Descubriendo la Programación Orientada a Objetos en Java: De Problemas a Soluciones

Ingeniería | Ingeniería de sistemas | Aprendizaje Basado en Problemas

Descripción

Este plan de clase está diseñado para introducir a los estudiantes universitarios de Ingeniería de Sistemas en los conceptos fundamentales de la programación orientada a objetos (POO) utilizando Java. A través de la metodología de Aprendizaje Basado en Problemas (ABP), los estudiantes analizarán situaciones reales o simuladas para identificar clases, atributos y métodos, implementando soluciones en código Java. Esta experiencia práctica no solo fortalece su comprensión técnica, sino también desarrolla habilidades críticas como el análisis, diseño de software y explicación clara del funcionamiento del código.

La programación orientada a objetos es un paradigma esencial en el desarrollo de software moderno, y su dominio es vital para la formación de ingenieros capaces de diseñar sistemas eficientes y mantenibles. Este enfoque conecta directamente con escenarios cotidianos y profesionales, donde modelar entidades y comportamientos mediante clases y objetos facilita la solución de problemas complejos. Así, el aprendizaje se vuelve significativo y aplicable, preparando a los estudiantes para desafíos reales en el ámbito tecnológico.

Objetivos de Aprendizaje

- Identificar clases, atributos y métodos a partir de una situación problemática concreta.
- Implementar una clase básica en Java con atributos y métodos definidos.
- Crear objetos usando constructores y emplear métodos para manipular datos y resolver problemas.
- Explicar el funcionamiento básico del código desarrollado, demostrando comprensión de los conceptos de POO.

Recursos Necesarios

- Computadoras con entorno de desarrollo Java instalado (Eclipse, IntelliJ IDEA o NetBeans) - 1 por estudiante o pareja.
- Proyector para presentación inicial.
- Material impreso con la situación problemática y guía de actividades (1 por estudiante).
- Acceso a internet para consulta de documentación Java oficial.
- Pizarras o rotafolios para trabajo en grupos.

Requisitos Previos

- Conocimientos básicos de programación estructurada (variables, tipos de datos, estructuras de control).

- Familiaridad inicial con sintaxis básica de Java (declaración de variables, impresión en consola).
- Habilidades básicas en el uso de un entorno de desarrollo integrado (IDE) para Java.

Actividades

Fase de Inicio

Tiempo estimado: 20 minutos

Propósito de la sesión:

Docente: Explica que en esta sesión se abordará cómo a partir de un problema real se pueden identificar elementos que se modelan como clases, atributos y métodos en Java, y cómo se implementan para dar soluciones concretas.

Estudiantes: Se preparan para conectar conceptos previos de programación con nuevos elementos de POO.

Activación de conocimientos previos:

- **Docente:** Presenta la pregunta detonadora: "¿Cómo podríamos representar en un programa un vehículo, un estudiante o un banco? ¿Qué características y acciones tendría cada uno?"
- **Estudiantes:** En parejas, discuten brevemente y anotan ejemplos de características (atributos) y acciones (métodos) para un objeto real.

Motivación y enganche:

Docente: Muestra un breve video (2 minutos) o presenta un dato curioso sobre cómo la programación orientada a objetos ha revolucionado la industria del software y ejemplos de aplicaciones famosas (juegos, apps móviles) desarrolladas con Java.

Estudiantes: Observan y reflexionan sobre la importancia de la POO en la tecnología actual.

Contextualización:

Docente: Conecta el tema con la vida cotidiana de los estudiantes: "Cuando usamos apps, jugamos o gestionamos información, detrás hay objetos que interactúan; hoy aprenderemos a crear esos objetos en Java."

Estudiantes: Reconocen la aplicabilidad práctica y se motivan para la sesión.

Fase de Desarrollo

Tiempo estimado: 80 minutos

Presentación del contenido:

Docente: Introduce la situación problemática: "Una biblioteca quiere automatizar el registro de sus libros y préstamos. Cada libro tiene un título, autor, año de publicación y un método para mostrar su información."

Se propone que los estudiantes identifiquen las clases, atributos y métodos necesarios para modelar esta situación.

Actividad 1: Identificación y análisis del problema

- **Objetivo específico:** Identificar clases, atributos y métodos a partir de la situación problemática.
- **Instrucciones:**
  - **Docente:** Divide a los estudiantes en grupos de 3-4. Pide que analicen el problema y respondan: ¿Qué objetos existen? ¿Qué características tienen? ¿Qué acciones deben realizar?
  - **Estudiantes:** Discuten y listan posibles clases, atributos y métodos en un rotafolio o pizarra.
- **Organización:** Grupos de 3-4 estudiantes.
- **Producto:** Lista de clases, atributos y métodos para la biblioteca.
- **Tiempo:** 20 minutos.
- **Rol del docente:** Facilita la discusión, hace preguntas guía como: "¿Qué información debe guardar el objeto libro? ¿Qué métodos ayudarían a mostrar o actualizar esa información?"

