

Vacuna.java

Método

public void mostrarMenuVacuna()

```
public void mostrarMenuVacuna() {
    Vacuna vacuna = new Vacuna();
    boolean salir = false;
    Scanner sn = new Scanner(System.in);
    int opcion;
    while (!salir) {
        System.out.println("Registrando la vacuna");
        escribirArchivoVacuna("archivo\\archivoVacuna.txt");
        System.out.println("¿Desea ver la cantidad de efectos
adversos?");
        while (!salir) {
            System.out.println("1.Si");
            System.out.println("2.No");
            System.out.println("3.Salir");
            System.out.println("Seleccione una opcion: ");
            opcion = sn.nextInt();
            switch (opcion) {
                case 1:
                    vacuna.informeCantidadEfectosAdversos("archivo\\archivoVacuna.txt");
                    System.out.println("¿Desea salir?");
                    while (!salir) {
                        System.out.println("1.Si");
                        System.out.println("2.No");
                        System.out.println("Seleccione una opcion:
");
                        opcion = sn.nextInt();
                        switch (opcion) {
                            case 1:
                                salir = true;
                                break;
                            case 2:
                                mostrarMenuVacuna();
                                salir = true;
                                break;
                            default:
                                System.out.println("Las opciones
son del 1 al 2, solo numeros enteros");
                                }
                            }
                        salir = true;
                        break;
                    case 2:
                        System.out.println("¿Desea salir?");
                        while (!salir) {
                            System.out.println("1.Si");
                            System.out.println("2.No");
                            System.out.println("Seleccione una opcion:
");
                            opcion = sn.nextInt();
                            switch (opcion) {
```

```

        case 1:
            salir = true;
            break;
        case 2:
            mostrarMenuVacuna();
            salir = true;
            break;
        default:
            System.out.println("Las opciones
son del 1 al 2, solo numeros enteros");
    }
    }
    salir = true;
    break;
case 3:
    salir = true;
    break;
default:
    System.out.println("Las opciones son del 1 al
3, solo numeros enteros");
}
}
}
}

```

Regla

Cognitive Complexity of methods should not be too high (java:S3776)

Adaptability issue | Not focused
 Maintainability
[What is clean code?](#)

Why is this an issue?

How can I fix it?

More Info

Cognitive Complexity is a measure of how hard it is to understand the control flow of a unit of code. Code with high cognitive complexity is hard to read, understand, test, and modify.

As a rule of thumb, high cognitive complexity is a sign that the code should be refactored into smaller, easier-to-manage pieces.

Solución:

- Método mostrarEfectosAdversos: Este método encapsula la lógica para mostrar la cantidad de efectos adversos.
- Método preguntarSalir: Este método maneja la lógica de preguntar si el usuario desea salir o no, evitando bucles anidados dentro de mostrarMenuVacuna.
- Uso de return: En lugar de establecer una variable salir para controlar el flujo, puedes usar return para salir del método cuando sea necesario, lo que simplifica el control de flujo.

```

• public void mostrarMenuVacuna() {
•     escribirArchivoVacuna("archivo\\archivoVacuna.txt");

```

```

•         while (true) {
•             System.out.println("Registrando la vacuna");
•             System.out.println("¿Desea ver la cantidad de efectos
adversos?");
•             System.out.println("1. Sí");
•             System.out.println("2. No");
•             System.out.println("3. Salir");
•             System.out.print("Seleccione una opción: ");
•
•             int opcion = scanner.nextInt();
•             switch (opcion) {
•                 case 1:
•                     mostrarEfectosAdversos();
•                     break;
•                 case 2:
•                     preguntarSalir();
•                     break;
•                 case 3:
•                     return; // Salir del método
•                 default:
•                     System.out.println("Las opciones son del 1 al
3, solo números enteros.");
•             }
•         }
•     }
•
•     private void mostrarEfectosAdversos() {
•         vacuna.informeCantidadEfectosAdversos("archivo\\archivoVacu
na.txt");
•         preguntarSalir();
•     }
•
•     private void preguntarSalir() {
•         while (true) {
•             System.out.println("¿Desea salir?");
•             System.out.println("1. Sí");
•             System.out.println("2. No");
•             System.out.print("Seleccione una opción: ");
•
•             int opcion = scanner.nextInt();
•             switch (opcion) {
•                 case 1:
•                     return; // Salir del método
•                 case 2:
•                     mostrarMenuVacuna();
•                     return; // Salir del método
•                 default:
•                     System.out.println("Las opciones son del 1 al
2, solo números enteros.");

```

```
•     }  
•     }  
• }
```

Metodo

```
public void mostrarMenuVacunaTest()  
public void mostrarMenuVacunaTest() {  
  
    Vacuna vacuna = new Vacuna();  
  
    boolean salir = false;  
  
    Scanner sn = new Scanner(System.in);  
  
    int opcion;  
  
    while (!salir) {  
  
        System.out.println("Registrando la vacuna");  
  
        System.out.println("1.Registrar nueva vacuna");  
  
        System.out.println("2.Mostrar todas las vacunas registradas");  
  
        System.out.println("3.Mostrar los datos de una vacuna especifica");  
  
        System.out.println("4.Mostrar cantidad de efectos adversos");  
  
        System.out.println("5.Salir");  
  
        System.out.println("Seleccione una opcion: ");  
  
        try {  
  
            opcion = sn.nextInt();  
  
            switch (opcion) {  
  
                case 1:  
  
                    vacuna.escribirArchivoVacuna("archivo\\archivoVacuna.txt");  
  
                    break;  
  
                case 2:  
  
                    vacuna.leerArchivoVacuna("archivo\\archivoVacuna.txt");  
  
                    break;  
  
                case 3:  
  
                    vacuna.mostrarVacunaEspecific("archivo\\archivoVacuna.txt");  
  
                    break;  
  
                case 4:  
  
                    vacuna.informeCantidadEfectosAdversos("archivo\\archivoVacuna.txt");  
  
            }  
  
        }  
  
    }  
}
```

```

        break;
    case 5:
        salir = true;
        break;
    default:
        System.out.println("Las opciones son del 1 al 5, solo numeros enteros");
    }
} catch (InputMismatchException e) {
    System.out.println("Las opciones son del 1 al 5, solo numeros enteros");
    sn.next();
}
}
}
}

