PUCE Virtual

CLASE

1

Programación orientada a objetos

Análisis y modelado orientado a objetos

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 Programación Orientada a Objetos (POO) constituye un enfoque fundamental para el análisis y diseño de sistemas de software modernos, ya que permite modelar problemas complejos a partir de entidades que representan elementos del mundo real y sus interacciones. En el contexto de la ciberseguridad, este paradigma adquiere especial relevancia porque facilita la estructuración de sistemas más robustos, controlados y mantenibles, donde la protección de la información puede integrarse desde las primeras etapas del diseño. Los fundamentos de la POO, basados en el paradigma orientado a objetos, proporcionan un marco conceptual que define cómo se organizan las responsabilidades, se encapsulan los datos sensibles y se establecen mecanismos de control que reducen errores de diseño y vulnerabilidades estructurales en el software.

En esta unidad se abordan las ventajas de la Programación Orientada a Objetos para la construcción de sistemas de software seguros, analizando cómo conceptos como clases y objetos permiten separar responsabilidades, controlar el acceso a la información y representar de manera precisa los componentes de un sistema. La distinción entre clase y objeto y la correcta representación de los objetos dentro de un sistema resultan esenciales para modelar escenarios como el control de accesos, la gestión de usuarios, roles y permisos. De este modo, el análisis y modelado orientado a objetos se convierte en una herramienta clave para comprender, diseñar y anticipar comportamientos del software, fortaleciendo la seguridad desde el diseño y alineando la solución técnica con los principios fundamentales de la ciberseguridad.

1. Fundamentos de la programación orientada a objetos

La programación orientada a objetos (POO) es un enfoque de desarrollo de software que organiza los sistemas a partir de entidades del mundo real o conceptual, denominadas objetos, los cuales encapsulan datos y comportamientos relacionados. Este paradigma resulta especialmente relevante en el ámbito de la ciberseguridad, ya que permite modelar de manera clara y controlada elementos críticos como usuarios, roles, permisos, recursos y políticas de acceso.

Desde la perspectiva de la ingeniería de software segura, la POO facilita la construcción de sistemas modulares, robustos y mantenibles, cualidades esenciales para reducir vulnerabilidades, controlar superficies de ataque y asegurar la correcta gestión de la información sensible (Hernández Bejarano, 2023).

Por ejemplo, en un *Sistema de control de acceso institucional*, podemos pensar en objetos como: Usuario, Rol, Permiso, Recurso, etc. Cada uno cumple una función específica dentro del sistema.

En la Figura 1 se muestra un ejemplo de los objetos Usuario, Rol y Recurso.

Figura 1

Ejemplo de objetos Usuario, Rol y Recurso en un Sistema de control de acceso institucional.

Autor: Héctor Avalos Silva.

1.1 PARADIGMA ORIENTADO A OBJETOS Y SU PROPÓSITO

El paradigma orientado a objetos (POO como paradigma) es una forma de pensar, analizar y modelar sistemas. No se centra aún en escribir código, sino en cómo se concibe el problema, cómo se representan sus elementos y cómo interactúan entre sí.

Un paradigma define:

- Qué se considera una entidad relevante.
- Cómo se relacionan esas entidades.
- Cómo se distribuyen responsabilidades.
- Cómo se controla la complejidad del sistema.

En este paradigma, el sistema se entiende como un conjunto de objetos que colaboran, cada uno con responsabilidades bien definidas (Oviedo Regino, 2015).

Este paradigma ayuda a que los programas se parezcan más a la realidad.

En sistemas de ciberseguridad, este enfoque ayuda a controlar mejor quién puede acceder a qué recurso.

Propósitos del paradigma orientado a objetos

a) Modelar la realidad de forma natural

Permite representar elementos del mundo real (usuarios, recursos, procesos) como objetos, facilitando el entendimiento del sistema desde el análisis.

b) Gestionar la complejidad

Divide sistemas grandes en componentes independientes y colaborativos, reduciendo el acoplamiento y aumentando la cohesión.

c) Definir responsabilidades claras

Cada objeto tiene una función específica, evitando concentrar demasiada lógica en un solo componente.

d) Facilitar el diseño antes de programar

Ayuda a pensar el sistema correctamente antes de implementarlo, disminuyendo errores de diseño que luego son costosos de corregir.

e) Servir de base para la construcción del software

Establece el marco conceptual que luego se implementa mediante la programación orientada a objetos.

