PUCE Virtual

CLASE

2

Programación orientada a objetos

Atributos y responsabilidades de las clases

QS WORLD UNIVERSITY RANKINGS

Educación de calidad

INTRODUCCIÓN DE LA CLASE

GRADO / POSGRADO / TECNOLOGÍAS

Estudia a tu tiempo
sin interrupciones

EDUCACIÓN EN LÍNEA DE CALIDAD

La correcta definición de atributos y responsabilidades de las clases es un paso fundamental en el análisis y modelado orientado a objetos, ya que permite representar de forma precisa las entidades del dominio del problema y delimitar claramente su función dentro del sistema. A través del estudio del estado de un objeto, el estudiante comprende cómo los valores de los atributos determinan la situación actual de una instancia y cómo dichos estados influyen directamente en el comportamiento del sistema. Asimismo, el análisis de la responsabilidad como principio de diseño introduce la importancia de asignar a cada clase un conjunto claro y limitado de funciones, evitando sobrecargas innecesarias que puedan generar errores de diseño, acoplamiento excesivo o riesgos de seguridad.

Por otra parte, la encapsulación se aborda como un principio clave para proteger la integridad de los datos y fortalecer la seguridad del software desde las etapas iniciales del desarrollo. Mediante el ocultamiento de información, se restringe el acceso directo a los atributos internos de las clases, reduciendo la posibilidad de manipulaciones indebidas. Complementariamente, el acceso controlado a los datos permite que las modificaciones al estado de los objetos se realicen bajo reglas definidas, contribuyendo a la confiabilidad y robustez del sistema. En conjunto, estos conceptos permiten al estudiante comprender cómo un adecuado modelado orientado a objetos no solo mejora la calidad del diseño, sino que también constituye un pilar esencial en la construcción de sistemas seguros (Hernández Bejarano, 2023).

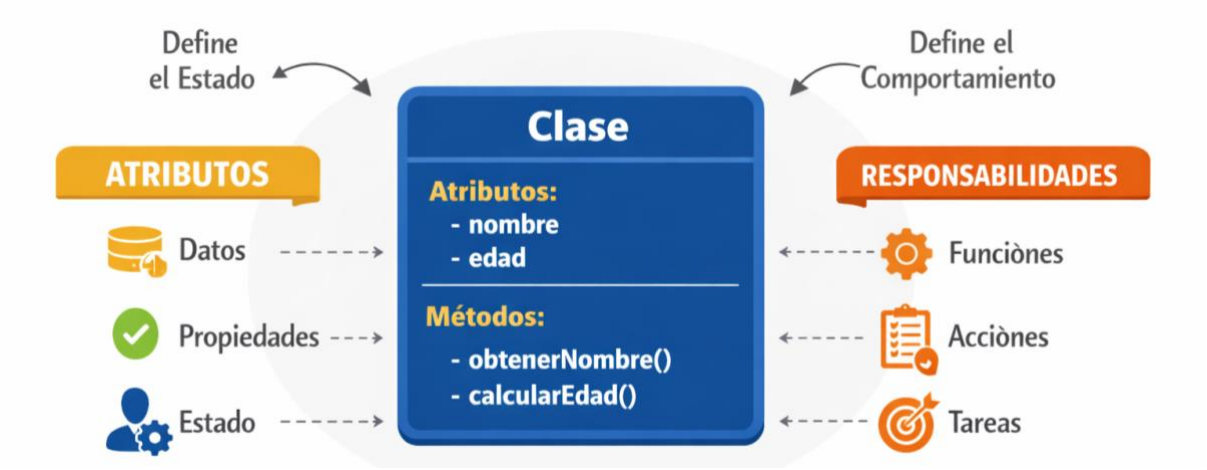
3. Atributos y responsabilidades de las clases

En el análisis orientado a objetos, una clase representa una abstracción de una entidad relevante dentro del dominio del problema. En sistemas con enfoque en ciberseguridad, estas entidades suelen corresponder a conceptos críticos como usuarios, roles, permisos y recursos. Cada clase se define a partir de dos elementos fundamentales: sus atributos, que describen su estado, y sus responsabilidades, que delimitan su función dentro del sistema (Oviedo Regino, 2015).

