**Java Date & Time API**

## **Pendahuluan**

**Sebelumnya harus sudah paham materi terkait** :

* Java Dasar
* Java Object Oriented Programming
* Java Generic
* Java Collection
* Java Lambda
* Apache Maven
* Java Unit Test
* Java Stream
* Java Internationalization

**Agenda**

* Pengenalan Date & Time API
* Date, Calendar dan TimeZone
* LocalDate, LocalTime dan LocalDateTime
* Zoneld, ZoneOffset dan ZonedDateTime
* Instant dan Clock
* Duration dan Period
* Temporal
* Dan lain-lain

## **Pengenalan Date & Time API**

**Sebelumnya Date dan Time API**

* Sejak awal, di Java representasi tipe data tanggal dan waktu adalah java.util.Date dan java.util.Calendar
* Java Date dan Time API merupakan fitur baru sejak java versi 8

**Masalah Dengan Class Date dan Calendar**

* Thread Safety, class Date dan Calendar tidak thread safe, dalam artian berbahaya jika diakses secara parallel(beberapa proses mengakses object yang sama).
* Desain class Date dan Calendar tidak terlalu bagus di desain dari awal, hal ini menyulitkan ketika kita butuh melakukan operasi yang melibatkan tanggal dan waktu, seperti mencari durasi waktu, periode, menambah waktu, dan operasi lainnya
* Desain tanggal dan waktu dengan timezone agak menyulitkan di class Date dan Calendar

**Package java.Time**

* Date & Time API yang baru sekarang berada dalam satu package, yaitu package java.time
* Sekarang ada banyak sekali class-class yang terdapat di package tersebut, dan kegunaannya berbeda-beda, yang akan kita bahas secara detail chapter-chapter selanjutnya
* <https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/time/package-summary.html>

**Perbedaan dengan Date dan Time API Baru**

* Object di Date dan Time API baru bersifat immutable dan thread safe, artinya tidak bisa diubah, jika diubah, itu akan membuat object yang baru sehingga aman digunakan proses parallel
* Terdapat pemisah antara Tanggal dan Waktu, tidak digabung seperti pada class Date
* Mendukung data lain seperti durasi, periode dan lain – lain

## **Date**

* Sebelum membahas Date dan Time API yang baru, kita akan sekilas membahas class Date
* Date adalah representasi tanggal dan juga waktu di Java sejak di versi awal
* Karena tidak ada pemisahan antara tanggal dan waktu di class Date, ini memang agak menyulitkan jika kita hanya butuh misal tanggal saja, dan waktu saja
* <https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/util/Date.html>



**Millisecond**

* Data menggunakan millisecond setelah UNIX epoch (January 1, 1970 00:00:00 UTC)
* Untuk mendapatkan millisecond saat ini, di java kitab isa menggunakan System.currentTimeMilis()
* <https://currentmilis.com/>

## **Calendar**

* Class Calendar adalah class yang digunakan sebagai pembantu class Date
* Pada class Date hampir semua method untuk melakukan manipulasi tanggal dan waktu sudah ditandai sebagai @Deprecated, artinya tidak direkomendasikan digunakan lagi
* Untuk itu, kita butuh menggunakan class Calendar untuk manipulasi tanggal dan waktu
* Class Calendar tidak memiliki public constructor, sehingga untuk membuat object Calendar, kita akan menggunakan static method milik calendar bernama getInstance()

**Manipulasi Tanggal dan Waktu**

* Salah satu fitur yang terdapat di Calendar adalah, kitab isa melakukan menipulasi tanggal dan waktu di object Calendar menggunakan method set(type, value)
* Dan untuk mengambil value tanggal atau waktu kita bisa menggunakan method get(type)

## **TimeZone**

* Class TimeZone merupakan representasi dari data time zone
* Secara default, jika kita membuat object Calendar dan tidak menggunakan TimeZone, secara otomatis objectnya akan menggunakan default TimeZone, yaitu timezone sistem operasi yang kita gunakan
* Untuk mengetahui default time zone, kitab isa gunakan method TimeZone.getDefault()
* Sedangkan jika ingin membuat object TImeZone, kitab isa gunakan method TimeZone.getTimeZone(“Zone ID”)
* Untuk mengetahui semua zone id yang didukung oleh java, kita bisa gunakan TimeZone.getAvailableIDs()