Desde la ciberseguridad, el paradigma orientado a objetos tiene como propósito identificar activos, actores, riesgos y controles desde el diseño del sistema, permitiendo:

- Detectar superficies de ataque
- Aplicar separación de responsabilidades
- Integrar la seguridad desde el inicio

En relación con el Reto 1 (Análisis inicial del problema), el objetivo es analizar un problema sencillo de control de accesos y reconocer los elementos principales del sistema.

1.1.1 PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos (POO como técnica de desarrollo) es la implementación práctica del paradigma orientado a objetos en un lenguaje de programación. Aquí ya no se analiza el problema, sino que se construye la solución mediante:

- Clases
- Objetos
- Métodos
- Relaciones como herencia, composición e interfaces

Mientras el paradigma responde al cómo pensar, la programación orientada a objetos responde al cómo construir (Hernández Bejarano, 2023).

El siguiente enlace nos lleva a un video que refuerza el concepto de la POO, expone la definición y significado práctico de los conceptos de clase, objeto, instancia, propiedades, atributos:

- Título**:** ¿Qué es la Programación Orientada a Objetos?
- Descripción**:** Video introductorio que explica la POO con ejemplos sencillos y lenguaje claro
- Enlace**:** https://www.youtube.com/watch?v=2rHmroUMJaQ

Propósito de la programación orientada a objetos en ciberseguridad

Desde la ciberseguridad, la POO tiene como propósito implementar mecanismos concretos que protejan el sistema frente a fallos, abusos y ataques, garantizando confidencialidad, integridad y disponibilidad.

a) Proteger datos mediante encapsulación

La encapsulación permite ocultar los datos internos de un objeto y controlar su acceso exclusivamente a través de métodos definidos.

En términos de seguridad, esto permite:

- Proteger información sensible (credenciales, claves, tokens).
- Evitar accesos directos no autorizados.
- Reducir errores de manipulación de datos.

La encapsulación actúa como una barrera lógica de seguridad dentro del software (Plott & Phillips, 2021).

b) Controlar accesos y privilegios

La POO facilita la implementación de controles de acceso mediante:

- Clases que representan roles.
- Métodos que validan permisos.
- Interfaces que limitan capacidades.

Esto permite aplicar de forma técnica el principio de mínimo privilegio, clave para prevenir ataques internos y externos.

c) Reducir errores que generan vulnerabilidades

Muchos problemas de seguridad surgen por:

- Código duplicado.
- Lógica dispersa.
- Falta de control sobre dependencias.

La programación orientada a objetos promueve:

- Reutilización controlada.
- Código más legible.
- Mantenimiento seguro.

Un sistema más mantenible es también un sistema más seguro (Guardati Buemo, 2016).

d) Facilitar auditoría y mantenimiento de seguridad

Gracias a su estructura modular, la POO permite:

- Auditar componentes críticos.
- Actualizar mecanismos de seguridad.
- Corregir vulnerabilidades sin afectar todo el sistema.

Esto es crucial frente a la evolución constante de amenazas.

e) Implementar patrones de diseño seguro

La programación orientada a objetos permite aplicar patrones que fortalecen la seguridad, como:

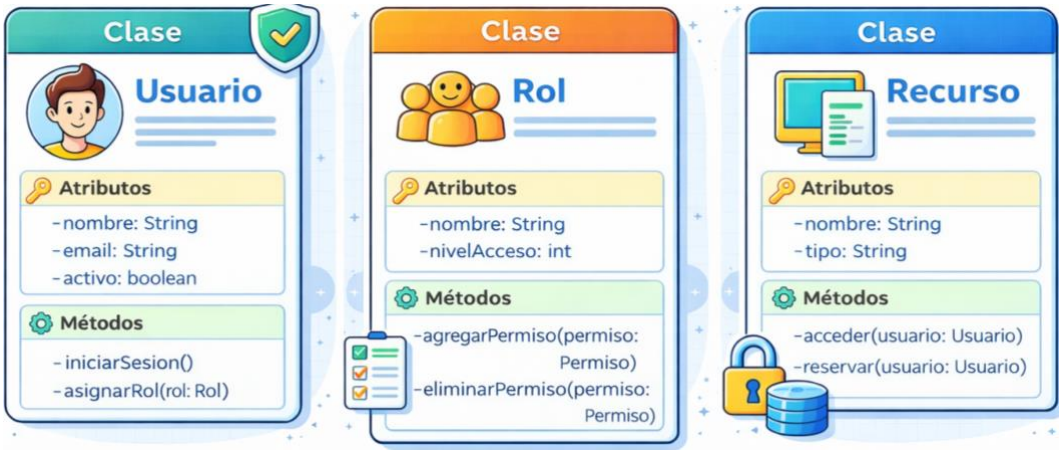
- Controladores de acceso.
- Fachadas de seguridad.
- Objetos inmutables para datos críticos.

