


# JAVA Programación Orientada a Objetos

## ACTIVIDADES PRACTICAS JAVA TIME API

Te invitamos a completar este conjunto de actividades a tu propio ritmo. Cada una te ayudará a desarrollar un dominio más preciso de las fechas. **Recuerda que no es necesario resolver todas estas actividades en un solo día.** Te sugerimos que las abordes en tus momentos libres, entre encuentros o cuando te resulte más conveniente. ¡Toma el control de tu aprendizaje y avanza a tu ritmo!

---

### Ejercicios Fundamentales

 Estos ejercicios son de tipo **fundamental**, esto quiere decir que es lo mínimo que necesitas resolver para asegurar la comprensión del tema.

1. Crea un programa que genere un objeto `LocalDate` representando la fecha actual.
2. Desarrolla un programa que genere un objeto `LocalDate` representando una fecha específica, como tu cumpleaños.
3. Implementa un programa que solicite al usuario ingresar por separado el día, el mes y el año, para luego convertir esta información en un objeto `LocalDate`.
4. Escribe un programa que permita al usuario ingresar la fecha en un formato predeterminado por ti a través de la consola, y luego utilice esta información para crear un objeto `LocalDate`.
5. Desarrolla un programa que solicite al usuario ingresar una fecha en formato 'dd-MM-yyyy'. Luego, el programa debe generar un objeto `LocalDate` basado en la información proporcionada. Posteriormente, añade 15 días, 2 meses y 3 años a esta fecha. Por último, muestra al usuario la nueva fecha obtenida y el día de la semana correspondiente.
6. Desarrolla un programa que solicite al usuario ingresar una fecha en formato 'dd-MM-yyyy'. Luego, el programa debe generar un objeto `LocalDate` basado en la información proporcionada. Posteriormente, resta 13 días, 10 meses y 1 año. a esta fecha. Por último, muestra al usuario la nueva fecha obtenida y el día de la semana correspondiente.

7. Dado un objeto `LocalDate` ingresado por el usuario, tu tarea es verificar si el año correspondiente es un año bisiesto o no.
8. Diseña un juego de adivinanza de fechas, donde crearás una `LocalDate` aleatoria y le preguntarás al usuario cuál es. Solo podrás decirle si la fecha ingresada es antes o después de la fecha objetivo. Cuando adivine la fecha, el juego terminará y lo felicitarás por haberlo logrado
9. Crea un objeto `LocalTime` representando la hora actual y obtén la cantidad de segundos que han pasado desde el inicio del día.
10. Crea un objeto `LocalTime` que represente una hora específica. Después, utiliza ese objeto para calcular cuántas horas, minutos y segundos faltan hasta la medianoche.
11. Dado un objeto `LocalTime` ingresado por el usuario, ajusta la hora al próximo valor exacto.
12. Dada una cantidad de segundos ingresada por el usuario, conviértela en un objeto `LocalTime` y muéstrala por consola.
13. Escribe un programa que calcule la diferencia en segundos entre dos objetos `LocalTime`.
14. Dado un objeto `LocalTime` que has ingresado, crea un nuevo `LocalTime` que represente exactamente la mitad del tiempo transcurrido entre la hora dada y la medianoche.
15. Crea un método que, dado un `LocalDateTime`, devuelva un `ZonedDateTime` que represente el mismo instante en la zona horaria de Tokio (Asia/Tokyo).
16. Solicita al usuario una fecha y hora y crea un `ZonedDateTime` para luego mostrarle que fecha y hora sería en todas las zonas horarias.
17. Crea un método que reciba por parámetro la fecha de nacimiento de una persona como `LocalDate` y que calcule su edad en años.
18. Escribe un programa que tome dos fechas en formato de texto, las convierta a `LocalDate` y use `ChronoUnit` para calcular la cantidad de años, meses y días entre las dos fechas.

## Ejercicios Complementarios

🌟 Estos ejercicios son de tipo complementario. Esto quiere decir que te ayudarán a avanzar en profundidad en el tema visto, pero no son obligatorios.