**TimeZone di Date**

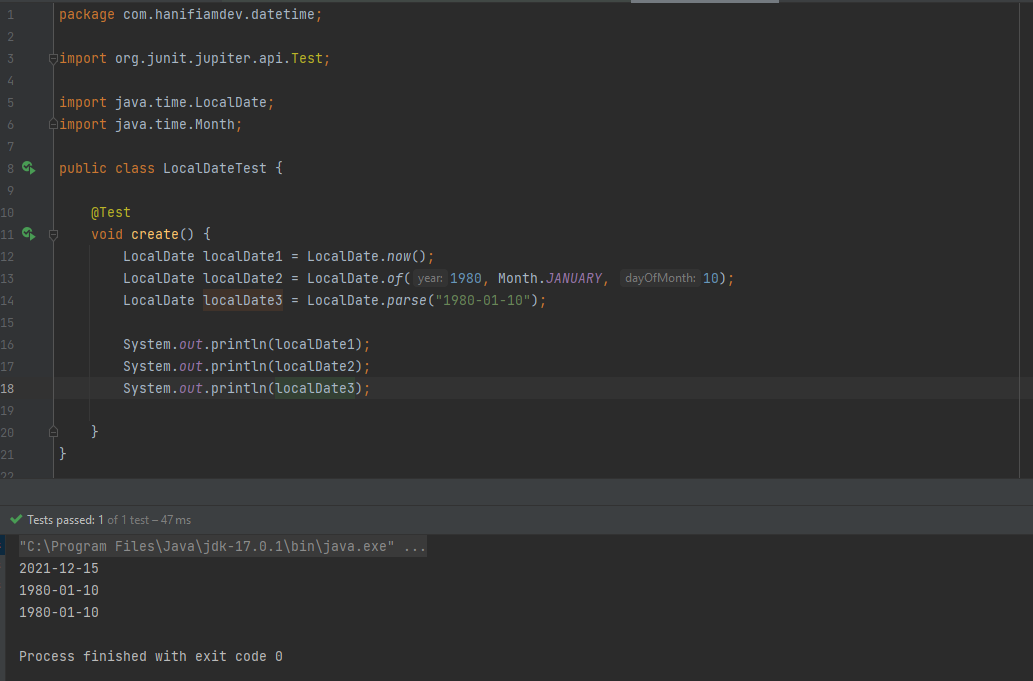
* Class Date tidak memiliki method apapun untuk mendapatkan time zone atau mengubah time zone
* Secarfa default, saat kita membuat object Date, Object tersebut akan menggunakan time zone default

**TimeZone di Calendar**

* Berbeda dengan Date, di Calendar, Informasi TimeZone yang bis akita ubah
* Untuk mengubah time zone di Calendar, kita bisa menggunakan method setTimeZona()

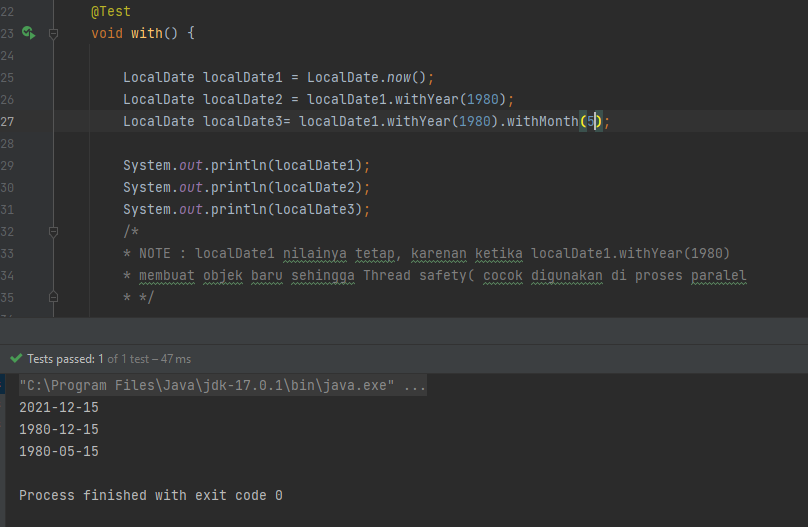
## **LocaleDate**

* LocalDate adalah class di Date & Time API baru
* LocalDate merupakan representasi untuk tipe data tanggal(tanpa waktu)
* Default format waktu untuk LocalDate dalah yyyy-MM-dd