La Figura 1, sirve como un apoyo visual para comprender el análisis y modelado de clases en POO, especialmente en etapas tempranas de diseño.

Figura 1

Atributos y responsabilidades de una clase



Autor: Héctor Avalos Silva

Los atributos de una clase representan la información necesaria para describir una entidad dentro de un sistema orientado a objetos y permiten definir el estado de los objetos que se crean a partir de ella. Se identifican a partir de las características relevantes del dominio del problema y responden a la pregunta sobre qué datos debe almacenar la clase para cumplir su función. En sistemas como el control de accesos, atributos como nombre, correo, rol y estado activo permiten diferenciar a cada objeto, explicando por qué instancias de una misma clase pueden comportarse de manera distinta.

En el sistema de control de acceso de ejemplo, consideremos la clase Usuario con sus atributos nombre, email, rol y activo:

```
//definición de la clase Usuario con sus atributos
class Usuario {
   String nombre; //atributo nombre
   String email; //atributo email
   String rol; //atributo rol
   boolean activo;//atributo activo
}
```

Definir adecuadamente los atributos de una clase implica identificar qué información es necesaria para representar la entidad sin exponer datos innecesarios.

Asignar responsabilidades claras evita que una clase concentre funciones que no le corresponden, reduciendo el riesgo de errores de diseño y vulnerabilidades.

3.1 Estado de un objeto

El estado de un objeto está determinado por los valores que toman sus atributos en un momento específico de la ejecución del sistema. La Figura 2 ilustra el estado de un objeto de la clase Usuario, sus atributos al momento tienen valores específicos.

Figura 2

Estado de un objeto (valor de sus atributos) de la clase Usuario

Autor: Héctor Avalos Silva

Cada instancia (objeto) de una clase mantiene su propio estado, el cual puede variar con el tiempo como resultado de las operaciones realizadas sobre ella. (Blanco Fernández, 2020) señala que

comprender el estado de los objetos es esencial para razonar sobre el comportamiento del sistema y anticipar posibles errores lógicos.

En un sistema de control de accesos, el estado de un objeto es especialmente sensible. Por ejemplo, el estado de un objeto Usuario puede indicar si una persona está habilitada para acceder al sistema, qué rol posee y qué nivel de privilegios tiene. Un cambio no controlado en el estado de un objeto puede permitir accesos indebidos o escalamiento de privilegios.

La Figura 3, ilustra los diferentes estados de un objeto (atributos de un objeto con diferentes valores en momentos distintos) de una clase.

Figura 3

Diferentes estados de un objeto de la clase Usuario

Autor: Héctor Avalos Silva

Ejemplo conceptual de implementación en Java de la clase Usuario con sus atributos y sus estados:

```
//Definición de la clase Usuario
class Usuario {
   String nombre;
   String email;
   String rol;
   boolean activo;
}
```

Esta definición establece el conjunto de atributos que describen el estado potencial de cualquier usuario del sistema. Sin embargo, es importante destacar que definir una clase no crea usuarios, sino que establece la estructura que tendrán todas las instancias (Hernández Bejarano, 2023).

```
public class Ejemplo {  //Ejemplo es la clase del programa principal Java

   public static void main(String[] args) {

      Usuario usuario1 = new Usuario(); //creación del 1er objeto Usuario (se instancia una clase)

      //estado del objeto usuario1
      usuario1.nombre = "María López";
      usuario1.email = "mlopez@institucion.edu";
      usuario1.rol = "Administrativo";
      usuario1.activo = true;
```

```
        Usuario usuario2 = new Usuario(); //Creación del 2do objeto de tipo Usuario

        //estado del objeto usuario2
        usuario2.nombre = "Juan Pérez";
        usuario2.email = "jperez@institucion.edu";
        usuario2.rol = "Estudiante";
        usuario2.activo = false;

        //cambio de estado del objeto usuario1
        usuario1.email = "mlopez2026@institucion.edu";
        usuario1.activo = false;
    }
}
```

Explicación:

En este caso, usuario1 representa un objeto concreto con un estado específico. Desde la perspectiva de la seguridad, el sistema debe asegurar que solo los componentes autorizados puedan modificar atributos como rol o activo (Gervais, 2025).

