

Clean Code

Ejemplo I: Nombres

Antes:

```
DAM1HernandezPuertas_Francisco_NombresPrevio.java > DAM1HernandezPuertas_Francisco_NombresPrevio
1  import java.util.Scanner;
2  public class DAM1HernandezPuertas_Francisco_NombresPrevio {
3      public static void main(String[] args) {
4          Scanner scan = new Scanner(System.in);
5          String numeroString;
6          int numeroInt;
7          do{
8              System.out.print(s:"Introduzca un número para el lado del cuadrado: ");
9              numeroString = scan.nextLine();
10             numeroInt = Integer.valueOf(numeroString);
11         }while(numeroInt<=0);
12
13         for (int i = 0; i < numeroInt; i++) {
14             for (int j = 0; j < numeroInt; j++) {
15                 System.out.print(s:"#");
16                 if (j < numeroInt - 1) {
17                     System.out.print(s:" ");
18                 }
19             }
20             System.out.println();
21         }
22     }
23 }
```

Después:

```
DAM1HernandezPuertas_Francisco_NombresNuevo.java > ...
1  import java.util.Scanner;
2  public class DAM1HernandezPuertas_Francisco_NombresNuevo { /*Aquí cambiaríamos el nombre de la clase a cuadrado, sin embargo,
3      para mantener el nombre del archivo y que se ejecute el código, lo mantenemos así*/
4      public static void main(String[] args) {
5          final int LONGITUD_MINIMA_LADO = 1;
6          final int CERO = 0;
7
8          Scanner escaner = new Scanner(System.in);
9          String entradaNumero;
10         int longitudLado;
11
12         do {
13             System.out.print(s:"Introduzca la longitud del lado del cuadrado: ");
14             entradaNumero = escaner.nextLine();
15             longitudLado = Integer.valueOf(entradaNumero);
16         } while (longitudLado < CERO);
17
18         for (int fila = 0; fila < longitudLado; fila++) {
19             for (int columna = 0; columna < longitudLado; columna++) {
20                 System.out.print(s:"#");
21                 if (columna < longitudLado - LONGITUD_MINIMA_LADO) {
22                     System.out.print(s:" ");
23                 }
24             }
25             System.out.println();
26         }
27     }
28 }
```

Explicación:**Nombres con significado:**

- Se ajustaron los nombres scan, numeroString y numeroInt por escaner, entradaNumero y longitudLado respectivamente para reflejar claramente su función dentro del código.

Nombres fáciles de pronunciar:

- Se simplificaron los nombres sin perder su significado para facilitar la pronunciación y comprensión del código.

Nombres que pueden buscarse:

- Se modificó numeroInt por longitudLado para ofrecer una representación más clara del propósito de la variable en el contexto del código.

Nombres de clases y métodos:

- Se renombró la clase a Cuadrado para describir su función principal de dibujar un cuadrado, y se mantuvo el método main como estándar de inicio en Java.

Una sola palabra por concepto:

- Se mantuvo la coherencia en los nombres utilizados a lo largo del código, asegurando consistencia y descripción en cada instancia.

Variables constantes para 0 y 1:

- Se crearon dos constantes LONGITUD_MINIMA_LADO (antes 1) y CERO (antes 0) para representar el valor mínimo del lado del cuadrado y el número que indica un solo carácter respectivamente.

Nombres descriptivos para constantes:

- Se asignaron nombres más claros y descriptivos a las constantes para reflejar mejor su propósito y significado en el contexto del código.

