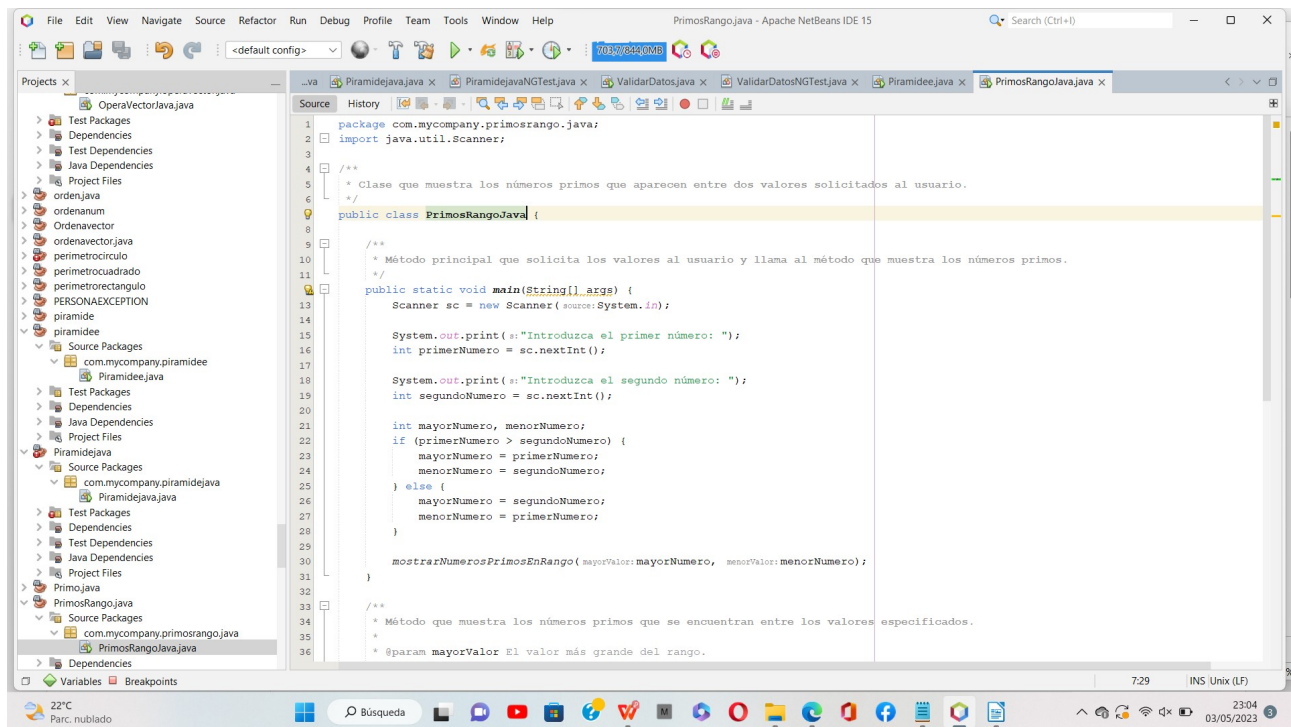


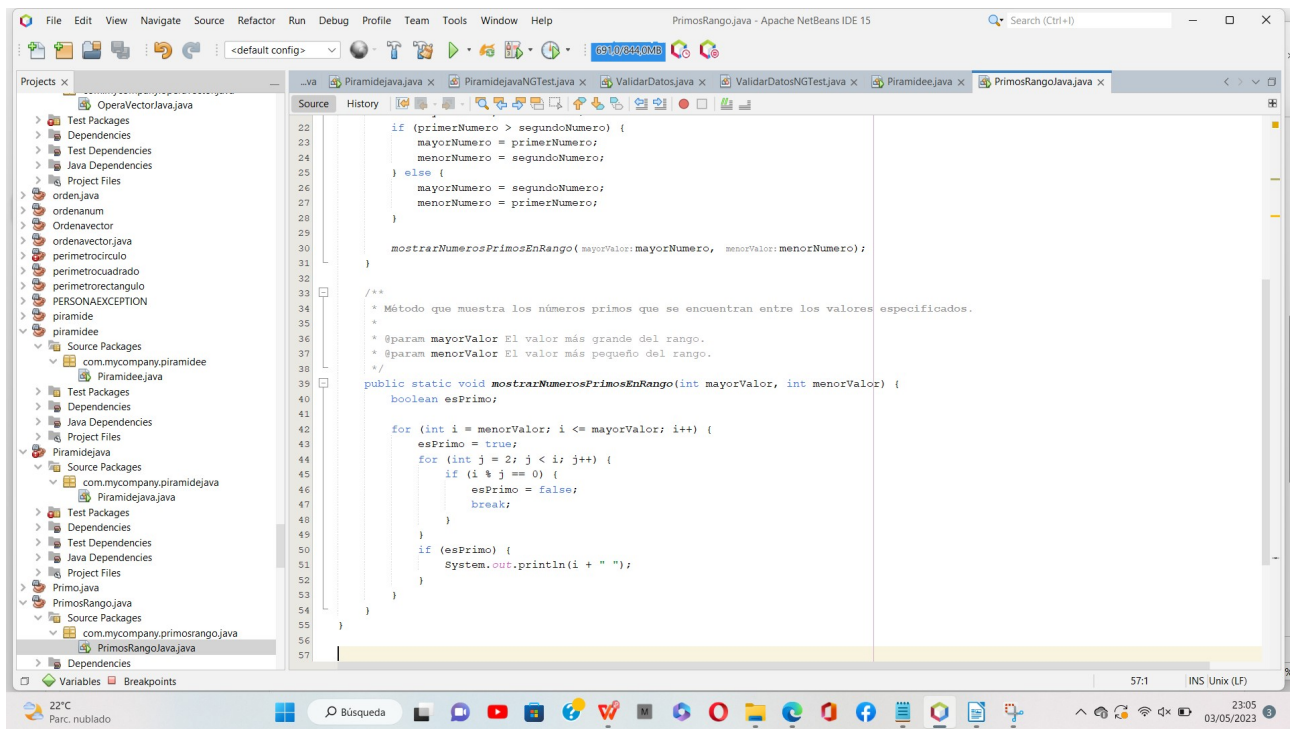
1. El siguiente código de Java, muestra los números primos que aparecen entre dos valores solicitados al usuario.