Las dos sentencias últimas del código cambian de estado al objeto usuario1:

usuario1.email = "mlopez2026@institucion.edu";

usuario1.activo = false;

Se cambia los valores de los atributos email y activo.

3.2 Responsabilidad como principio de diseño

La responsabilidad como principio de diseño es uno de los pilares del análisis y diseño orientado a objetos. Este principio establece que cada clase debe tener claramente definido qué hace y de qué información se encarga, evitando asumir tareas que no le corresponden. En otras palabras, una clase debe existir por una razón concreta dentro del sistema y cumplir un propósito bien delimitado (Blanco Fernández, 2020).

Desde el punto de vista del diseño, asignar responsabilidades correctamente permite construir sistemas cohesivos, mantenibles y seguros, ya que cada clase concentra su lógica en torno a una función específica. Cuando una clase asume demasiadas responsabilidades, el diseño se vuelve frágil: los cambios en una parte del sistema afectan a otras de forma inesperada, se incrementa el acoplamiento y se dificulta la comprensión del código (Hernández Bejarano, 2023).

¿Qué es una responsabilidad en POO?

En POO, una responsabilidad puede entenderse como:

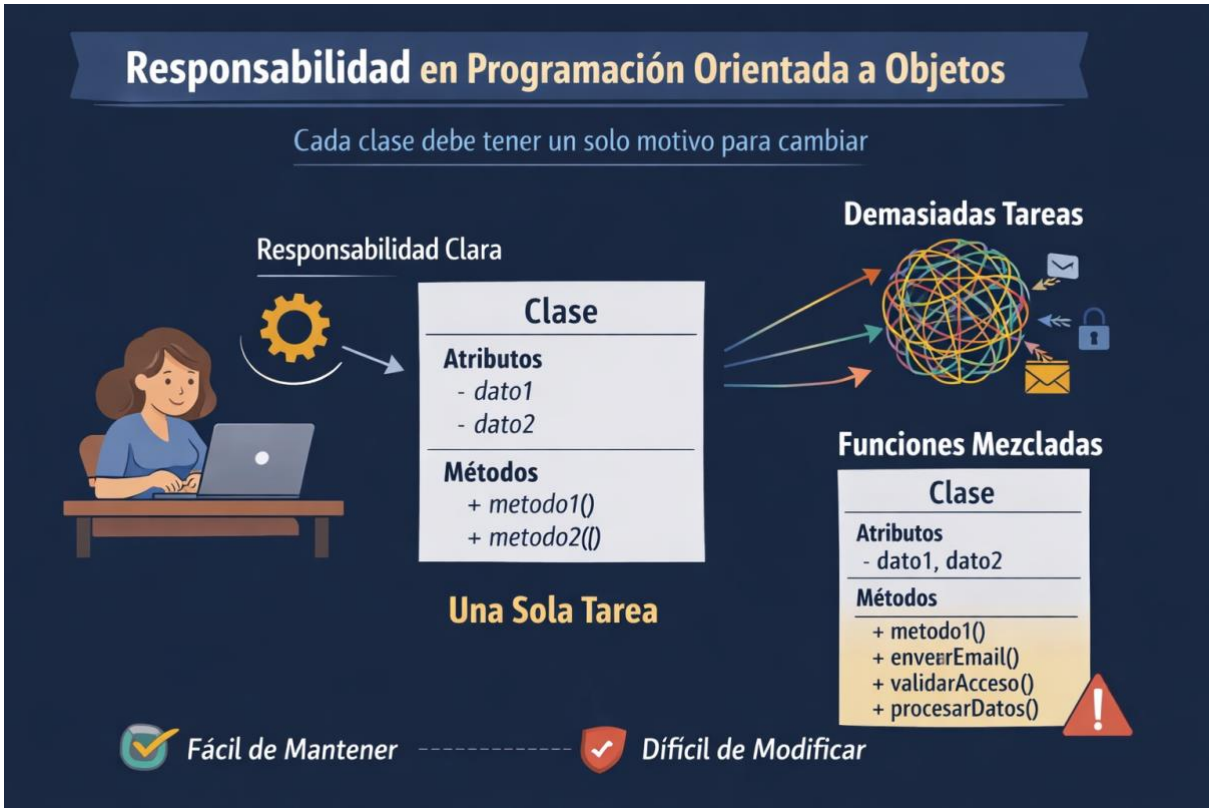
- Una acción que la clase debe realizar

- Una decisión que la clase debe tomar
- La gestión de cierta información relacionada con su propósito

La Figura 4 representa el concepto de responsabilidad en la POO, mostrando que cada clase debe cumplir una única función clara dentro del sistema. Se contrasta una clase con responsabilidades bien definidas (más fácil de mantener y segura), frente a otra que concentra demasiadas tareas, lo que provoca desorden, mayor complejidad y riesgos en el diseño.

Figura 4

Responsabilidad en POO



Autor: Héctor Avalos Silva

Por ejemplo, en un sistema de control de accesos:

- La clase Usuario es responsable de conocer su estado (activo/inactivo)
- La clase Rol es responsable de agrupar permisos
- La clase Permiso es responsable de describir una acción autorizada
- La clase Recurso es responsable de representar un elemento protegido del sistema

Cada una cumple un rol específico y no invade las responsabilidades de las demás.