Estos patrones ayudan a prevenir fallos de diseño que podrían ser explotados.

En la figura 2, se muestra las clases principales del problema en análisis.

Figura 2

Identificación básica de clases: Usuario, Rol y Recurso.



Autor: Héctor Avalos Silva.

Identificar bien las clases desde el inicio facilita el desarrollo del sistema más adelante.

1.2 VENTAJAS DE LA POO EN LA ESTRUCTURACIÓN DE SISTEMAS DE SOFTWARE

La programación orientada a objetos ofrece múltiples ventajas para la estructuración de sistemas, particularmente relevantes en aplicaciones donde la seguridad de la información es un requisito crítico.

a) Encapsulación y protección de datos

La encapsulación permite ocultar los detalles internos de un objeto y controlar el acceso a sus datos a través de métodos bien definidos. Desde el punto de vista de la ciberseguridad, esto evita manipulaciones directas de información sensible y reduce la posibilidad de errores o abusos por parte de otros componentes del sistema (Phillips, 2021).

Por ejemplo, las credenciales de un usuario no deberían ser accesibles directamente, sino gestionadas mediante métodos seguros de autenticación y validación.

b) Modularidad y reducción de la superficie de ataque

La POO fomenta la modularidad, dividiendo el sistema en componentes independientes. Cada módulo bien diseñado reduce la superficie de ataque, ya que limita las interacciones no controladas entre partes del sistema (Hernández Bejarano, 2023).

En un sistema orientado a objetos, una vulnerabilidad en una clase no necesariamente compromete todo el sistema si las dependencias están correctamente definidas.

c) Reutilización segura mediante herencia y composición

La herencia y la composición permiten reutilizar código probado y validado, lo cual disminuye la probabilidad de introducir fallos de seguridad al reescribir funcionalidades críticas. No obstante, en contextos de seguridad se recomienda un uso cuidadoso, priorizando la composición para evitar exposiciones indebidas de comportamiento (Guardati Buemo, 2016).

d) Facilidad para aplicar principios de diseño seguro

La POO facilita la aplicación de principios de diseño como:

- Separación de responsabilidades.
- Principio de responsabilidad única.
- Abstracción de mecanismos de seguridad.

Estos principios contribuyen a crear sistemas más predecibles, auditables y resistentes a ataques, aspectos fundamentales en el desarrollo de software seguro (Guardati Buemo, 2007).

e) Mantenibilidad y adaptación a nuevas amenazas

Finalmente, la estructura orientada a objetos mejora la mantenibilidad del software, permitiendo adaptar el sistema ante nuevas amenazas o cambios en las políticas de seguridad sin necesidad de rediseñar completamente la aplicación (Blanco Fernández, 2020).

La figura 3 muestra en forma gráfica las ventajas de la POO.

Figura 3

Ventajas de la POO

Autor: Héctor Avalos Silva

2. ENTIDADES, CLASES Y OBJETOS

En POO, se parte con el análisis del problema, donde se identifican las entidades relevantes que interactúan dentro del sistema. Por ejemplo, en un sistema de control de accesos institucional, se inicia identificando y analizando las entidades (los elementos) que existen y cómo se relacionan entre sí.

Este enfoque resulta especialmente relevante en el ámbito de la ciberseguridad, donde una mala modelación puede derivar en fallos de control, accesos indebidos o vulnerabilidades lógicas.

Una entidad es un concepto del mundo real o del dominio del problema que tiene significado propio y que es relevante para el sistema que se está analizando. Una entidad representa algo que existe, ocurre o se gestiona dentro del contexto del sistema, y sobre lo cual se necesita almacenar información o realizar acciones.

