Práctico 4: Orientación a Objetos

Objetivo:

Comprender y aplicar conceptos de Programación Orientada a Objetos en Java, incluyendo el uso de this, constructores, sobrecarga de métodos, encapsulamiento, atributos y métodos estáticos, y modelado en UML, para mejorar la modularidad, reutilización y diseño del código.

Resultados de aprendizaje:

1. **Referencia y gestión del objeto actual:** Utilizar this para referenciar la instancia actual, desambiguar atributos y mejorar la claridad del código.
2. **Inicialización y sobrecarga de constructores:** Definir y aplicar constructores para inicializar objetos, implementando sobrecarga para permitir múltiples formas de instanciación.
3. **Definición y reutilización de métodos:** Aplicar la sobrecarga de métodos para definir múltiples versiones con distintos parámetros, mejorando la flexibilidad y modularidad del código.
4. **Representación y depuración de objetos:** Implementar el método `toString` para generar representaciones legibles de objetos, facilitando su visualización y depuración.
5. **Atributos y métodos estáticos:** Diferenciar entre variables de instancia y de clase, aplicando `static` para compartir valores y funciones entre objetos.
6. **Modelado de clases y relaciones en UML:** Representar clases, atributos y métodos en diagramas UML, identificando relaciones de asociación, composición y agregación para modelar estructuras del mundo real.
7. **Encapsulamiento y control de acceso:** Implementar modificadores de acceso (`public`, `private`, `protected`) para restringir la visibilidad de atributos y métodos, utilizando getters y setters para garantizar el acceso seguro a los datos.
8. **Relaciones y dependencias entre clases:** Aplicar relaciones de asociación, composición y agregación en Java, diseñando interacciones adecuadas entre objetos y reflejando estructuras del mundo real.
9. **Principio "Tell, Don't Ask":** Identificar dependencias entre clases y aplicar el principio "Tell, Don't Ask" para delegar responsabilidades correctamente, reduciendo el acoplamiento y mejorando el diseño del software.

¿Qué es una Kata y cómo se utiliza en programación?

Una kata es un ejercicio de programación diseñado para mejorar habilidades de codificación mediante la repetición y el aprendizaje progresivo. El término proviene de las artes marciales, donde las katas son secuencias de movimientos que se practican repetidamente para perfeccionar la técnica.

En programación, las katas ayudan a los programadores a reforzar conceptos, mejorar la comprensión del código y desarrollar buenas prácticas. **Se recomienda resolver una kata varias veces, intentando mejorar el código en cada iteración, utilizando mejores estructuras, nombres más claros y principios de diseño.**

Importante:

Intentar resolver cada kata sin mirar la solución.

Comprobar la solución y corregir errores si es necesario.

Repetir 2 o 3 veces para mejorar la comprensión, lógica y el código.

Experimentar con diferentes valores para reforzar el aprendizaje.

Resolver el Caso: Soporte Técnico de Usuarios

El soporte técnico es un proceso estructurado para gestionar y resolver incidencias reportadas por los usuarios de un sistema.

Flujo de Trabajo:

Proceso de Soporte Técnico vinculado a las Katas de Programación en Java

El soporte técnico es un proceso estructurado para gestionar y resolver incidencias reportadas por los usuarios de un sistema. La implementación de este proceso en Java sigue una serie de etapas, reflejadas en las tres Katas progresivas que hemos diseñado. A continuación, explicamos el flujo del soporte técnico y cómo cada Kata contribuye a su desarrollo.

📌 Fases del Proceso de Soporte Técnico

1. **Registro del Ticket (Kata 1.1 - Nivel Básico)**
    - Un usuario reporta un problema generando un ticket de soporte.
    - El ticket recibe un ID único y se almacena con una descripción del problema.
    - Se asigna un estado inicial de "Abierto" y se registra la fecha de creación.
    - Un técnico o administrador puede consultar los tickets y cerrarlos cuando estén resueltos.
2. **Asignación de un Usuario y Gestión del Ticket (Kata 1.2 - Nivel Medio)**
    - Se introduce la entidad `Usuario`, quien es el responsable de generar un ticket.
    - Un ticket puede estar asociado a un usuario específico, lo que permite rastrear quién reportó el problema.
    - Se agrega la posibilidad de asignar un técnico al ticket mediante el método `asignarTecnico(String tecnico)`.
    - Se permite la sobrecarga de constructores en `TicketSoporte`, permitiendo la creación de tickets con o sin un usuario.
3. **Gestión Avanzada y Seguimiento del Ticket (Kata 1.3 - Nivel Avanzado)**
    - Se crea una entidad `Tecnico`, la cual representa al personal de soporte técnico.
    - Se introduce `SistemaSoporte`, una clase que gestiona múltiples tickets en una lista.
    - Se agregan funciones para:
        - Crear nuevos tickets.
        - Asignar técnicos a tickets.
        - Listar tickets abiertos y cerrados para su seguimiento.
    - Se mejora la visualización de objetos con el método `toString()`, facilitando la depuración.