Ejemplo II: Funciones

Antes:

```
DAMIHernandezPuertas_Francisco_FuncionesPrevio.java > ...
1  import java.util.Scanner;
2  public class DAMIHernandezPuertas_Francisco_FuncionesPrevio {
3      public static void main(String[] args) {
4          Scanner scan = new Scanner(System.in);
5          System.out.print(s:"Introduzca una frase : ");
6          String frase = scan.nextLine();
7          if (frase.indexOf(str:" ") >= 0)
8              System.out.println("La frase contiene " + frase.length() + " caracteres y contiene espacios");
9          else
10             System.out.println("La frase contiene " + frase.length() + " caracteres y no contiene espacios");
11             if (frase.length() > 0) {
12                 char primerCaracter = frase.charAt(index:0); // Obtiene el primer carácter de la frase
13                 char ultimoCaracter = frase.charAt(frase.length() - 1); //Obtiene el último carácter de la frase
14
15                 switch (Character.toLowerCase(primerCaracter)) {
16                     case 'a':
17                     case 'e':
18                     case 'i':
19                     case 'o':
20                     case 'u':
21                         System.out.println(x:"La frase empieza por vocal");
22                         break;
23                     default:
24                         System.out.println(x:"La frase empieza por consonante");
25                 }
26
27                 switch (Character.toLowerCase(ultimoCaracter)) {
28                     case 'a':
29                     case 'e':
30                     case 'i':
31                     case 'o':
32                     case 'u':
33                         System.out.println(x:"La frase termina por vocal");
34                         break;
35                     default:
36                         System.out.println(x:"La frase termina por consonante");
37                 }
38             } else {
39                 System.out.println(x:"La frase está vacía.");
40             }
41         }
42     }
```

Después:

```
DAMIHernandezPuertas_Francisco_FuncionesNuevo.java > ...
1  import java.util.Scanner;
2
3  public class DAMIHernandezPuertas_Francisco_FuncionesNuevo {
4      public static void main(String[] args) {
5          Scanner scan = new Scanner(System.in);
6          System.out.print(s:"Introduzca una frase : ");
7          String frase = scan.nextLine();
8
9          imprimirInformacionFrase(frase);
10         verificarYMostrarCaracter(frase, tipo:"inicia");
11         verificarYMostrarCaracter(frase, tipo:"finaliza");
12     }
13
14     private static void imprimirInformacionFrase(String frase) {
15         if (frase.contains(s:" ")) {
16             System.out.println("La frase contiene " + frase.length() + " caracteres y contiene espacios");
17         } else {
18             System.out.println("La frase contiene " + frase.length() + " caracteres y no contiene espacios");
19         }
20     }
21
22     private static void verificarYMostrarCaracter(String frase, String tipo) {
23         if (!frase.isEmpty()) {
24             char caracter = obtenerCaracter(frase, tipo);
25             imprimirTipoCaracter("La frase " + tipo + " por", caracter);
26         } else {
27             System.out.println(x:"La frase está vacía.");
28         }
29     }
30
31     private static char obtenerCaracter(String frase, String tipo) {
32         switch (tipo) {
33             case "inicia":
34                 return (frase.length() > 0) ? frase.charAt(index:0) : '\0';
35             case "finaliza":
36                 return (frase.length() > 0) ? frase.charAt(frase.length() - 1) : '\0';
37             default:
38                 return '\0';
39         }
40     }
41
42     private static void imprimirTipoCaracter(String mensaje, char caracter) {
43         if (esVocal(caracter)) {
44             System.out.println(mensaje + " vocal");
45         } else {
46             System.out.println(mensaje + " consonante");
47         }
48     }
49
50     private static boolean esVocal(char caracter) {
51         switch (Character.toLowerCase(caracter)) {
52             case 'a':
53             case 'e':
54             case 'i':
55             case 'o':
56             case 'u':
57                 return true;
58             default:
59                 return false;
60         }
61     }
62 }
```

Explicación:

1. Funciones específicas y responsabilidad única:

- Antes: El código contenía una función larga y monolítica que realizaba múltiples tareas: verificaba la existencia de espacios, imprimía la longitud de la frase, identificaba vocales o consonantes en el primer y último carácter, y manejaba mensajes de frases vacías.
- Después: Se dividieron las tareas en funciones más específicas: `imprimirInformacionFrase`, `verificarYMostrarCaracter`, `obtenerCaracter` y `imprimirTipoCaracter`. Cada una se encarga de una tarea concreta.

2. Eliminación de código redundante y duplicado:

- Antes: Existía repetición de código en la lógica para identificar el primer y último carácter.
- Después: Se creó la función `obtenerCaracter` para obtener el carácter inicial y final, eliminando la duplicación de código.

3. Nombres descriptivos y claros:

- Antes: Los nombres de las funciones y variables no eran muy descriptivos.
- Después: Se ajustaron los nombres de las funciones y variables para ser más claros y descriptivos, facilitando la comprensión del código.

4. Manejo de lógica y simplificación:

- Antes: La lógica estaba anidada y resultaba compleja de seguir.
- Después: Se simplificó la estructura, reduciendo la complejidad, y se agregó una verificación para manejar frases vacías.