La entidad pertenece al análisis del problema; la clase pertenece al diseño y la implementación del sistema.

El siguiente enlace lleva a un video que explica la forma cómo dado el contexto de un problema se puede identificar las correspondientes entidades:

- Título: Identificación de clases en problemas computacionales
- Descripción: Recurso introductorio que recomienda como reconocer y elegir a las clases en un problema a resolver.
- Enlace: https://www.youtube.com/watch?v=tkyYiflzZ4g

En la figura 4 se representa de forma gráfica una entidad identificada dentro de un problema.

Figura 4

Representación de una entidad dentro de un problema

Autor: Héctor Avalos Silva

En el ejemplo del sistema de control de accesos, se identifican las siguientes entidades, la figura 5 los muestra.

Figura 5

Representación conceptual de entidades del sistema de control de accesos

Autor: Héctor Avalos Silva

Estas entidades se modelan mediante clases, a partir de las cuales se crean objetos que representan instancias concretas del sistema (Oviedo Regino, 2015).

La figura 6, ilustra la relación entre una entidad y una clase

Figura 6

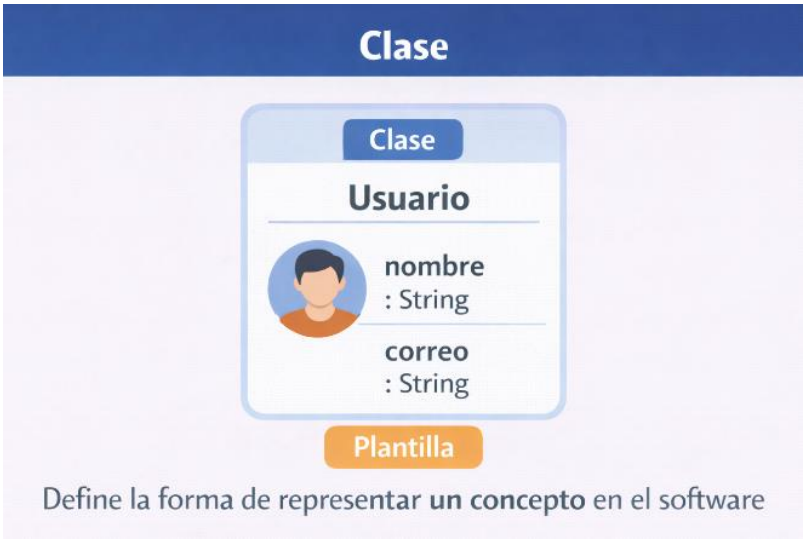
Relación entre Entidad y Clase

Autor: Héctor Avalos Silva

La clase es la representación técnica de una entidad dentro del software. La figura 7 ilustra la representación lógica de una clase.

Figura 7

Representación lógica de la clase Usuario



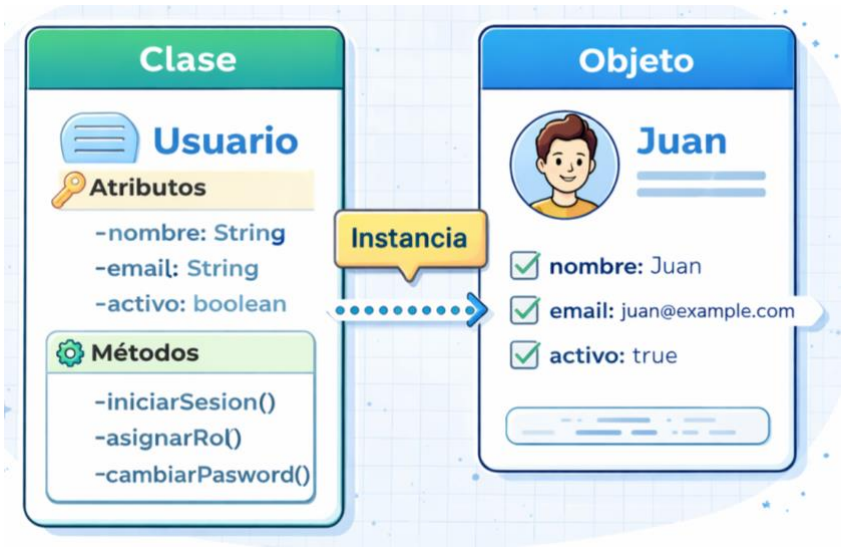
Autor: Héctor Avalos Silva

Una clase es una especie de plantilla o molde que se usa para crear objetos. Primero se define la clase y luego, a partir de ella, se crean los objetos.

