Práctica Clean Code (https://github.com/FranciscoHernandezPuertas/CleanCode1Ev)

Ejemplo I: Nombres

Antes:

DAM1HernandezPuertas_Francisco_NombresPrevio.java > DAM1HernandezPuertas_Francisco_NombresPrevio

```java
import java.util.Scanner;
public class DAM1HernandezPuertas_Francisco_NombresPrevio {
    Run | Debug
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        String numeroString;
        int numeroInt;
        do{
            System.out.print(s:"Introduzca un número para el lado del cuadrado: ");
            numeroString = scan.nextLine();
            numeroInt = Integer.valueOf(numeroString);
        }while(numeroInt<=0);

        for (int i = 0; i < numeroInt; i++) {
            for (int j = 0; j < numeroInt; j++) {
                System.out.print(s:"#");
                if (j < numeroInt - 1) {
                    System.out.print(s:" ");
                }
            }
            System.out.println();
            }
    }
}
```

Después:

DAM1HernandezPuertas_Francisco_NombresNuevo.java > ...

```java
import java.util.Scanner;
public class DAM1HernandezPuertas_Francisco_NombresNuevo { /*Aquí cambiaríamos el nombre de la clase a cuadrado, sin embargo,
    para mantener el nombre del archivo y que se ejecute el código, lo mantenemos así*/
    Run | Debug
    public static void main(String[] args) {
        final int LONGITUD_MINIMA_LADO = 1;
        final int CERO = 0;

        Scanner escaner = new Scanner(System.in);
        String entradaNumero;
        int longitudLado;

        do {
            System.out.print(s:"Introduzca la longitud del lado del cuadrado: ");
            entradaNumero = escaner.nextLine();
            longitudLado = Integer.valueOf(entradaNumero);
        } while (longitudLado < CERO);

        for (int fila = CERO; fila < longitudLado; fila++) {
            for (int columna = CERO; columna < longitudLado; columna++) {
                System.out.print(s:"#");
                if (columna < longitudLado - LONGITUD_MINIMA_LADO) {
                    System.out.print(s:" ");
                }
            }
            System.out.println();
        }
    }
}
```

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 (entradaNumero, escaner, longitudLado, LONGITUD_MINIMA_LADO, CERO).

Nombres que pueden buscarse:

- Se modificó numeroInt por longitudLado para ofrecer una representación más clara del nombre de la variable facilitando su búsqueda.

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.
  Nota: esto debería cumplirse, pero para mantener un nombrado claro de los archivos .java y evitar confusiones, no cambiamos el nombre de la clase (y por tanto, del archivo) para permitir que el código se ejecute.

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 con nombres descriptivos 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, siendo nombres claros y descriptivos es para reflejar mejor su propósito y significado en el contexto del código.

Ejemplo II: Funciones

Antes:

```java
import java.util.Scanner;
public class DAM1HernandezPuertas_Francisco_FuncionesPrevio {
    Run | Debug
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print(s:"Introduzca una frase : ");
        String frase = scan.nextLine();
        if (frase.indexOf(str:" ") >= 0)
            System.out.println("La frase contiene " + frase.length() + " caracteres y contiene espacios");
        else
            System.out.println("La frase contiene " + frase.length() + " caracteres y no contiene espacios");
            if (frase.length() > 0) {
                        char primerCaracter = frase.charAt(index:0); // Obtiene el primer carácter de la frase
                        char ultimoCaracter = frase.charAt(frase.length() - 1); //Obtiene el último carácter de la frase

                        switch (Character.toLowerCase(primerCaracter)) {
                            case 'a':
                            case 'e':
                            case 'i':
                            case 'o':
                            case 'u':
                                System.out.println(x:"La frase empieza por vocal");
                                break;
                            default:
                                System.out.println(x:"La frase empieza por consonante");
                        }

                        switch (Character.toLowerCase(ultimoCaracter)) {
                            case 'a':
                            case 'e':
                            case 'i':
                            case 'o':
                            case 'u':
                                System.out.println(x:"La frase termina por vocal");
                                break;
                            default:
                                System.out.println(x:"La frase termina por consonante");
                        }
                    } else {
                        System.out.println(x:"La frase está vacía.");
                    }
                }
            }
```