```

Regla

Local variables should not shadow class fields (java:S1117)

Intentionality issue | Not clear
 Maintainability
[What is clean code?](#)

[Why is this an issue?](#)
[More Info](#)

Shadowing occurs when a local variable has the same name as a variable or a field in an outer scope.

This can lead to three main problems:

- **Confusion:** The same name can refer to different variables in different parts of the scope, making the code hard to read and understand.
- **Unintended Behavior:** You might accidentally use the wrong variable, leading to hard-to-detect bugs.
- **Maintenance Issues:** If the inner variable is removed or renamed, the code's behavior might change unexpectedly because the outer variable is now being used.

To avoid these problems, rename the shadowing, shadowed, or both identifiers to accurately represent their purpose with unique and meaningful names.

Solución:

- **Definición del Logger:** Se define el logger como una constante static para que se comparta entre todas las instancias de VacunaManager.
- **Uso de logger:** Se reemplazan todas las llamadas a System.out.println por logger.info y logger.warning, dependiendo de la naturaleza del mensaje. Esto permite un mejor manejo de los registros.

```

• public void mostrarMenuVacunaTest() {
•     boolean salir = false;
•     while (!salir) {
•         logger.info("Registrando la vacuna");
•         logger.info("1. Registrar nueva vacuna");
•         logger.info("2. Mostrar todas las vacunas
registradas");
•         logger.info("3. Mostrar los datos de una vacuna
específica");
•         logger.info("4. Mostrar cantidad de efectos adversos");
•         logger.info("5. Salir");
•         logger.info("Seleccione una opción: ");
•
•         try {
•             int opcion = sn.nextInt();
•             switch (opcion) {
•                 case 1:
•                     vacuna.escribirArchivoVacuna("archivo\\arch
ivoVacuna.txt");
•                     break;
•                 case 2:
•                     vacuna.leerArchivoVacuna("archivo\\archivoV
acuna.txt");
•                     break;
•                 case 3:
•                     vacuna.mostrarVacunaEspecificas("archivo\\ar
chivoVacuna.txt");
•                     break;
•                 case 4:
•                     vacuna.informeCantidadEfectosAdversos("arch
ivo\\archivoVacuna.txt");
•                     break;
•                 case 5:
•                     salir = true;
•                     break;
•                 default:
•                     logger.warning("Las opciones son del 1 al
5, solo números enteros");
•             }
•             } catch (InputMismatchException e) {
•                 logger.warning("Las opciones son del 1 al 5, solo
números enteros");
•                 sn.next(); // Limpiar el buffer del scanner
•             }
•         }
•     }
• }

```

Metodo

```
private void preguntarSalir() {
```

```

while (true) {
    System.out.println("¿Desea salir?");
    System.out.println("1. Sí");
    System.out.println("2. No");
    System.out.print("Seleccione una opción: ");

    int opcion = scanner.nextInt();
    switch (opcion) {
        case 1:
            return; // Salir del método
        case 2:
            mostrarMenuVacuna();
            return; // Salir del método
        default:
            System.out.println("Las opciones son del 1 al 2, solo
números enteros.");
    }
}
}

```

Regla

Standard outputs should not be used directly to log anything (java:S106)

Adaptability issue | Not modular | **Maintainability** 🚩 | [What is clean code?](#)

[Why is this an issue?](#) | [More Info](#)

In software development, logs serve as a record of events within an application, providing crucial insights for debugging. When logging, it is essential to ensure that the logs are:

- easily accessible
- uniformly formatted for readability
- properly recorded
- securely logged when dealing with sensitive data

Those requirements are not met if a program directly writes to the standard outputs (e.g., `System.out`, `System.err`). That is why defining and using a dedicated logger is highly recommended.

Solucion:

- **Definición del Logger:** El logger se define como una constante static en la clase para ser utilizado en cualquier método de la misma.
- **Uso de logger:** Se reemplazan todas las llamadas a `System.out.println` por `logger.info` para mensajes informativos y `logger.warning` para advertencias. Esto permite un manejo más adecuado de los mensajes de registro.

- **Manejo de Excepciones:** Se agrega un bloque try-catch para capturar posibles excepciones `InputMismatchException`, asegurando que el programa no falle en caso de una entrada no válida.

```
private void preguntarSalir() {
    while (true) {
        logger.info("¿Desea salir?");
        logger.info("1. Sí");
        logger.info("2. No");
        logger.info("Seleccione una opción: ");

        try {
            int opcion = scanner.nextInt();
            switch (opcion) {
                case 1:
                    logger.info("El usuario ha decidido salir.");
                    return; // Salir del método
                case 2:
                    logger.info("El usuario ha decidido no salir.");
                    mostrarMenuVacuna();
                    return; // Salir del método
                default:
                    logger.warning("Las opciones son del 1 al 2, solo números enteros.");
            }
        } catch (InputMismatchException e) {
            logger.warning("Error en la entrada: Las opciones son del 1 al 2, solo números enteros.");
            scanner.next(); // Limpiar el buffer del scanner
        }
    }
}
```