La Figura 5 refuerza la idea de que un diseño orientado a objetos bien estructurado contribuye directamente a la protección del sistema, evitando concentrar decisiones críticas en una sola clase.

Figura 5

Distribución conceptual de responsabilidades en un sistema seguro.

Autor: Héctor Avalos Silva.

Principio de responsabilidad única (SRP)

En el enfoque de análisis y diseño orientado a objetos, donde se asignan responsabilidades claras a cada clase, este criterio se relaciona directamente con el Principio de Responsabilidad Única, el cual indica que una clase debe tener una sola razón para cambiar.

Esto significa que, si una clase cambia por múltiples motivos (lógica de negocio, validaciones, persistencia, seguridad), entonces tiene demasiadas responsabilidades y debe ser refactorizada (Hernández Bejarano, 2023).

Responsabilidades conceptuales vs. técnicas

Durante el análisis orientado a objetos se definen principalmente responsabilidades conceptuales, es decir, qué debe hacer la clase según el dominio del problema. Las responsabilidades técnicas, como el uso de métodos, validaciones o estructuras de datos se abordan posteriormente durante el diseño y la implementación (Oviedo Regino, 2015).

Beneficios de una correcta asignación de responsabilidades

Aplicar este principio permite:

- Mayor cohesión de las clases
- Menor acoplamiento entre componentes
- Código más legible y mantenible
- Mayor facilidad para extender el sistema
- Mejor alineación con requisitos de seguridad

En sistemas sensibles, como los de control de accesos, una mala asignación de responsabilidades puede generar vulnerabilidades, por ejemplo, cuando una clase expone información o toma decisiones que no le corresponden (Phillips & Plott, 2021).

Ejemplo aplicado en Java: Sistema de control de accesos

- **Ejemplo incorrecto (mala asignación de responsabilidades)**

```
class Usuario {
   String nombre;
   String rol;

   public boolean puedeAcceder(String recurso) {
      // Lógica de permisos mezclada con Usuario
      if (rol.equals("Administrador")) {
         return true;
      }
      return false;
   }
}
```

Problemas:

- Usuario asume responsabilidades de autorización
- Dificulta la extensión del sistema
- Viola el principio de responsabilidad única