🔄 Flujo de Trabajo para Proceso de Soporte Técnico

1. **Un usuario detecta un problema y reporta un ticket.**

- Se registra en TicketSoporte con estado "Abierto".
- Se le asigna un ID y se guarda su fecha de creación.

2. **El ticket se asigna a un técnico de soporte.**
   - Un técnico recibe el ticket según su especialidad.
   - Se registra en TicketSoporte con el nombre del técnico asignado.
3. **El técnico analiza el problema y brinda una solución.**
   - Puede actualizar el estado del ticket a "En proceso" mientras trabaja en la solución.
   - Se pueden adjuntar comentarios o actualizar la descripción del problema.
4. **El ticket se cierra cuando la incidencia es resuelta.**
   - El técnico cambia el estado a "Cerrado".
   - Se almacena la información para futuras referencias y auditoría.
5. **El sistema genera un reporte de los tickets gestionados.**
   - SistemaSoporte puede listar todos los tickets, mostrando:
     - Cantidad de tickets abiertos y cerrados.
     - Qué usuario generó cada ticket.
     - Qué técnico resolvió cada ticket.

Resolver Katas para caso de soporte técnico

A continuación, te presento 3 enunciados de katas en Java para fortalecer los conceptos vistos en el módulo 4.

Aquí tienes tres katas de complejidad progresiva para el caso "Seguimiento de Soporte Técnico" en Java, enfocadas en los conceptos de POO mencionados.

Kata 1.1: Seguimiento de Soporte Técnico (Nivel Básico)

Enunciado

Debes modelar la clase `TicketSoporte` que representará una solicitud de soporte técnico. Esta clase debe incluir atributos esenciales y métodos básicos para gestionar el ticket.

Atributos:

- **`id`: Identificador único del ticket (entero).**

- `descripcion`: Breve descripción del problema (cadena de texto).
- `estado`: Estado del ticket (abierto, en proceso, cerrado).
- `fechaCreacion`: Fecha de creación del ticket.

Métodos:

- `constructor`: Inicializa un ticket con un ID, descripción y estado "abierto".
- `cerrarTicket()`: Cambia el estado del ticket a "cerrado".
- `mostrarDetalle()`: Devuelve un resumen del ticket en formato legible.

🎯 Tarea a realizar

1. Implementa la clase `TicketSoporte` en Java, aplicando encapsulamiento y métodos adecuados.
2. Crea un pequeño programa que instancie un ticket de soporte y lo muestre en pantalla.
3. Cierra el ticket y muestra nuevamente su estado.

Kata 1.2: Seguimiento de Soporte Técnico (Nivel Medio)

Enunciado

Ahora, extenderás el modelo del seguimiento de soporte técnico con una clase `Usuario`, que representa a la persona que genera los tickets.

Atributos de `Usuario`:

- `id`: Identificador único del usuario.
- `nombre`: Nombre completo del usuario.
- `email`: Correo electrónico del usuario.

Nuevas características en `TicketSoporte`:

- Agregar una relación con `Usuario`, indicando quién creó el ticket.
- Implementar sobrecarga de constructores para permitir la creación de tickets con y sin usuario.
- Añadir un método `asignarTecnico(String tecnico)`, que asigne un técnico al ticket.

🎯 Tarea a realizar

1. Implementa la clase `Usuario` con los atributos y métodos adecuados.
2. Modifica `TicketSoporte` para incluir una relación con `Usuario`.

3. Implementa sobrecarga de constructores para los tickets.
4. Crea un programa que cree un usuario, asocie tickets a este usuario y asigne un técnico a un ticket.

Kata 1.3: Seguimiento de Soporte Técnico (Nivel Avanzado)

Enunciado

En este nivel, estructurarás mejor la gestión de tickets usando clases adicionales y aplicando relaciones entre objetos.

Nuevas clases:

- `Tecnico`: Representa un técnico de soporte, con atributos `id`, `nombre` y `especialidad`.
- `SistemaSoporte`: Gestiona los tickets en una lista y permite operar sobre ellos.

Nuevas funcionalidades:

- La clase `SistemaSoporte` debe permitir:
  - Crear un nuevo ticket.
  - Asignar un técnico a un ticket.
  - Listar todos los tickets abiertos y cerrados.
- Implementar `toString()` en las clases para mejorar la visualización.
- Aplicar encapsulamiento y métodos estáticos para identificar la cantidad de tickets creados.

🎯 Tarea a realizar

1. Implementa las clases `Tecnico` y `SistemaSoporte`.
2. Agrega métodos para gestionar los tickets en `SistemaSoporte`.
3. Implementa el método `toString()` en todas las clases.
4. Crea un programa que agregue varios tickets, asigne técnicos y muestre el estado general del sistema.

Estos ejercicios guían progresivamente desde la creación de una sola clase con encapsulamiento hasta la implementación de relaciones entre clases y gestión de colecciones, aplicando principios POO en Java. 🚀