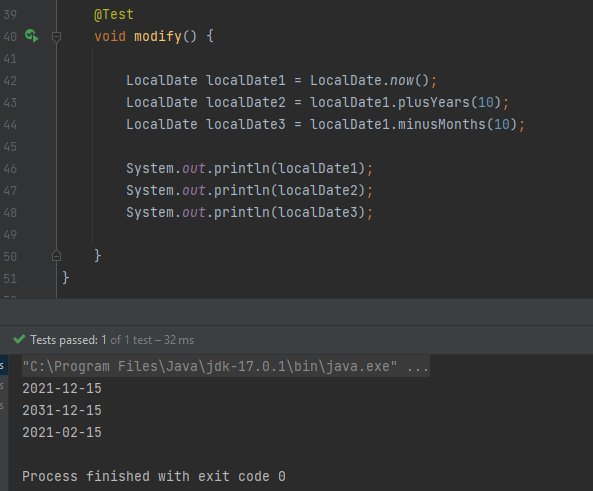
**Mengubah LocalDate**

* Object LocalDate juga bis akita ubah tanggal nya jika kita mau
* Untuk mengubah tanggal, kita bisa menggunakan method with, seperti withYear, withMonth, dan lain – lain
* Perlu diingat, mengubah LocalDate akan menciptakan object LocalDate baru, artinya object aslinya tidak akan berubah, karena bersifat immutable



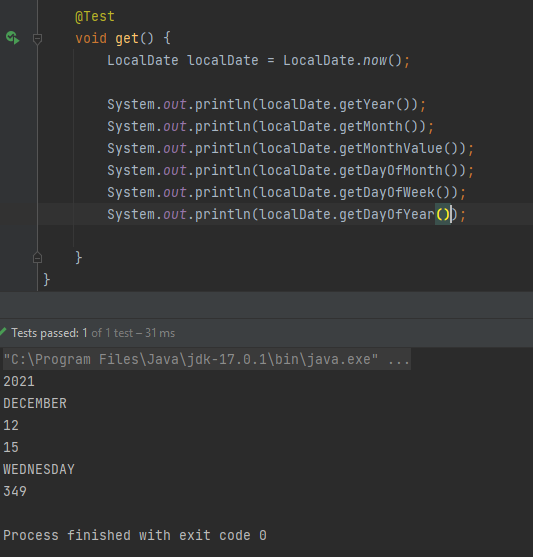
**Manipulasi LocalDate**

* Object LocalDate juga bisa kita manipulasi, seperti menambah tanggal atau mengurangi tanggal.
* Untuk menambah tanggal, kitab isa gunakan method plus, seperti plusYears, plusMonths, plusDays, dan lain – lain
* Untuk mengurangi tanggal, kitab isa gunakan method minus, seperti minusYears, minusMonths, minusDays, dan lain – lain
* Perlu diingat, manipulasi LocalDate akan menciptakan object LocalDate baru, artinya object aslinya tidak akan berubah, karena bersifat immutable