- **Ejemplo correcto (responsabilidades bien distribuidas)**

```
class Usuario {
   private String nombre;
   private Rol rol;

   public Usuario(String nombre, Rol rol) {
      this.nombre = nombre;
      this.rol = rol;
   }

   public Rol obtenerRol() {
      return rol;
   }
```

```
}
class Rol {
    private String nombre;

    public Rol(String nombre) {
        this.nombre = nombre;
    }

    public boolean tieneAccesoA(Recurso recurso) {
        // Lógica de autorización asociada al rol
        return true; // Simplificado para el ejemplo
    }
}

class Recurso {
    private String identificador;

    public Recurso(String identificador) {
        this.identificador = identificador;
    }
}
```

En este diseño:

- Usuario es responsable de representar al usuario
- Rol es responsable de decidir permisos
- Recurso representa el elemento protegido
- Cada clase cumple una responsabilidad clara

Relación con el diseño orientado a objetos

Este principio se define antes del diseño detallado y guía decisiones posteriores como:

- Definición de interfaces
- Estructura interna de las clases
- Mecanismos de control de acceso
- Reglas de validación

Una vez asignadas correctamente las responsabilidades, el sistema está preparado para evolucionar hacia el diseño y la implementación sin perder coherencia.

Observación: En POO, no se pregunta primero "qué métodos tiene la clase", sino "qué responsabilidad cumple en el sistema".

Cuando cada clase tiene responsabilidades bien definidas, el sistema se vuelve más claro, organizado y fácil de ampliar.

El siguiente enlace explica cómo se identifican clases, atributos y responsabilidades en un modelo orientado a objetos partiendo del dominio del problema.

- **Título:** Modelo mental de POO: clases, objetos y responsabilidades
- **Descripción**: Incluye una guía práctica para encontrar comportamientos (responsabilidades) y cómo ubicarlos en las clases que los deben asumir, relacionándose directamente con el análisis de atributos y responsabilidades en la POO
- **Enlace:** https://cursa.app/es/pagina/modelo-mental-de-poo-clases-objetos-y-responsabilidades

4. Encapsulación

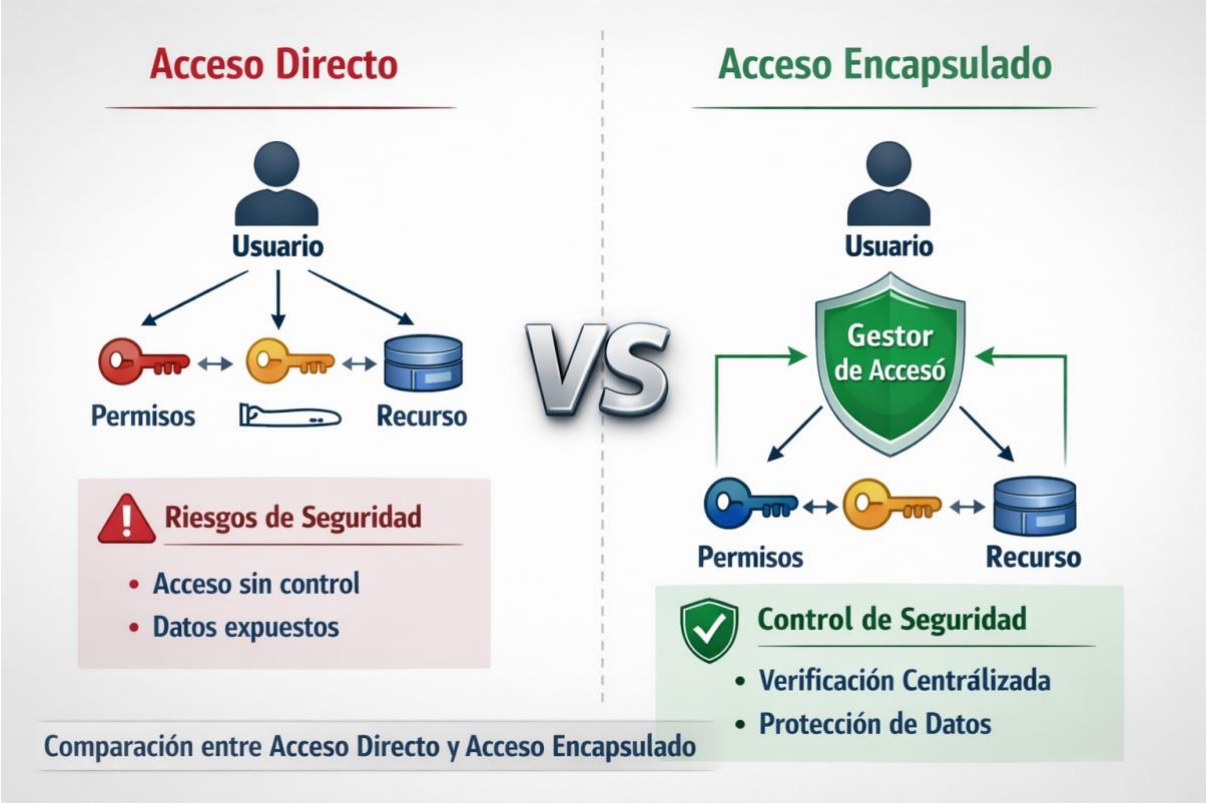
La encapsulación es uno de los pilares fundamentales de la POO y tiene una relación directa con la seguridad del software. Este principio establece que los detalles internos de una clase deben estar protegidos y que el acceso a sus datos debe realizarse de forma controlada. Phillips & Plott (2021) destacan que la encapsulación no solo mejora la mantenibilidad del código, sino que también reduce la superficie de ataque del sistema (métodos o funciones accesibles, interfaces de usuario, entradas de datos, servicios o componentes expuestos, accesos a información o estado interno).