Transición:

Docente: Resume las propuestas y conecta con la siguiente actividad: "Ahora que sabemos qué necesitamos, vamos a implementar la clase Libro en Java."

Actividad 2: Implementación de la clase en Java

- **Objetivo específico:** Implementar una clase en Java con atributos y métodos.
- **Instrucciones:**
  - **Docente:** Presenta un ejemplo base de código Java para la clase Libro en el proyector, explicando sintaxis básica y estructura.
  - **Estudiantes:** En sus computadoras, escriben la clase Libro con atributos (título, autor, año), constructor y un método para mostrar información.
  - **Docente:** Asiste individualmente, resolviendo dudas y asegurando que el código compile.
- **Organización:** Individual o parejas.
- **Producto:** Código fuente de la clase Libro implementada.
- **Tiempo:** 30 minutos.
- **Rol del docente:** Observa, corrige errores de sintaxis, fomenta buenas prácticas de codificación.

Transición:

Docente: Explica que el siguiente paso es crear objetos y usar métodos para resolver el problema.

Actividad 3: Creación de objetos y uso de métodos

- **Objetivo específico:** Crear objetos mediante constructores y utilizar métodos para manipular datos y resolver problemas.
- **Instrucciones:**

- **Docente:** Solicita a los estudiantes que en el mismo programa creen varios objetos Libro con diferentes datos y llamen al método para mostrar su información.
- **Estudiantes:** Implementan y prueban el código, verificando la salida en consola.
- **Docente:** Propone preguntas para explicar el código: "¿Qué hace el constructor? ¿Cómo se llaman los métodos y qué resultados producen?"

- **Organización:** Individual o parejas.
- **Producto:** Programa Java con creación de objetos y uso de métodos, con salida correcta.
- **Tiempo:** 30 minutos.
- **Rol del docente:** Facilita la reflexión, corrige errores lógicos, fomenta explicación oral del código.

Diferenciación:

- **Para estudiantes que terminan antes:** Se les asigna extender la clase con un método adicional (por ejemplo, actualizar año de publicación) y probarlo.
- **Para estudiantes que requieren apoyo:** El docente ofrece ejemplos más guiados, ayuda con la sintaxis y fomenta la colaboración en parejas.

Fase de Cierre

Tiempo estimado: 20 minutos

Síntesis:

Docente: Propone realizar un "ticket de salida" digital o en papel donde cada estudiante escribe tres ideas clave aprendidas sobre clases, atributos y métodos en Java.

Estudiantes: Completan el ticket individualmente y entregan al docente.

Reflexión metacognitiva:

- ¿Cómo identificaste las clases y atributos en la situación problemática planteada?
- ¿Qué función cumple un constructor en una clase Java?
- ¿Cómo explicarías a un compañero el propósito de un método en un objeto?

Retroalimentación:

Docente: Lee algunos tickets en voz alta, aclara dudas comunes y felicita los logros, destacando el progreso en comprensión y habilidades.

Transferencia:

Docente: Vincula el aprendizaje con futuras sesiones donde se abordarán conceptos avanzados como herencia y polimorfismo, y su aplicación en proyectos de ingeniería.

Tarea o reto:

Crear una clase adicional (por ejemplo, Usuario o Biblioteca) con al menos dos atributos y un método, implementarla en Java y preparar una breve explicación para la próxima clase.

Evaluación

Tipo de evaluación:

- **Diagnóstica:** En la fase de inicio, mediante la pregunta detonadora para conocer conocimientos previos.
- **Formativa:** Durante la fase de desarrollo, a través de la observación directa, resolución de problemas y producción de código.
- **Sumativa:** En la fase de cierre, mediante el ticket de salida y la explicación oral o escrita del código desarrollado.

Criterios de evaluación:

- Identificación correcta de clases, atributos y métodos en la situación problemática.
- Implementación funcional y sintácticamente correcta de una clase en Java.
- Creación adecuada de objetos mediante constructores y uso efectivo de métodos.
- Capacidad para explicar el código y su funcionamiento básico con claridad.

Instrumentos sugeridos:

- Lista de cotejo para evaluar identificación de elementos en el problema.
- Rúbrica para valoración del código Java implementado (estructura, sintaxis, funcionalidad).
- Observación directa y notas del docente durante actividades prácticas.
- Autoevaluación a través del ticket de salida y reflexión metacognitiva.

Evidencias de aprendizaje:

- Listados de clases, atributos y métodos generados en grupos.
- Código fuente de la clase Libro y programa de prueba con creación de objetos.
- Respuesta escrita en el ticket de salida y explicaciones orales o escritas sobre el código.

Enriquecimientos

Inicio - Contextualizar

Contextualización para la Fase de Inicio

Imagina que todos los días utilizas tu teléfono móvil o accedes a aplicaciones que te ayudan a organizar tu vida, desde redes sociales hasta plataformas de aprendizaje y videojuegos. Muchas de estas herramientas están diseñadas utilizando conceptos de programación orientada a objetos, una forma eficiente y organizada de construir software que facilita la creación, mantenimiento y escalabilidad de aplicaciones complejas.