Vamos a Refactorizar este programa, utilizando cualquiera la opción de Refactorización de alguno de los IDE que hemos visto (Eclipse, IntelliJ IDEA, Visual Studio Code o NetBeans) y realizando lo siguiente: a) (0,25 puntos.) Formatea el código para que cada línea contenga una única instrucción y se ajuste debidamente el sangrado (tabulación). b) (0,75 puntos.) Renombrar las variables para que hagan alusión a la función que realizan en el programa. c) (1 punto.) Extrae los métodos posibles y deja el método main con lo mínimo posible (llamadas a los diferente métodos creados). d) (1 punto.) Genera los comentarios de JavaDoc, de la clase y de cada uno de los métodos.

El código refactorizado quedaría así:



```
1 package com.mycompany.primosrango.java;
2 import java.util.Scanner;
3
4 /**
5  * Clase que muestra los números primos que aparecen entre dos valores solicitados al usuario.
6  */
7 public class PrimosRangoJava {
8
9     /**
10      * Método principal que solicita los valores al usuario y llama al método que muestra los números primos.
11      */
12     public static void main(String[] args) {
13         Scanner sc = new Scanner(System.in);
14
15         System.out.print("Introduzca el primer número: ");
16         int primerNumero = sc.nextInt();
17
18         System.out.print("Introduzca el segundo número: ");
19         int segundoNumero = sc.nextInt();
20
21         int mayorNumero, menorNumero;
22         if (primerNumero > segundoNumero) {
23             mayorNumero = primerNumero;
24             menorNumero = segundoNumero;
25         } else {
26             mayorNumero = segundoNumero;
27             menorNumero = primerNumero;
28         }
29
30         mostrarNumerosPrimosEnRango(mayorNumero, menorNumero);
31     }
32
33     /**
34      * Método que muestra los números primos que se encuentran entre los valores especificados.
35      *
36      * @param mayorValor El valor más grande del rango.
```

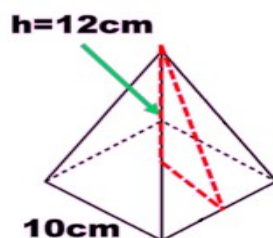


Explicación de los cambios realizados:

- Se ha formateado el código para que cada línea contenga una única instrucción y se ajuste el sangrado (tabulación) de forma adecuada, para mejorar la legibilidad.
- Se han renombrado las variables para que hagan alusión a la función que realizan en el programa. Por ejemplo, se ha renombrado la variable "t" por "sc" (de scanner) y se han renombrado "a" y "b" por "primerNumero" y "segundoNumero", respectivamente.
- Se han extraído los métodos posibles para mejorar la legibilidad del código y la facilidad de mantenimiento. En este caso se ha extraído un método "mostrarNumerosPrimosEnRango" que realiza la lógica principal del programa y se llama desde el método principal "main".
- Se han generado los comentarios de JavaDoc, tanto para la clase como para cada uno de los métodos, para documentar adecuadamente el código y hacerlo más fácil de entender para otros desarrolladores.

2. Desarrolla una aplicación de consola en Java que sea modular mediante el uso de la clase Piramide. Deberá solicitar al usuario los parámetros necesarios para crear una pirámide de base cuadrada: lado de la base y altura, estos valores serán atributos de la clase.

$$\text{Volumen} = \frac{\text{área de la base} \times \text{altura}}{3}$$



a) Desarrolla el código de forma modular con los siguientes métodos : ◦ Constructor que reciba como parámetros todos los atributos para crear la pirámide. (0,25 puntos) ◦ Getters y setters para el acceso a todos los atributos. (0,25 puntos) ◦ double calcularAreaBase(...) de la pirámide. (1 punto) ◦ float calcularVolumen(...) de la pirámide. (1 punto) ◦ Solicitar los valores (lado de la base y la altura) al usuario. Utilizando un método. (0,75 puntos) ◦ Imprimir el volumen de la pirámide. (0.75 puntos) ◦ Crear dos pirámides y determinar si una pirámide cabe dentro de otra. (1,5 puntos) Probar su funcionamiento. b)) Genera la documentación de la clase con sus métodos empelando comentarios JavaDoc. (1,5 puntos)

```

1 package com.myccompany.piramide;
2 import java.util.Scanner;
3 /**
4  *
5  * Clase que representa una pirámide con un lado de base y una altura.
6  */
7 public class Piramide {
8     /**
9      *
10     Lado de la base de la pirámide.
11     */
12     private double ladoBase;
13     /**
14     Altura de la pirámide.
15     */
16     private double altura;
17     /**
18     Constructor de la clase Piramide.
19     @param ladoBase el lado de la base de la pirámide.
20     @param altura la altura de la pirámide.
21     */
22     public Piramide(double ladoBase, double altura) {
23         this.ladoBase = ladoBase;
24         this.altura = altura;
25     }
26     /**
27     Devuelve el lado de la base de la pirámide.
28     @return el lado de la base de la pirámide.
29     */
30     public double getLadoBase() {
31         return ladoBase;
32     }
33     /**
34     */
35     }
36     /**
37     */
38 }

```