En Java, por ejemplo, se encapsula una clase cuando sus atributos son private y solo se accede a ellos mediante métodos controlados (get, set u otros), reduciendo riesgos y errores (Hernández Bejarano, 2023).

Para una mejor comprensión, la Figura 6 indica que el acceso encapsulado es un enfoque más seguro y alineado con los principios de la POO, ya que protege el estado interno del sistema, distribuye responsabilidades y minimiza riesgos de seguridad frente al acceso directo.

Figura 6

Comparación entre acceso directo y acceso encapsulado



Autor: Héctor Avalos Silva

4.1 Ocultamiento de información

El ocultamiento de información consiste en restringir el acceso directo a los atributos de una clase, permitiendo que solo puedan ser consultados o modificados a través de mecanismos definidos. En términos de ciberseguridad, la encapsulación actúa como una barrera que limita las posibilidades de manipulación indebida del estado de un objeto (Phillips & Plott, 2021).

Ejemplo sin encapsulación:

```
class Usuario {
   public String rol;
}
```

Este diseño permite que cualquier parte del sistema modifique el rol del usuario sin restricciones, lo cual representa un riesgo significativo.

Ejemplo con encapsulación:

```
class Usuario {
   private String rol;
   public String getRol() {
      return rol;
   }
}
```

Encapsular los datos reduce la superficie de ataque del sistema.

4.2 Acceso controlado a los datos

El acceso controlado implica que las modificaciones al estado de un objeto se realicen bajo reglas claras. En lugar de permitir cambios directos, la clase define operaciones específicas (métodos) para modificar su estado (Hernández Bejarano, 2023).

La Figura 7 ilustra cómo el acceso encapsulado protege el estado del objeto, reduce riesgos de seguridad y asegura que las reglas de negocio y validación se apliquen antes de cualquier cambio interno.

Figura 7

Flujo de acceso controlado al estado de un objeto.

Autor: Héctor Avalos Silva.

Implementación en Java de acceso controlado al estado de un objeto.

```
class Usuario {
   private boolean activo;

   public void activarUsuario() {
      this.activo = true;
   }

   public void desactivarUsuario() {
      this.activo = false;
   }
}
```

Este enfoque permite incorporar validaciones y controles sin exponer directamente los atributos. En sistemas críticos, este principio es esencial para prevenir errores y ataques.