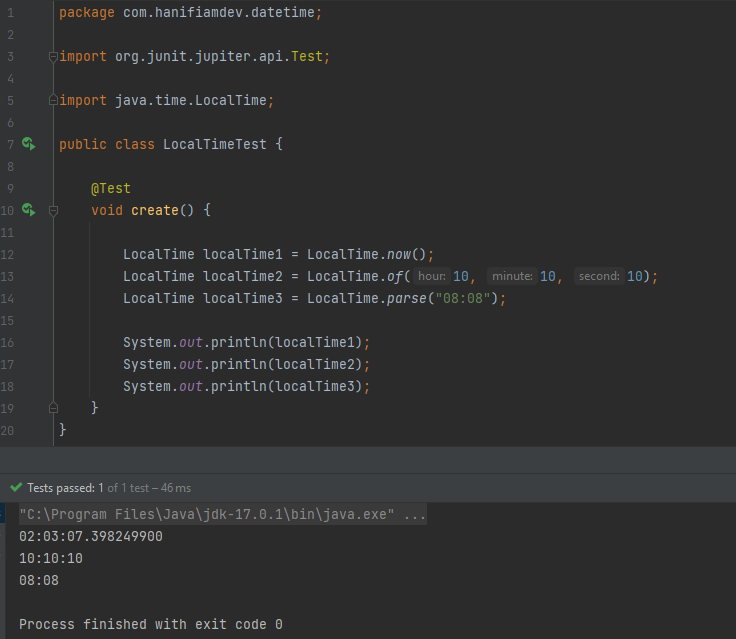
**Mengambil Tanggal LocalDate**

* LocalDate juga mendukung pengambilan detail data tanggal nya, seperti tahun, bulan, jari dalam bulan, hari dalam tahun, dan lain – lain
* Untuk mengambil data tanggal, kita bisa menggunakan method get, seperti getYear, getMonth, dan lain-lain

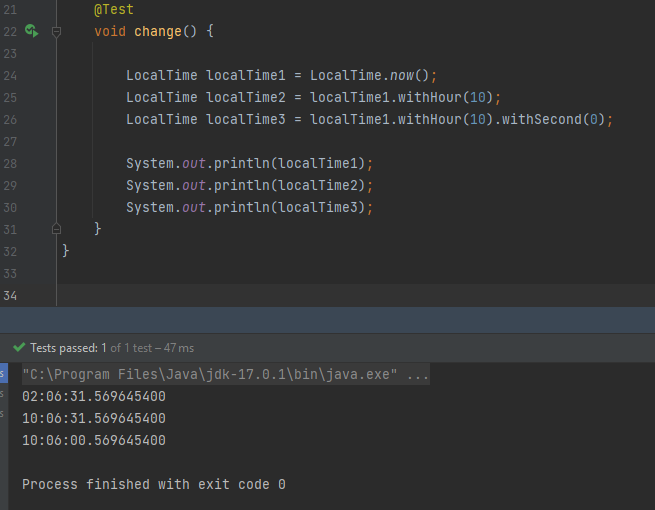


## **LocaleTime**

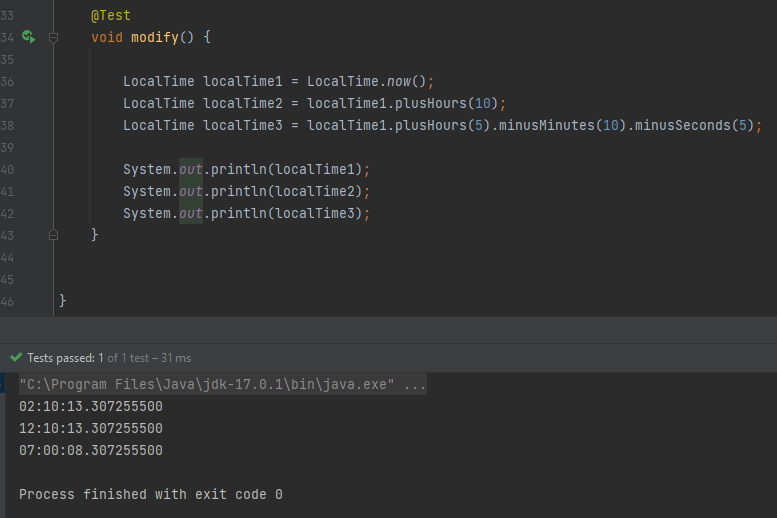
* Berbeda dengan LocalDate, LocalTime merupakan representasi data waktu (tanpa tanggal)
* Cara penggunaanya dan method-method nya hampir sama dengan LocalDate
* LocalTime juga immutable, jadi kita tidak bisa mengubahnya setelah datanya dibuat, jika kita ubah, maka akan menghasilkan object LocalTime baru
* Format standard LocalTime adalah HH:mm:ss:nano, dimana second dan nano second nya optional