5. Reutilización de código:

- Antes: No había reutilización de código; la lógica se repetía en diferentes partes.
- Después: Se crearon funciones reutilizables (`obtenerCaracter`, `imprimirTipoCaracter`) para manejar operaciones comunes y reducir la duplicación.

Ejemplo III: Comentarios

Antes:

```
DAMIHernandezPuertas_Francisco_ComentariosPrevio.java > ...
1  import java.util.Scanner;
2  public class DAMIHernandezPuertas_Francisco_ComentariosPrevio {
3      public static void main(String[] args) {
4          Scanner scan = new Scanner(System.in);
5          int contadorNumeros = 1;
6          String numeroString;
7          int numero;
8          int numerosPositivos = 0;
9          int numerosPositivosOperacion = 0;
10         int numerosNegativos = 0;
11         int numerosNegativosOperacion = 0;
12         int numerosCero = 0;
13         do{
14             System.out.print("Introduce el " + contadorNumeros + "º número: ");
15             numeroString = scan.nextLine();
16             numero = Integer.valueOf(numeroString);
17             if(numero>0) {
18                 numerosPositivos++;
19                 numerosPositivosOperacion += numero;
20             }
21             else if(numero<0) {
22                 numerosNegativos++;
23                 numerosNegativosOperacion += numero;
24             }
25             else if(numero==0) {
26                 numerosCero++;
27             }
28             else
29                 System.out.println(x:"Número inválido");
30             contadorNumeros++;
31         }while(contadorNumeros<21);
32         System.out.println("Cantidad de números positivos: " + numerosPositivos);
33         System.out.println("Media de números positivos: " + numerosPositivosOperacion/numerosPositivos);
34         System.out.println("Cantidad de números negativos: " + numerosNegativos);
35         System.out.println("Media de números negativos: " + numerosNegativosOperacion/numerosNegativos);
36         System.out.println("Cantidad de ceros: " + numerosCero);
37     }
38 }
```

Después:

```
DAMIHernandezPuertas_Francisco_ComentariosNuevo.java > ...
1  import java.util.Scanner;
2  public class DAMIHernandezPuertas_Francisco_ComentariosNuevo {
3      public static void main(String[] args) {
4          Scanner escaner = new Scanner(System.in);
5
6          // Inicializamos variables para conteo y operaciones
7          int contadorNumeros = 1;
8          String numeroString;
9          int numero;
10         int numerosPositivos = 0;
11         int sumaNumerosPositivos = 0;
12         int numerosNegativos = 0;
13         int sumaNumerosNegativos = 0;
14         int numerosCero = 0;
15
16         // Inicio de la iteración para recoger números
17         do {
18             System.out.print("Introduce el " + contadorNumeros + "º número: ");
19             numeroString = escaner.nextLine();
20             numero = Integer.valueOf(numeroString);
21
22             // clasificación de números en una estructura de control
23             if (numero > 0) {
24                 numerosPositivos++;
25                 sumaNumerosPositivos += numero;
26             } else if (numero < 0) {
27                 numerosNegativos++;
28                 sumaNumerosNegativos += numero;
29             } else if (numero == 0) {
30                 numerosCero++;
31             } else {
32                 System.out.println(x:"Número inválido");
33             }
34
35             contadorNumeros++;
36         } while (contadorNumeros < 21);
37
38         // Impresión de resultados
39         System.out.println("Cantidad de números positivos: " + numerosPositivos);
40         System.out.println("Media de números positivos: " + sumaNumerosPositivos / numerosPositivos);
41         System.out.println("Cantidad de números negativos: " + numerosNegativos);
42         System.out.println("Media de números negativos: " + sumaNumerosNegativos / numerosNegativos);
43         System.out.println("Cantidad de ceros: " + numerosCero);
44     }
45 }
```

Explicación:**Comentarios descriptivos y concisos:**

- Los comentarios agregados son descriptivos y concisos, explicando el propósito de las secciones del código donde se encuentran.

Comentarios redundantes omitidos:

- Omitimos comentarios que digan cómo hace el código para funcionar, y en su lugar, utilizamos comentarios que muestren qué hace el código.

Comentarios innecesarios omitidos:

- No se requieren más comentarios para que el programador sepa qué ocurre ya que el código es autoexplicativo, por lo que evitamos utilizar comentarios que mientan o confundan al lector.