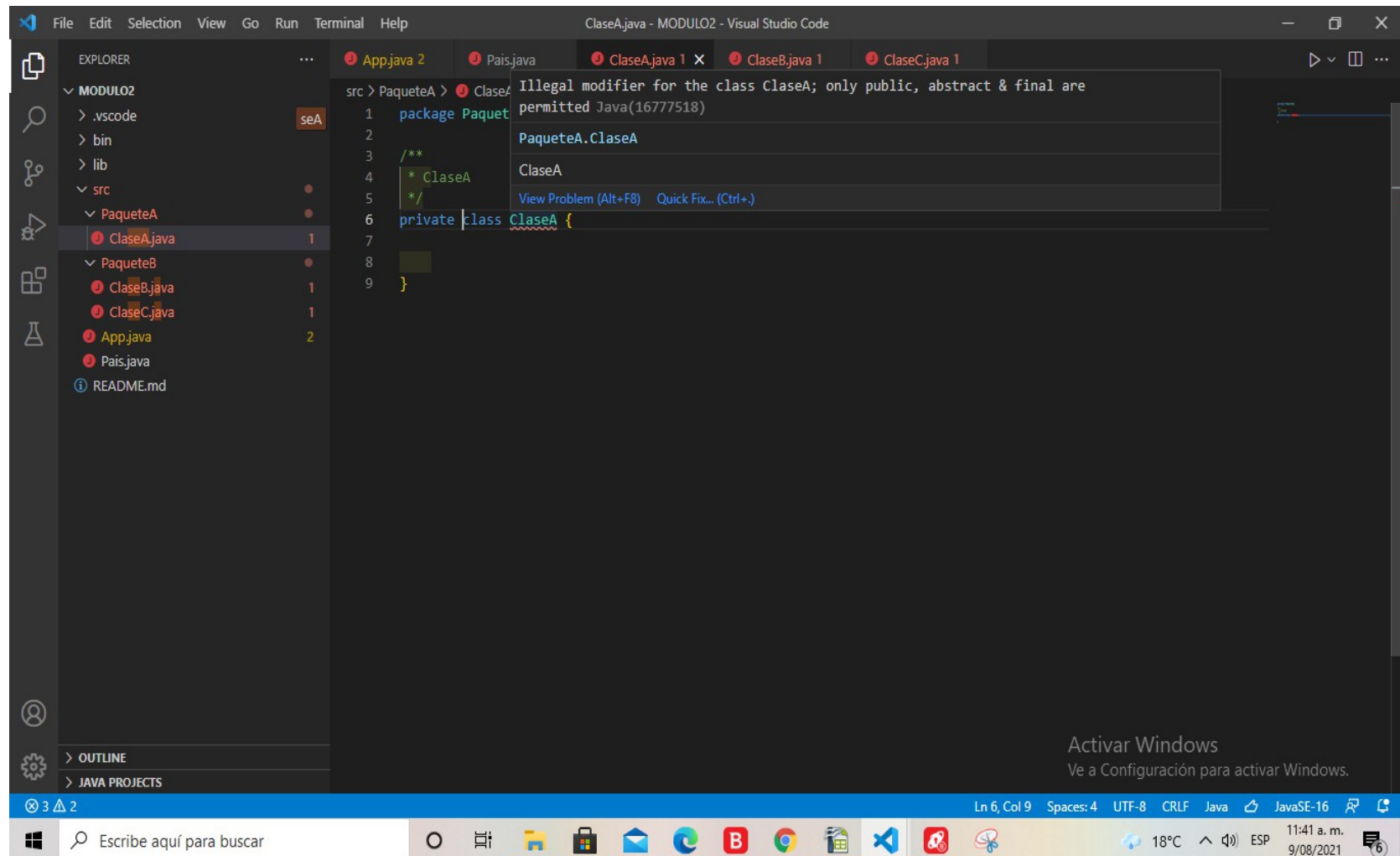


The screenshot shows the Visual Studio Code editor with a Java file named `ClaseA.java`. The code in the editor is as follows:

```
src > PaqueteA > ClaseA.java
1 package PaqueteA;
2
3 /**
4  * ClaseA
5  */
6 protected class ClaseA {
7
8
9 }
```

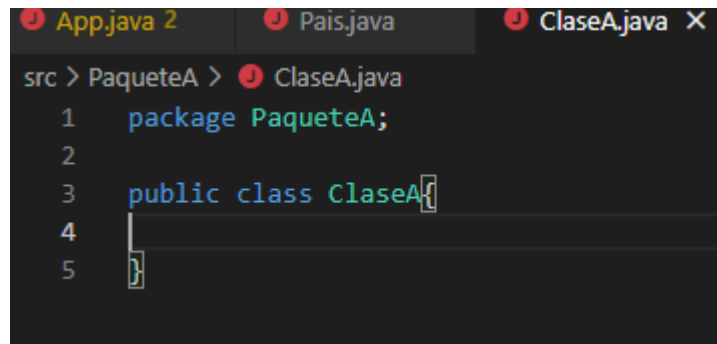
A red squiggly line is under the `protected` keyword on line 6. A tooltip is displayed over this line, showing the error message: "Illegal modifier for the class ClaseA; only public, abstract & final are permitted Java(16777518)". The tooltip also shows the file path `PaqueteA.ClaseA` and provides actions like "View Problem (Alt+F8)" and "Quick Fix... (Ctrl+.)". The top of the editor shows several tabs: `App.java`, `PaqueteA.java`, `ClaseA.java` (which is active and has a red 'X' icon), `ClaseB.java`, and `ClaseC.java`.