En la Figura 8 se ilustra la relación entre una clase y un objeto.

Figura 8

Relación entre Clase y Objeto



Autor: Héctor Avalos Silva.

Relación entre entidad, clase y objeto

La clase define cómo debe ser algo y el objeto es ese algo en la práctica. Una clase describe qué información tendrá un elemento (atributos), qué acciones podrá realizar (métodos).

Por ejemplo, en un sistema de control de accesos, la clase Usuario, define que todo usuario tendrá, por ejemplo, nombre, email, rol, activo. Pero la clase no representa a una persona específica, solo la idea general de “usuario”.

Un objeto puede representar a una persona, un auto, un archivo, un recurso o cualquier elemento importante del problema a resolver (Blanco Fernández, 2020).

Un objeto es una instancia concreta de una clase, tiene valores reales, representa un caso específico. Por ejemplo, objetos de la clase Usuario pueden ser, el usuario llamado Ana, con email ana@gmail.com, rol Administrador, activo true y el usuario llamado Carlos, con email carlos@gmail.com, rol Estudiante, activo false.

La figura 9 representa la relación entre una entidad, una clase y un objeto.

Figura 9

Relación entre Entidad, Clase y Objeto

Autor: Héctor Avalos Silva

2.1 DIFERENCIA CONCEPTUAL ENTRE CLASE Y OBJETO

Una clase puede entenderse como un molde o plantilla que define las características y comportamientos comunes de un conjunto de entidades.

Una clase es la materialización de la entidad, es la representación técnica de una entidad en el software. Define:

- Atributos (datos).
- Métodos (comportamiento).
- Reglas internas (validaciones, restricciones).

Desde la ciberseguridad, una clase:

- Encapsula datos sensibles.
- Centraliza controles de acceso.
- Reduce exposición directa del estado interno.

Ejemplo en Java de la clase "Usuario"

```
//Definición de la clase Usuario
public class Usuario {
  //Atributos
```

```
    private String nombre;
    private String email;
    private String rol;
    private boolean activo;

   //Constructor
    public Usuario(String nombre, String email, String rol) {
        this.nombre = nombre;
        this.email = email;
        this.rol = rol;
        this.activo = true;
    }

   //Métodos
    public boolean esActivo() {
        return activo;
    }

    public String getRol() {
        return rol;
    }

    public void bloquear() {
        this.activo = false;
    }
} //Final de la clase
```

Observación de seguridad:

- Los atributos son privados (private), con este modificador de acceso se cumple con el pilar de encapsulación.
- El estado del usuario solo puede modificarse mediante métodos.

Un objeto, en cambio, es una instancia concreta de una clase, con valores específicos en sus atributos y creada en tiempo de ejecución (Blanco Fernández, 2020).

Por ejemplo, dentro de un sistema de control de accesos:

- La clase Usuario define atributos como nombre, email, rol y activo.

- Un objeto de tipo Usuario podría representar a *Ana Torres*, con el email institucional específico, el rol de administrador y activo.

Comprender esta diferencia es clave para evitar confundir el diseño del sistema con su ejecución concreta.

En términos de ciberseguridad:

- El objeto representa un actor real dentro del sistema.
- Posee un estado que puede ser explotado si no se controla.
- Es el verdadero objetivo de ataques.

Ejemplo de objetos:

Usuario admin = new **Usuario(**"Ana", "ana@gmail.com", "ADMIN"**);**

Usuario invitado = new **Usuario(**"Carlos","carlos@gmail.com", "INVITADO"**);**

Ambos son objetos de la misma clase Usuario, pero:

- admin tiene mayor impacto en la seguridad.
- invitado debe tener privilegios limitados.

Las clases definen reglas; los objetos ejecutan acciones y asumen riesgos.

2.2 REPRESENTACIÓN DE OBJETOS EN UN SISTEMA DE SOFTWARE

Cuando hablamos de representar objetos en un sistema de software, nos referimos a cómo los elementos reales del problema se convierten en elementos concretos dentro del programa.