**Mengubah LocalTime**



**Manipulasi LocalTime**

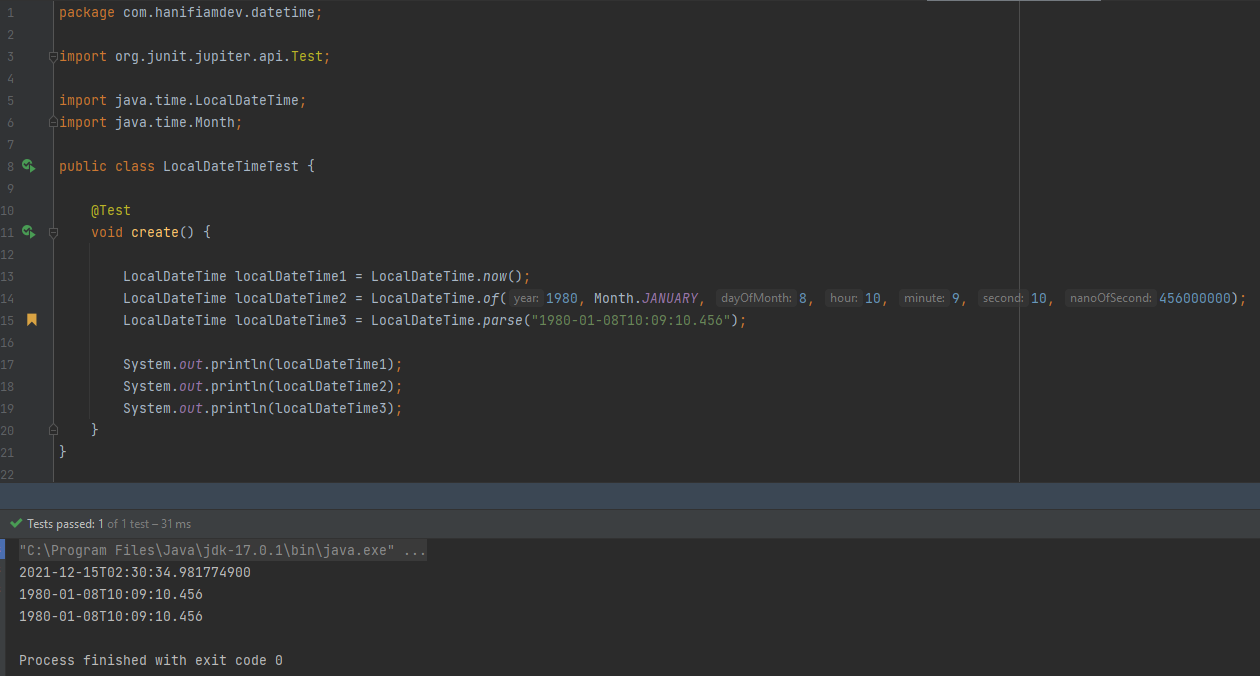


**Mengambil Waktu LocalTime**

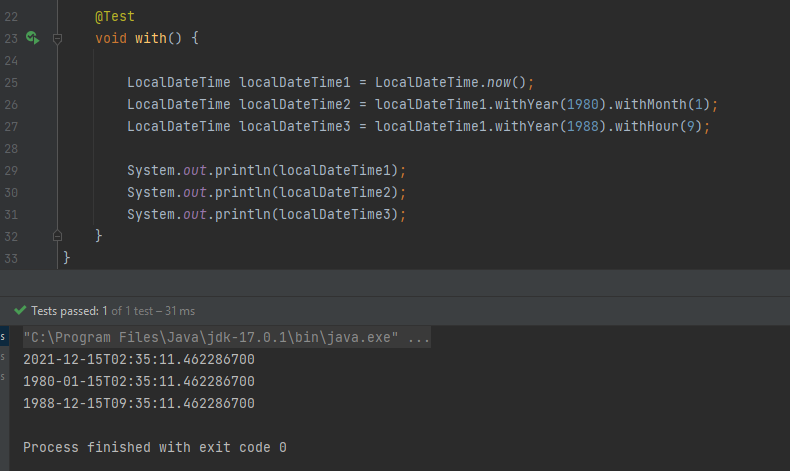


## **LocalDateTime**

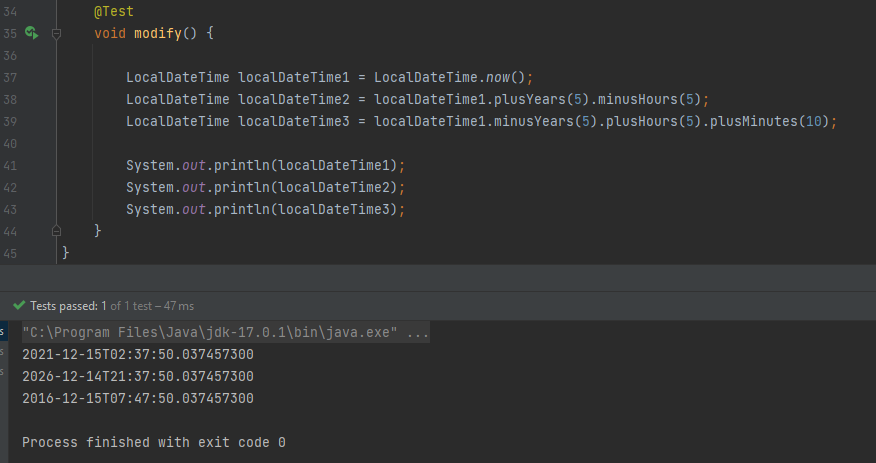
* Seperti dari nama class nya, LocalDateTime, class ini digunakan sebagai representasi tanggal dan waktu, mirip seperti class Date
* LocalDateTime juga immutable, jadi tidak bisa diubah setelah dibuat, jika diubah otomatis akan membuat object baru
* Cara pembuatan, cara mengubah, memanipulasi nya pun sama seperti LocalDate dan LocalTime
* Dan format default untuk LocaleDate Time adalah menggunakan format yyyy-MM-ddTHH:mm:ss.nano