Después:

```java
import java.util.Scanner;

public class DAM1HernandezPuertas_Francisco_FuncionesNuevo {
    Run | Debug
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print(s:"Introduzca una frase : ");
        String frase = scan.nextLine();

        imprimirInformacionFrase(frase);
        verificarYMostrarCaracter(frase, tipo:"inicia");
        verificarYMostrarCaracter(frase, tipo:"finaliza");
    }

    private static void imprimirInformacionFrase(String frase) {
        if (frase.contains(s:" ")) {
            System.out.println("La frase contiene " + frase.length() + " caracteres y contiene espacios");
        } else {
            System.out.println("La frase contiene " + frase.length() + " caracteres y no contiene espacios");
        }
    }

    private static void verificarYMostrarCaracter(String frase, String tipo) {
        if (!frase.isEmpty()) {
            char caracter = obtenerCaracter(frase, tipo);
            imprimirTipoCaracter("La frase " + tipo + " por", caracter);
        } else {
            System.out.println(x:"La frase está vacía.");
        }
    }

    private static char obtenerCaracter(String frase, String tipo) {
        switch (tipo) {
            case "inicia":
                return (frase.length() > 0) ? frase.charAt(index:0) : '\0';
            case "finaliza":
                return (frase.length() > 0) ? frase.charAt(frase.length() - 1) : '\0';
            default:
                return '\0';
        }
    }

    private static void imprimirTipoCaracter(String mensaje, char caracter) {
        if (esVocal(caracter)) {
            System.out.println(mensaje + " vocal");
        } else {
            System.out.println(mensaje + " consonante");
        }
    }

    private static boolean esVocal(char caracter) {
        switch (Character.toLowerCase(caracter)) {
            case 'a':
            case 'e':
            case 'i':
            case 'o':
            case 'u':
                return true;
            default:
                return false;
        }
    }
}
```

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:

DAM1HernandezPuertas_Francisco_ComentariosPrevio.java

```java
import java.util.Scanner;
public class DAM1HernandezPuertas_Francisco_ComentariosPrevio {
    Run | Debug
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int contadorNumeros = 1;
        String numeroString;
        int numero;
        int numerosPositivos = 0;
        int numerosPositivosOperacion = 0;
        int numerosNegativos = 0;
        int numerosNegativosOperacion = 0;
        int numerosCero = 0;
        do{
            System.out.print("Introduce el " + contadorNumeros + "º número: ");
            numeroString = scan.nextLine();
            numero = Integer.valueOf(numeroString);
            if(numero>0) {
                numerosPositivos++;
                numerosPositivosOperacion += numero;
            }
            else if(numero<0) {
                numerosNegativos++;
                numerosNegativosOperacion += numero;
            }
            else if(numero==0) {
                numerosCero++;
            }
            else
            System.out.println(x:"Número inválido");
            contadorNumeros++;
        }while(contadorNumeros<21);
        System.out.println("Cantidad de números positivos: " + numerosPositivos);
        System.out.println("Media de números positivos: " + numerosPositivosOperacion/numerosPositivos);
        System.out.println("Cantidad de números negativos: " + numerosNegativos);
        System.out.println("Media de números positivos: " + numerosNegativosOperacion/numerosNegativos);
        System.out.println("Cantidad de ceros: " + numerosCero);
    }
}
```

Después:

DAM1HernandezPuertas_Francisco_ComentariosNuevo.java

```java
import java.util.Scanner;
public class DAM1HernandezPuertas_Francisco_ComentariosNuevo {
    Run | Debug
    public static void main(String[] args) {
        Scanner escaner = new Scanner(System.in);

        // Inicializamos variables para conteo y operaciones
        int contadorNumeros = 1;
        String numeroString;
        int numero;
        int numerosPositivos = 0;
        int sumaNumerosPositivos = 0;
        int numerosNegativos = 0;
        int sumaNumerosNegativos = 0;
        int numerosCero = 0;

        // Inicio de la iteración para recoger números
        do {
            System.out.print("Introduce el " + contadorNumeros + "º número: ");
            numeroString = escaner.nextLine();
            numero = Integer.valueOf(numeroString);

            // Clasificación de números en una estructura de control
            if (numero > 0) {
                numerosPositivos++;
                sumaNumerosPositivos += numero;
            } else if (numero < 0) {
                numerosNegativos++;
                sumaNumerosNegativos += numero;
            } else if (numero == 0) {
                numerosCero++;
            } else {
                System.out.println(x:"Número inválido");
            }

            contadorNumeros++;
        } while (contadorNumeros < 21);

        // Impresión de resultados
        System.out.println("Cantidad de números positivos: " + numerosPositivos);
        System.out.println("Media de números positivos: " + sumaNumerosPositivos / numerosPositivos);
        System.out.println("Cantidad de números negativos: " + numerosNegativos);
        System.out.println("Media de números negativos: " + sumaNumerosNegativos / numerosNegativos);
        System.out.println("Cantidad de ceros: " + numerosCero);
    }
}
```

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 de los que ya hemos establecido para que el programador sepa qué ocurre ya que el código es autoexplicativo, debido a ello, no saturamos el código de comentarios innecesarios, evitando así utilizar comentarios que mientan o confundan al lector.