En la POO, un objeto es la forma en que el sistema "materializa" algo de la realidad. Es decir, no basta con definir una clase; el sistema funciona cuando existen objetos creados a partir de esas clases (Blanco Fernández, 2020).

Cada objeto debe tener una responsabilidad clara y limitada, evitando concentrar múltiples funciones críticas en una sola clase.

Antes de programar, observamos la realidad. Por ejemplo, en un sistema de control de accesos institucional:

- En la realidad existen personas que usan el sistema
- En el software, esas personas se representan como objetos Usuario

Cada objeto representa un caso concreto, no una idea general.

Por ejemplo, en el mundo real existe Carlos Pérez, en el software es un objeto Usuario. Ana Torres existe en el mundo real y el software es otro objeto Usuario. Ambos son usuarios, pero no son el mismo objeto, aunque provengan de la misma clase.

Qué información tiene un objeto

Un objeto guarda información propia, llamada estado.

Por ejemplo, un objeto Usuario puede tener:

- nombre = "María López"

- email = mlopez@institucion.edu
- rol = "Administrador"
- activo = true

Esa información diferencia a un objeto de otro, incluso si ambos pertenecen a la misma clase (Oviedo Regino, 2015).

En Java, esta distinción se refleja claramente entre la definición de la clase y la creación de objetos mediante el operador new (Hernández Bejarano, 2023).

Ejemplo de representación de una clase y un objeto en Java:

```
// Definición de la clase Usuario
class Usuario {
   String nombre;
   String email
   String rol;
   boolean activo;
}
// Creación de un objeto “usuario1”
Usuario usuario1 = new Usuario();
//Uso del objeto usuario1
usuario1.nombre = "María López";
usuario1.email= "mlopez@institucion.edu";
usuario1.rol = "Administrador";
usuario1.activo = True;
```

Aquí ocurre lo siguiente:

- Usuario, es la clase
- usuario1, es el objeto
- new Usuario(), crea un objeto Usuario en memoria
- nombre, email, rol y activo son atributos

Este objeto existe mientras el programa se ejecuta y representa a un usuario real dentro del sistema (Hernández Bejarano, 2023).

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, 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

DEFINICION DE TERMINOS:

- **Programación orientada a objetos:** Es la aplicación práctica del paradigma mediante un lenguaje de programación. Implica escribir código usando clases, objetos, métodos y atributos. Por ejemplo: Implementar en Java las clases Usuario, Rol y Permiso con sus atributos y métodos.
- **Clase y objeto:** Una clase es el modelo o plantilla que define las características y comportamientos comunes de un conjunto de entidades, mientras que un objeto es una instancia concreta de esa clase, con valores específicos, que representa una entidad real del sistema y participa activamente en su funcionamiento.

RECURSOS DE PROFUNDIZACION:

Recurso 1:

- **Título:** Preguntas para el análisis orientado a objetos aplicado a sistemas seguros.
- **Descripción:** Recurso de preguntas con respuestas justificadas que permiten al estudiante consolidar los fundamentos del análisis y modelado orientado a objetos desde una perspectiva de ciberseguridad. A través del razonamiento conceptual sobre el paradigma orientado a objetos, sus ventajas en la estructuración de sistemas y la correcta diferenciación entre clases y objetos, el recurso enfatiza cómo un modelado adecuado contribuye a la prevención de vulnerabilidades en etapas tempranas del desarrollo de software.
- **Documento:** Recurso de profundización 1.docx

Recurso 2:

- **Título:** Ejemplo de identificación de entidades, modelado de clases y su representación en Java
- **Descripción:** Ayuda al estudiante a comprender cómo a partir del análisis del enunciado de un problema, se identifican entidades, modelan en clases y se implementan en un lenguaje de programación.
- **Documento:** Recurso de profundización 2.docx

ACTIVIDAD DE EVALUACIÓN:

PREGUNTAS DE AUTOEVALUACIÓN

INSTRUCCIONES:	Lea cuidadosamente cada pregunta y seleccione una sola opción. Luego de responder, revise la retroalimentación asociada a cada alternativa para reforzar su aprendizaje.					
Titulo:	Análisis y modelado orientado a objetos: Fundamentos, clases y objetos en la POO					
Dificultad:	**x**	Baja		Media		Alta
RDAs Evaluados:	**X**	RDA 1				
		RDA 2				
		RDA 3				
Reto:	**1**					
Secciones evaluadas:	**1**	1. Análisis y modelado orientado a objetos 1.1 Fundamentos de la Programación Orientada a Objetos 1.1.1 Paradigma orientado a objetos y su propósito 1.1.2 Ventajas de la POO en la estructuración de sistemas de software 1.2 Clases y objetos 1.2.1 Diferencia conceptual entre clase y objeto 1.2.2 Representación de objetos en un sistema de software.				
PREGUNTA 1	¿Cuál es el propósito principal de la Programación Orientada a Objetos?					

CORRECTA	Facilitar la organización, comprensión y mantenimiento de los programas.
Incorrecta 1	Hacer que los programas sean más largos.
Incorrecta 2	Eliminar el uso de clases en el software.
Incorrecta 3	Evitar el análisis previo de los problemas.
Retroalimentación de las respuestas	
Correcta	La POO permite crear programas más claros, organizados y fáciles de mantener gracias al uso de clases y objetos bien definidos (GERVAIS, 2025).
Incorrecta 1	El objetivo de la POO no es aumentar el tamaño del código, sino mejorar su estructura y calidad.
Incorrecta 2	Las clases son un elemento central de la POO.
Incorrecta 3	La POO promueve el análisis previo del problema para identificar correctamente las clases y sus responsabilidades (Oviedo Regino, 2015).
PREGUNTA 2	¿Cuál es el propósito principal de una clase en un sistema de software?
CORRECTA	Definir las características y acciones que tendrán los objetos.
Incorrecta 1	Ejecutar directamente las acciones del programa.
Incorrecta 2	Almacenar únicamente datos sin comportamiento.
Incorrecta 3	Representar un único elemento real del sistema.
Retroalimentación de las respuestas	
Correcta	Es correcta porque la clase define atributos y métodos que luego usarán los objetos (Guardati Buemo, 2016).
Incorrecta 1	Es incorrecta porque las acciones se ejecutan a través de objetos, no directamente desde la clase.
Incorrecta 2	Es incorrecta porque una clase incluye datos y acciones, no solo datos.
Incorrecta 3	Es incorrecta porque una clase puede generar muchos objetos, no solo uno.
PREGUNTA 3	¿Cuál es la diferencia entre una clase y un objeto en POO?
CORRECTA	Una clase es una plantilla o modelo, y un objeto es una instancia creada a partir de esa clase.
Incorrecta 1	Una clase y un objeto significan exactamente lo mismo.
Incorrecta 2	Un objeto es una plantilla y una clase es un ejemplo concreto.
Incorrecta 3	Una clase solo existe cuando el programa está en ejecución.
Retroalimentación de las respuestas	

CORRECTA	Una clase define cómo serán los objetos, y los objetos son ejemplos concretos creados a partir de ella (Blanco Fernández, 2020) (Oviedo Regino, 2015).
Incorrecta 1	Clase y objeto no son lo mismo; cumplen roles distintos dentro del programa.
Incorrecta 2	Invierte los conceptos; la clase es el modelo, no el objeto.
Incorrecta 3	La clase existe incluso antes de ejecutar el programa, como definición del sistema.
PREGUNTA 4	¿Cuál de las siguientes opciones representa mejor la representación de objetos en un sistema de software?
CORRECTA	La creación de instancias concretas que interactúan entre sí durante la ejecución del sistema.
Incorrecta 1	La lista de reglas de seguridad del sistema.
Incorrecta 2	El documento de requisitos del proyecto.
Incorrecta 3	La descripción teórica del paradigma orientado a objetos.
Retroalimentación de las respuestas	
CORRECTA	Los objetos existen y se comunican durante la ejecución del sistema, representando elementos reales del problema (Blanco Fernández, 2020) (Oviedo Regino, 2015).
Incorrecta 1	Las reglas no son objetos, sino restricciones del sistema.
Incorrecta 2	El documento de requisitos es parte del análisis, no de la representación en software.
Incorrecta 3	La teoría no representa directamente objetos dentro del sistema.

PUCE Virtual

síguenos en:

www.pucevirtual.puce.edu.ec