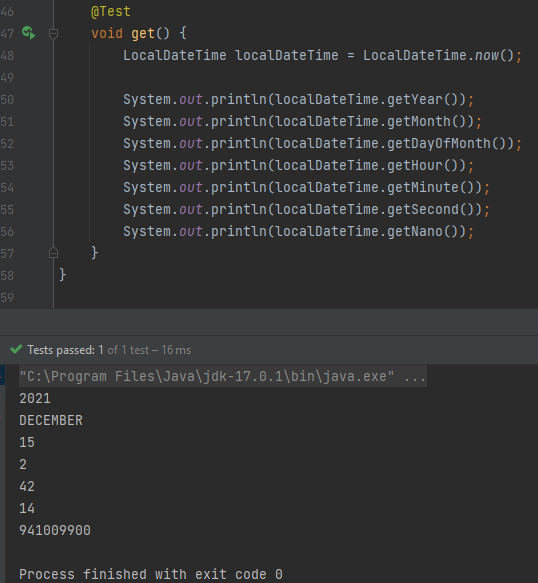
**Mengubah LocalDateTime**



**Modify LocalDateTime**

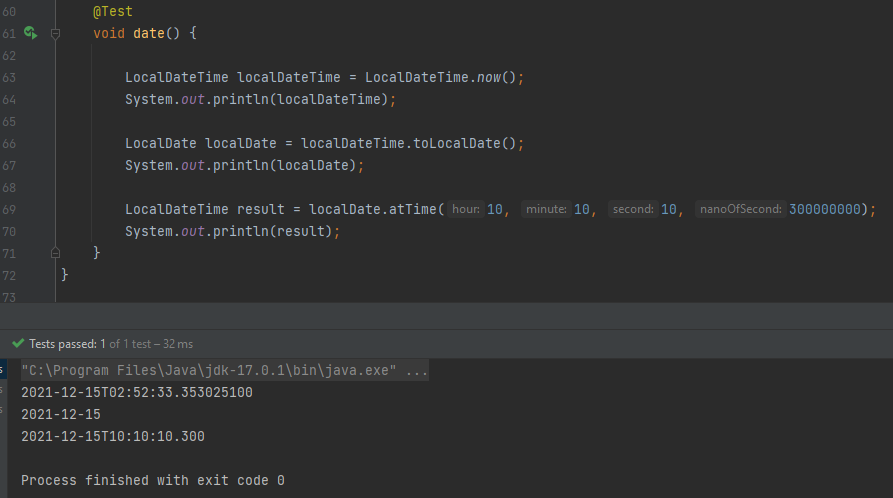


**Mengambil Data LocalDateTime**



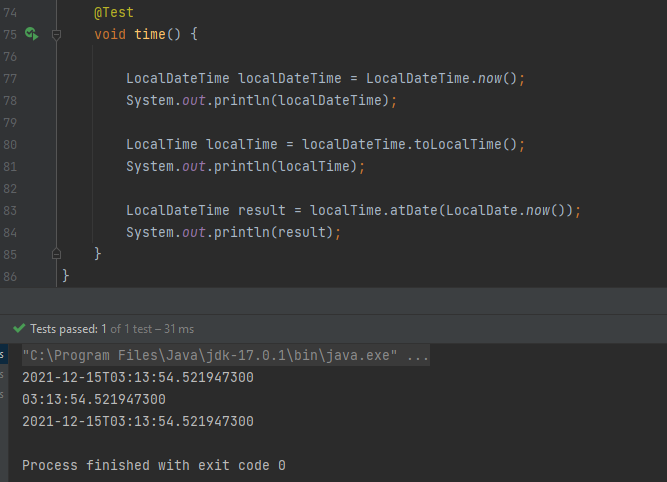
**Konversi dari dan ke LocalDate**

* Kadang ada kebutuhan kita melakukan konversi data dari LocalDate ke LocalDateTime atau juga sebaliknya
* Untuk melakukan konversi LocalDateTime ke LocalDate, kitab isa menggunakan method toLocalDate()
