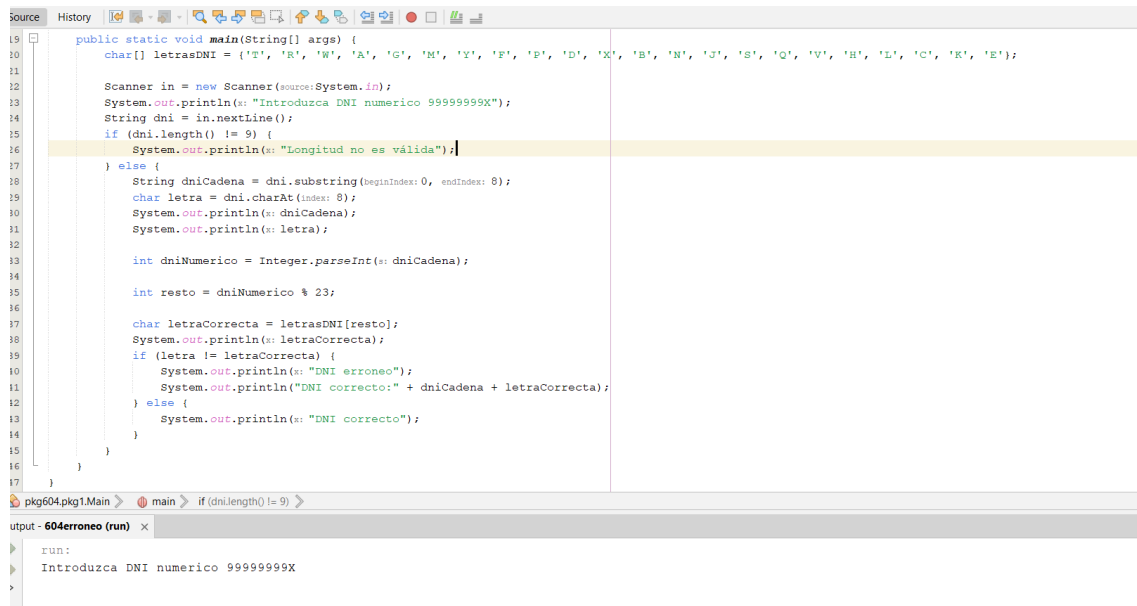


CLEAN CODE



```
19 public static void main(String[] args) {
20     char[] letrasDNI = {'T', 'R', 'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B', 'N', 'J', 'S', 'Q', 'V', 'H', 'L', 'C', 'K', 'E'};
21
22     Scanner in = new Scanner(System.in);
23     System.out.println("Introduzca DNI numerico 99999999X");
24     String dni = in.nextLine();
25     if (dni.length() != 9) {
26         System.out.println("Longitud no es válida");
27     } else {
28         String dniCadena = dni.substring(beginIndex: 0, endIndex: 8);
29         char letra = dni.charAt(index: 8);
30         System.out.println("dniCadena");
31         System.out.println("letra");
32
33         int dniNumerico = Integer.parseInt(dniCadena);
34
35         int resto = dniNumerico % 23;
36
37         char letraCorrecta = letrasDNI[resto];
38         System.out.println("letraCorrecta");
39         if (letra != letraCorrecta) {
40             System.out.println("DNI erroneo");
41             System.out.println("DNI correcto:" + dniCadena + letraCorrecta);
42         } else {
43             System.out.println("DNI correcto");
44         }
45     }
46 }
47 }
```

pkg604.pkg1.Main | main | if (dni.length() != 9)

output - 604erroneo (run)

run:

Introduzca DNI numerico 99999999X

Considero que es un código limpio porque cumple con las características que lo definen así.

En primer lugar, los nombres de las variables tienen significado y nombres fáciles que lo definen, por ejemplo “letrasDNI”, “dniCadena”, “letra”, “dniNumerico” o “letraCorrecta”.

Además, a la vez que lees los nombres de los atributos deduces cuál es su uso y no necesitas nada más que lo defina.

En segundo lugar, las funciones que observamos en el ejercicio:

- Declaración del array de letras del DNI (sirve para almacenar las letras del DNI en un orden determinado).
- Se pide al usuario la entrada de su DNI con unas condiciones determinadas. (8 dígitos seguidos de una letra).
- La siguiente función es la validación de la longitud del DNI.
- Separa los números y la letra del DNI ingresado. (función substring y charAt).
- Conversión de la parte numérica del DNI a un número entero. (Integer.parseInt(dniCadena)).
- Función utilizada para el cálculo del resto de dividir el número del DNI entre 23.
- Obtención de la letra correcta según el número de DNI ingresado, sirve para buscar la letra que corresponde al índice que se obtiene en el array letrasDNI.
- Comprobación y comparación de que la letra obtenida coincida con la letra ingresada. Si la letra obtenida es diferente a la ingresada, muestra un mensaje de "DNI erróneo" y proporciona la combinación correcta de números y letras. Si son iguales, indica que el DNI es correcto.

Como resumen, las funciones están bien definidas, son sencillas, con pocas líneas y con un solo uso sin ambigüedades.

Para finalizar, en el código podemos observar que no hay ningún comentario, considero que en ocasiones los comentarios son redundantes si repiten lo que ya está claro en el código además que en este caso particular no son necesarios porque los métodos, atributos, funciones... son auto explicativos.

Si el código está bien escrito y es fácil de entender sin comentarios, incluso puede ser más fácil de mantener.

El código sigue un orden lógico bien estructurado que ayuda a una mejor comprensión sobre la función tiene cada parte del código.

A pesar de esto, en otros contextos o en proyectos más grandes o complejos, los comentarios pueden ser esenciales para explicar el propósito de ciertas secciones de código, especialmente cuando la lógica es complicada o no es evidente a primera vista.