Resumen de Métodos de la API Date-Time de Java

LocalDate

- **of(int year, int month, int dayOfMonth)**: Crea una fecha específica.
  ```
  LocalDate date = LocalDate.of(2018, 1, 1);
  ```
- **withMonth(int month)**: Cambia el mes de la fecha.
  ```
  LocalDate newDate = date.withMonth(6);
  ```
- **plusDays(long days)**: Agrega días a la fecha.
  ```
  LocalDate newDate = date.plusDays(100);
  ```
- **minusDays(long days)**: Resta días de la fecha.
  ```
  LocalDate newDate = date.minusDays(50);
  ```
- **isLeapYear()**: Verifica si el año es bisiesto.
  ```
  boolean leap = date.isLeapYear();
  ```
- **getDayOfMonth()**: Obtiene el día del mes.
  ```
  int day = date.getDayOfMonth();
  ```

LocalTime

- **parse(CharSequence text)**: Convierte un texto a LocalTime.
  ```
  LocalTime time = LocalTime.parse("20:30:40");
  ```
- **with(TemporalField, long)**: Modifica un campo específico (ej. minutos).
  ```
  LocalTime newTime = time.with(ChronoField.MINUTE_OF_DAY, 10);
  ```
- **isAfter(LocalTime other)**: Verifica si es posterior a otro tiempo.
  ```
  boolean isAfter = newTime.isAfter(time);
  ```

LocalDateTime

- **atTime(LocalTime time)**: Combina fecha y hora en LocalDateTime.
  ```
  LocalDateTime dateTime = date.atTime(time);
  ```
- **plusMonths(long months)**: Suma meses a la fecha-hora.
  ```
  LocalDateTime newDateTime = dateTime.plusMonths(2);
  ```

Instant

- **now()**: Obtiene el instante actual en UTC.
  ```
  Instant now = Instant.now();
  ```
- **ofEpochMilli(long)**: Crea un instante desde milisegundos desde la época.
  ```
  Instant instant = Instant.ofEpochMilli(1500000000L);
  ```
- **isBefore(Instant other)**: Verifica si es anterior a otro instante.
  ```
  boolean isBefore = instant.isBefore(now);
  ```

Period

- **of(int years, int months, int days)**: Crea un período específico.
  ```
  Period period = Period.of(1, 2, 3);
  ```
- **minusMonths(long months)**: Resta meses del período.
  ```
  Period newPeriod = period.minusMonths(1);
  ```
- **plusDays(long days)**: Agrega días al período.
  ```
  Period newPeriod = period.plusDays(5);
  ```

Duration

- **parse(CharSequence text)**: Parsea una duración en formato ISO-8601.
  ```
  Duration duration = Duration.parse("PT2H30M");
  ```
- **between(Temporal start, Temporal end)**: Calcula la duración entre dos puntos.
  ```
  Duration duration = Duration.between(time1, time2);
  ```
- **equals(Duration other)**: Compara dos duraciones.
  ```
  boolean isEqual = duration.equals(anotherDuration);
  ```

ZoneId

- **of(String zoneId)**: Obtiene una zona horaria por su ID.
  ```
  ZoneId zone = ZoneId.of("America/Chicago");
  ```
- **getAvailableZoneIds()**: Obtiene todas las zonas horarias disponibles.
  ```
  Set<String> zones = ZoneId.getAvailableZoneIds();
  ```

ZoneOffset

- **of(String offsetId)**: Obtiene un desplazamiento horario por su ID.
  ```
  ZoneOffset offset = ZoneOffset.of("+06:00");
  ```
- **ofHours(int hours)**: Crea un ZoneOffset a partir de horas de diferencia.
  ```
  ZoneOffset offset = ZoneOffset.ofHours(6);
  ```