El siguiente video aborda el concepto de encapsulación en POO, explicando cómo se ocultan los atributos y cómo se controla el acceso mediante métodos.

- **Título:** java 2020 programación orientado a objetos
- **Descripción:** Video relacionado con los temas de ocultamiento de información y acceso controlado a los datos
- **Enlace**: (1614) CURSO de JAVA 2020 POO ☕ ENCAPSULAMIENTO - YouTube

Glosario:

Blanco Fernández, Y. (2020). *Introducción a la programación orientada a objetos.* Andavira.

Gervais, L. (2025). *Aprender Programación Orientada a Objetos con Java.* (eni, Ed.) https://doi.org/978-2-409-05028-2

Guardati Buemo, S. (2016). *Estructuras de datos básicas: Programación orientada a objetos con Java.* Alfaomega.

Hernández Bejarano, M. &. (2023). *Programación Orientada a Objetos en Java: Buenas prácticas. Ingeniería de sistemas.* https://doi.org/978-958-792-463-3

Oviedo Regino, E. M. (2015). *Lógica de programación orientada a objetos.* Ecoe Ediciones. https://doi.org/978-958-711-136-3

Phillips, D., & Plott, S. F. (2021). *Python Object-Oriented Programming: Build robust and maintainable object-oriented Python applications and libraries* (4ta ed.). Packt Publishing Limited. https://doi.org/978-1801077262

Definiciones:

- **Encapsulación:** Principio de la Programación Orientada a Objetos que consiste en ocultar los detalles internos de una clase y controlar el acceso a sus datos mediante interfaces definidas.
- **Responsabilidad:** Conjunto de funciones y conocimientos que una clase debe asumir dentro de un sistema, evitando asumir tareas que no le corresponden.

Recursos de profundizacion:

Recurso 1:

- **Título:** Preguntas y respuestas de análisis de atributos, responsabilidades y encapsulación en sistemas seguros.

- **Descripción:** Es un recurso de profundización basado en preguntas y respuestas que analiza un sistema de control de accesos seguro, permitiendo comprender cómo la correcta definición de atributos, responsabilidades y encapsulación en la programación orientada a objetos contribuye a reducir riesgos de seguridad desde el diseño.
- **Documento:** Recurso de profundización 1 Preguntas y respuestas.docx

Recurso 2:

- **Título:** Ejercicio guiado de análisis de atributos, responsabilidades y encapsulación en un sistema de control de accesos.
- **Descripción:** El recurso consiste en un ejercicio guiado que permite aplicar de forma práctica el análisis de atributos, responsabilidades y encapsulación en un sistema de control de accesos, facilitando la comprensión de cómo un modelado orientado a objetos adecuado contribuye a la seguridad del sistema desde la etapa de diseño.
- **Documento:** Recurso de profundización 2 Ejemplo guiado.docx

ACTIVIDAD DE EVALUACION:

PREGUNTAS DE AUTOEVALUACION

INSTRUCCIONES	Lea cuidadosamente cada pregunta y seleccione la alternativa que considere correcta. Una vez respondida cada pregunta, revise la retroalimentación proporcionada para reforzar o corregir su comprensión de los conceptos relacionados con el análisis y modelado orientado a objetos con enfoque en ciberseguridad.					
Titulo	Evaluación formativa: Atributos, responsabilidades y encapsulación en POO					
Dificultad		Baja	**X**	Media		Alta
RDAs Evaluados:	**X**	RDA 1				
		RDA 2				
		RDA 3				
Secciones evaluadas	1	1.3 Atributos y responsabilidades de las clases 1.3.1 Estado de un objeto 1.3.2 Responsabilidad como principio de diseño 1.4 Encapsulación 1.4.1 Ocultamiento de información 1.4.2 Acceso controlado a los datos				
PREGUNTA 1	¿Cuál de las siguientes opciones describe correctamente el concepto de estado de un objeto en un sistema orientado a objetos?					
CORRECTA	El conjunto de valores que toman los atributos de un objeto en un momento determinado.					
Incorrecta 1	El conjunto de métodos que definen el comportamiento de una clase.					