Modificador de Acceso private clase

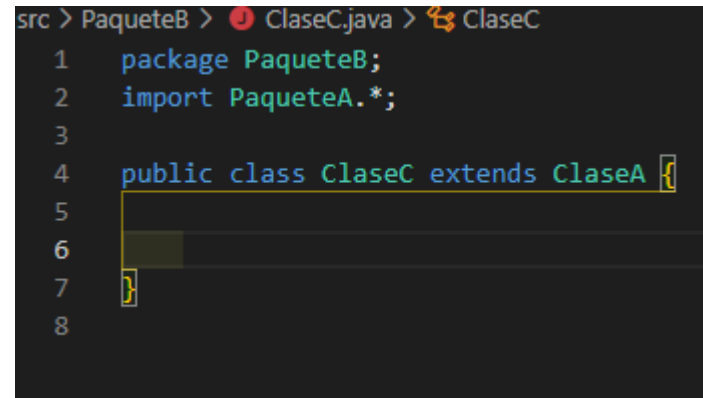


Modificadores de no acceso clase

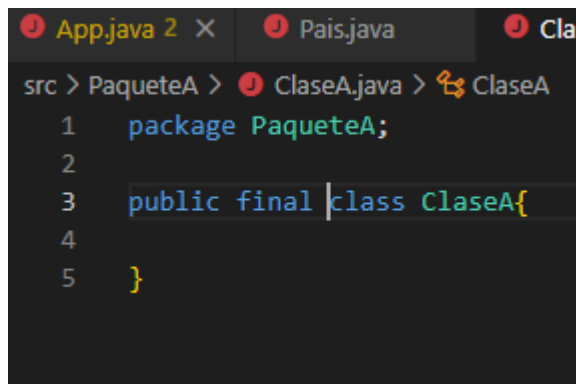
- Final : La clase no puede ser extendida.
- Abstract : La clase no puede ser instanciada.



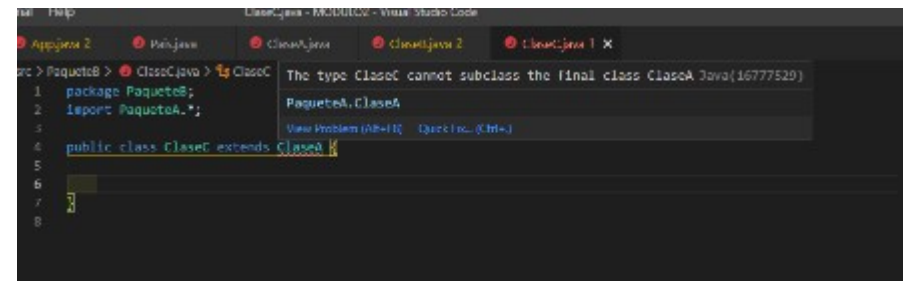
```
src > PaqueteA > ClaseA.java
1 package PaqueteA;
2
3 public class ClaseA{
4
5 }
```



```
src > PaqueteB > ClaseC.java > ClaseC
1 package PaqueteB;
2 import PaqueteA.*;
3
4 public class ClaseC extends ClaseA {
5
6
7
8 }
```



```
src > PaqueteA > ClaseA.java > ClaseA
1 package PaqueteA;
2
3 public final class ClaseA{
4
5 }
```



```
src > PaqueteB > ClaseC.java > ClaseC
1 package PaqueteB;
2 import PaqueteA.*;
3
4 public class ClaseC extends ClaseA {
5
6
7
8 }
```

The type ClaseC cannot subclass the final class ClaseA Java(16777529)

Modificadores de acceso para métodos y variables

<https://javadesdecero.es/poo/modificadores-de-acceso/>

Resumen Modificadores de Acceso

Modifiers	Members
public	Accessible everywhere.
protected	Accessible by any class in the same package as its class, and accessible only by subclasses of its class in other packages.
default (no modifier)	Only accessible by classes, including subclasses, in the same package as its class(package accessibility).
private	Only accessible in its own class and not anywhere else.

More restrictive ↓

Access Modifier	Its own class	Class in Same Package	Subclass in Same Package	Subclass in Different Package	Class in Different Package
public	Yes	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	Yes	No
default	Yes	Yes	Yes	No	No
private	Yes	No	No	No	No

Modificadores de No acceso para métodos y variables

- METODO

- Final: El método no puede sobrescrito en una subclase.
- Abstract: El método no tiene implementación.
- Static: se llama a través de su nombre de clase, sin que se cree ningún objeto de esa clase.

- VARIABLES

- Final: La variable no puede cambiar su valor una vez inicializada. (Para crear constantes).
- Static: son, esencialmente, variables globales.

Modificadores de No acceso para métodos y variables

Métodos y atributos estáticos



El futuro digital
es de todos

MinTIC

Palabra reservada **static**.

Define si un método o atributo es llamado desde la clase o desde una instancia de la clase.

Static

```
public class Animal{  
    ...  
    public static void comer(){}  
}
```



```
Animal.comer()
```

Sin Static

```
public class Animal{  
    ...  
    public void comer(){}  
}
```



```
Animal x = new Animal();  
x.comer()
```



UNIVERSIDAD
SERGIO ARBOLEDA

Misión
TIC2022