### ACTIVIDAD 1: Prediciendo Eventos

Crea un método en Java que calcule las próximas fechas de varios eventos que ocurren en ciclos regulares. Este método debe recibir como parámetros la fecha del último evento y la cantidad de años del ciclo, y debe devolver un array de `LocalDate` con las próximas fechas en las que ocurrirá el evento. Los eventos que debes calcular son:

- **Invasión de cigarras:** Estos insectos tienen ciclos de vida de 13 o 17 años. El último brote importante emergió en el este de los Estados Unidos en 2021. Predice cuándo ocurrirán los próximos tomando un ciclo de 17 años.
- **Cometa Halley:** Este cometa pasa cerca de la Tierra aproximadamente cada 76 años. Fue visible por última vez en 1986. Calcula cuándo serán los próximos años que nos visitará.
- **Tránsito de Venus:** Este fenómeno ocurre en un patrón que se repite cada 243 años, con un par de tránsitos separados por 8 años entre sí e intervalos de 121.5 y 105.5 años, es decir, el ciclo sería: 105,5 años, 8 años, 121,5 años, 8 años, 105,5 años, 8 años, 121,5 años ... Los tránsitos más recientes fueron el 8 de junio de 2004 y el 5 de junio de 2012. Predice cuándo ocurrirán los próximos.

## ACTIVIDAD 2 : Seguimiento de Tiempo

Desarrolla una aplicación que permita a los usuarios registrar y gestionar sus tareas. Cada tarea tiene un tiempo de inicio (se registra automáticamente cuando se crea la tarea) vencimiento (ingresada por el usuario como `LocalDate`), finalización, una descripción y un id.

- **Crear tareas:** Al crear una tarea, los usuarios deben ingresar una descripción de la tarea e ingresar su fecha de vencimiento. La aplicación deberá registrar la fecha y hora actuales como tiempo de inicio de la tarea utilizando `LocalDateTime`. El id lo puedes obtener creando un atributo static que funcione como contador y cada vez que se crea una instancia de una tarea aumente en 1 y le asigne el valor a el atributo id de la instancia.
- **Finalizar tareas:** Los usuarios deben tener la opción de marcar una tarea como finalizada. Al hacerlo, la aplicación debe registrar la fecha y hora actuales como tiempo de finalización de la tarea.
- **Ver tareas:** Los usuarios deben poder ver una lista de todas las tareas, tanto completadas como pendientes. Para cada tarea completada, la aplicación debe mostrar la descripción, la fecha de inicio y fin, y la duración de la tarea . Para las tareas pendientes, la aplicación debe mostrar la descripción y la fecha de inicio, y cuántos días quedan para el vencimiento.
- **Eliminar tareas:** Los usuarios deben poder eliminar una tarea ingresando su id
- **Editar tareas:** Los usuarios deben poder editar la descripción de una tarea.
- **Tiempo total de trabajo:** La aplicación debe proporcionar un resumen que muestre el tiempo total de trabajo, sumando las duraciones de todas las tareas completadas.

- **Buscar tareas:** Los usuarios deben poder buscar tareas por descripción. La aplicación debe mostrar todas las tareas que coincidan con la descripción ingresada.
- **Ordenar tareas:** Los usuarios deben poder ordenar tareas alfabéticamente, por fecha de inicio, por id, por fecha de vencimiento, y por fecha de finalización (ten cuidado con la fecha de finalización que puede ser nula para tareas que no han finalizado).

### ACTIVIDAD 3 : Gestor de Programación de Televisión

Desarrolla una aplicación que permita a los administradores de un canal de televisión crear y gestionar una grilla de programas. Cada programa tiene una hora de inicio y fin en UTC, un nombre y una descripción.

- **Crear programas:** Al crear un programa, los administradores deben ingresar la hora de inicio en UTC, la duración en minutos, un nombre y una descripción. La aplicación debe calcular la hora de fin según la duración ingresada. Y debe validar que no se superpongan los horarios con estos programas, en tal caso debe pedirle de vuelta información con la sugerencia de horarios e intervalos disponibles.
- **Ver programas:** Los administradores deben poder ver una lista de todos los programas ordenados por hora de inicio. Para cada programa, la aplicación debe mostrar el nombre, la descripción, la hora de inicio y fin en tiempo UTC, y la duración del programa (calculada con `Duration.between()`).
- **Hora de transmisión del programa en diferentes zonas horarias:** Los administradores deben poder seleccionar un programa y ver la hora de transmisión del programa en diferentes zonas horarias. La aplicación debe mostrar la hora de inicio y fin del programa en todas las zonas horarias de los países de Sudamérica, en la Costa Este (Nueva York) y Oeste (Los Ángeles) de Estados Unidos, en España, Alemania, Sudáfrica, Israel, Hong Kong, Japón, Australia y Nueva Zelanda.