OffsetTime

- **of(LocalTime time, ZoneOffset offset)**: Combina hora local con un desplazamiento.
  ```
  OffsetTime offsetTime = OffsetTime.of(localTime, offset);
  ```
- **plusHours(long hours)**: Suma horas al tiempo con desplazamiento.
  ```
  OffsetTime newOffsetTime = offsetTime.plusHours(3);
  ```
- **withOffsetSameInstant(ZoneOffset offset)**: Ajusta el desplazamiento manteniendo el instante.
  ```
  OffsetTime adjustedTime = newOffsetTime.withOffsetSameInstant(offset);
  ```

OffsetDateTime

- **of(LocalDateTime dateTime, ZoneOffset offset)**: Combina fecha-hora local con un desplazamiento.
  ```
  OffsetDateTime offsetDateTime = OffsetDateTime.of(dateTime, offset);
  ```
- **isBefore(OffsetDateTime other)**: Verifica si es anterior a otro OffsetDateTime.
  ```
  boolean isBefore = offsetDateTime.isBefore(otherOffsetDateTime);
  ```

TemporalAdjusters

- **firstDayOfMonth()**: Ajusta al primer día del mes.
  ```
  LocalDate firstDay = date.with(TemporalAdjusters.firstDayOfMonth());
  ```
- **lastDayOfMonth()**: Ajusta al último día del mes.
  ```
  LocalDate lastDay = date.with(TemporalAdjusters.lastDayOfMonth());
  ```
- **firstDayOfNextMonth()**: Ajusta al primer día del próximo mes.
  ```
  LocalDate firstDayNextMonth =
  date.with(TemporalAdjusters.firstDayOfNextMonth());
  ```
- **next(DayOfWeek day)**: Ajusta al próximo día de la semana dado.
  ```
  LocalDate nextMonday = date.with(TemporalAdjusters.next(DayOfWeek.MONDAY));
  ```
- **previous(DayOfWeek day)**: Ajusta al día de la semana anterior.
  ```
  LocalDate lastFriday =
  date.with(TemporalAdjusters.previous(DayOfWeek.FRIDAY));
  ```

DateTimeFormatter

- **ofPattern(String pattern)**: Define un patrón de formato.
  ```
  DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
  ```
- **ISO_LOCAL_DATE**: Formato predefinido para fechas (yyyy-MM-dd).
  ```
  String formattedDate = date.format(DateTimeFormatter.ISO_LOCAL_DATE);
  ```

Chrono Units

- **ChronoUnit.DAYS**: Representa días para cálculos.
  ```
  long daysBetween = ChronoUnit.DAYS.between(date1, date2);
  ```
- **ChronoUnit.WEEKS**: Representa semanas para cálculos.
  ```
  long weeksBetween = ChronoUnit.WEEKS.between(date1, date2);
  ```
- **ChronoUnit.MONTHS**: Representa meses para cálculos.
  ```
  LocalDate newDate = date.plus(2, ChronoUnit.MONTHS);
  ```
- **ChronoUnit.YEARS**: Representa años para cálculos.
  ```
  LocalDate newDate = date.plus(1, ChronoUnit.YEARS);
  ```

ChronoField

- **ChronoField.DAY_OF_MONTH**: Campo para el día del mes.
  ```
  int dayOfMonth = date.get(ChronoField.DAY_OF_MONTH);
  ```
- **ChronoField.MONTH_OF_YEAR**: Campo para el mes del año.
  ```
  LocalDate updatedDate = date.with(ChronoField.MONTH_OF_YEAR, 3);
  ```
- **ChronoField.YEAR**: Campo para el año.
  ```
  int year = date.get(ChronoField.YEAR);
  ```

Clock

- **Clock.systemDefaultZone()**: Obtiene el reloj de la zona horaria del sistema.
  ```
  Clock clock = Clock.systemDefaultZone();
  ```
- **Clock.systemUTC()**: Obtiene el reloj en la zona UTC.
  ```
  Clock utcClock = Clock.systemUTC();
  ```