Incorrecta 2	La relación de herencia que existe entre distintas clases.
Incorrecta 3	El rol que cumple una clase dentro del sistema.
Retroalimentación de las respuestas	
Correcta	El estado de un objeto está determinado por los valores actuales de sus atributos, lo que explica su comportamiento en un momento específico del sistema (Oviedo Regino, 2015; Blanco Fernández, 2020).
Incorrecta 1	Los métodos representan comportamientos o responsabilidades, no el estado del objeto.
Incorrecta 2	La herencia es un mecanismo de reutilización, no una característica del estado.
Incorrecta 3	El rol de una clase se relaciona con sus responsabilidades, no con el estado de sus objetos.
PREGUNTA 2	Desde el principio de responsabilidad como criterio de diseño, ¿cuál de las siguientes situaciones representa una mala asignación de responsabilidades con impacto en la seguridad?
CORRECTA	Una clase que gestiona datos del usuario y además decide permisos de acceso al sistema.
Incorrecta 1	Una clase que representa a un usuario y mantiene su información básica.
Incorrecta 2	Una clase que agrupa permisos asociados a un rol.
Incorrecta 3	Una clase que representa un recurso protegido del sistema.
Retroalimentación de las respuestas	
CORRECTA	Cuando una clase asume múltiples responsabilidades (datos, decisiones de seguridad, lógica de negocio), se incrementa el acoplamiento y el riesgo de errores o vulnerabilidades (Hernández Bejarano & Baquero Rey, 2023).
Incorrecta 1	Representar información básica del usuario es una responsabilidad válida y coherente.
Incorrecta 2	Agrupar permisos es una responsabilidad propia del rol dentro del dominio del problema.
Incorrecta 3	Representar un recurso no implica tomar decisiones de acceso, por lo que no viola el principio.
PREGUNTA 3	¿Cuál es el principal objetivo de la encapsulación en un sistema orientado a objetos con enfoque en ciberseguridad?
CORRECTA	Proteger el estado interno de los objetos y controlar el acceso a sus datos.
Incorrecta 1	Aumentar la cantidad de métodos públicos disponibles.

Incorrecta 2	Facilitar la herencia entre clases.
Incorrecta 3	Permitir el acceso directo a los atributos desde cualquier parte del sistema.
Retroalimentación de las respuestas	
CORRECTA	La encapsulación limita el acceso directo al estado interno, reduce la superficie de ataque y mejora la seguridad del sistema (Phillips & Plott, 2021; Hernández Bejarano & Baquero Rey, 2023).
Incorrecta 1	Un exceso de métodos públicos puede incrementar los riesgos de uso indebido.
Incorrecta 2	La herencia es un mecanismo distinto que no garantiza protección de datos.
Incorrecta 3	El acceso directo a atributos expone el sistema a manipulaciones indebidas.
PREGUNTA 4	¿Cuál de las siguientes prácticas refleja correctamente el acceso controlado a los datos en Java?
CORRECTA	Modificar el estado de un objeto únicamente a través de métodos que aplican reglas y validaciones.
Incorrecta 1	Declarar todos los atributos como públicos para facilitar el acceso.
Incorrecta 2	Permitir que cualquier clase modifique directamente los atributos.
Incorrecta 3	Acceder a los datos sin restricciones para mejorar el rendimiento.
Retroalimentación de las respuestas	
CORRECTA	El acceso controlado asegura que los cambios al estado del objeto se realicen bajo reglas definidas, fortaleciendo la seguridad y la coherencia del sistema (Guardati Buemo, 2016; Oviedo Regino, 2015).
Incorrecta 1	Los atributos públicos eliminan el control y aumentan la superficie de ataque.
Incorrecta 2	El acceso indiscriminado rompe la encapsulación y facilita errores y vulnerabilidades.
Incorrecta 3	La falta de restricciones puede comprometer la integridad y seguridad del sistema.

PUCE Virtual

síguenos en:

www.pucevirtual.puce.edu.ec