Actualmente, el desarrollo de software orientado a objetos es fundamental en la industria tecnológica, y Java es uno de los lenguajes más utilizados en el mundo profesional, especialmente en sistemas empresariales, aplicaciones móviles y

grandes proyectos que requieren claridad y orden en su estructura.

Durante esta sesión, te invitamos a descubrir cómo, a partir de problemas cotidianos y situaciones reales, es posible modelar soluciones utilizando clases, atributos y métodos en Java. Este enfoque no solo te permitirá entender cómo organizar y estructurar tu código, sino también cómo pensar como un ingeniero de software que transforma ideas en soluciones funcionales.

Prepara tu mente para explorar, cuestionar y experimentar: en estas dos horas aprenderás a traducir problemas concretos en código orientado a objetos, un paso clave para tu formación profesional y para participar activamente en el desarrollo de software moderno.

Desarrollo - Ejemplos

Ejemplos Prácticos y Casos de Estudio para la Sesión

Para implementar la metodología de Aprendizaje Basado en Problemas (ABP) y cumplir con los objetivos de aprendizaje en 2 horas, se proponen los siguientes ejemplos y casos de estudio. Cada uno está diseñado para que los estudiantes identifiquen elementos de la programación orientada a objetos (POO) y desarrollen una solución en Java, reforzando la comprensión mediante la práctica y la explicación del código.

Ejemplo Práctico 1: Gestión de Estudiantes en una Universidad

Contexto problemático:

- La universidad necesita un sistema simple para almacenar información básica de los estudiantes, como nombre, número de matrícula, carrera y promedio.
- Se requiere que el sistema permita crear estudiantes, actualizar su promedio y mostrar su información.

Objetivos para el estudiante:

- Identificar la clase principal "Estudiante".
- Determinar atributos: nombre, matrícula, carrera, promedio.
- Definir métodos: constructor, actualizarPromedio(), mostrarInfo().
- Implementar la clase en Java, crear objetos y utilizar métodos.

Actividades ABP:

- Analizar el problema y discutir qué elementos representan clases y atributos.
- Escribir el diseño de la clase en pseudocódigo o diagramas simples.
- Codificar la clase en Java y crear al menos dos objetos con diferentes datos.
- Ejecutar métodos para modificar y mostrar la información.
- Explicar en grupo el funcionamiento y la relación entre atributos y métodos.

Ejemplo Práctico 2: Sistema Básico de Inventario para una Tienda

Contexto problemático:

- Una tienda desea mantener información sobre sus productos, incluyendo el nombre, código, precio y cantidad en stock.
- El sistema debe permitir añadir productos, actualizar stock y mostrar detalles.

Objetivos para el estudiante:

- Identificar la clase "Producto".
- Definir atributos y métodos necesarios.
- Implementar el constructor y métodos para modificar el stock y mostrar detalles.
- Crear objetos representando productos y manipularlos.

Actividades ABP:

- Analizar el problema para identificar qué datos se necesitan y cómo representan conceptos POO.
- Diseñar la clase y discutir posibles métodos útiles.
- Codificar la clase y probar con objetos reales.
- Resolver preguntas guía del tipo: ¿Qué ocurre si actualizamos el stock con un valor negativo?
- Presentar brevemente la solución y el código al grupo.

Caso de Estudio Integrador: Biblioteca Virtual

Contexto problemático:

- Se requiere diseñar un sistema para gestionar libros en una biblioteca virtual. Cada libro tiene título, autor, ISBN y número de copias disponibles.
- El sistema debe permitir crear libros, prestar libros (disminuir copias) y mostrar información.
- Se busca que los estudiantes trabajen en grupos para fomentar colaboración y discusión.

Objetivos para el estudiante:

- Identificar la clase "Libro" con atributos y métodos relevantes.
- Implementar constructor, método prestarLibro() y mostrarDetalles().
- Crear objetos y manejar la lógica de préstamo correctamente.
- Explicar el diseño y funcionamiento entre pares.

Actividades ABP:

- En grupos, analizar el problema y definir atributos y métodos.
- Diseñar la clase y discutir cómo manejar prestamo y disponibilidad.
- Programar la solución en Java.
- Probar diferentes escenarios (ej. préstamo cuando no hay copias disponibles).
- Compartir la solución con la clase y reflexionar sobre el uso de POO para resolver problemas.

Recomendaciones para el Docente

- Iniciar con una breve explicación teórica para contextualizar la POO.
- Presentar el problema antes de explicar conceptos para motivar la exploración.
- Guiar a los estudiantes con preguntas que los lleven a identificar clases, atributos y métodos.
- Facilitar el trabajo en grupos para compartir y discutir ideas.
- Reservar tiempo para la implementación práctica y posterior reflexión sobre el código.
- Fomentar que los estudiantes expliquen su código en sus propias palabras para afianzar el aprendizaje.