* Sedangkan untuk melakukan konversi dari LocalDate ke LocalDateTime, kitab isa menggunakan method atTime()



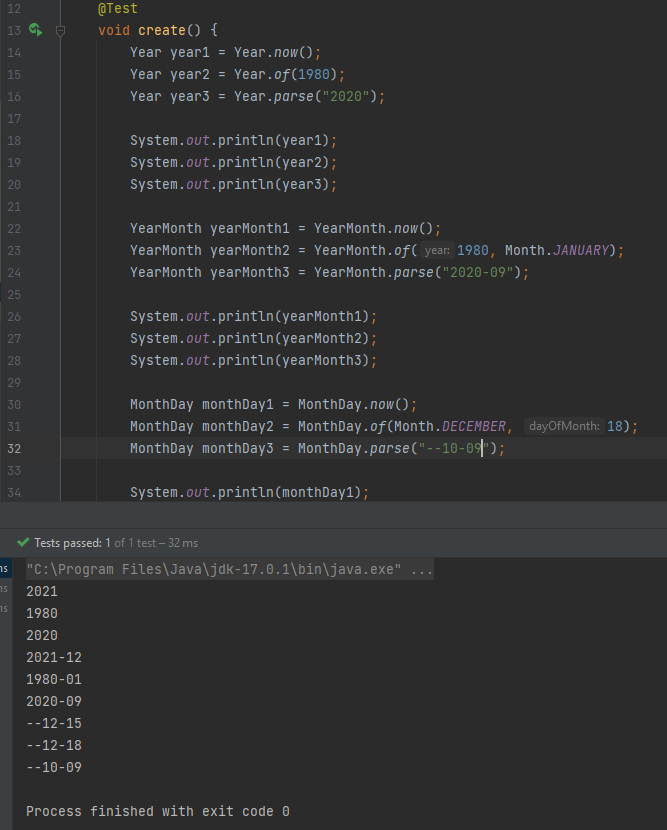
**Konversi dari dan ke LocalTime**

* Selain melakukan konversi untuk tipe data LocalDate, kita juga bisa lakukan ke dan dari tipe data LocalTime.
* Untuk melakukan konveersi dari LocalDateTime ke LocalTime, kita bisa menggunakan method toLocalTime()
* Sedangkan untuk melakukan konversi dari LocalTime ke LocalDateTime, kita bisa menggunakan method atDate()

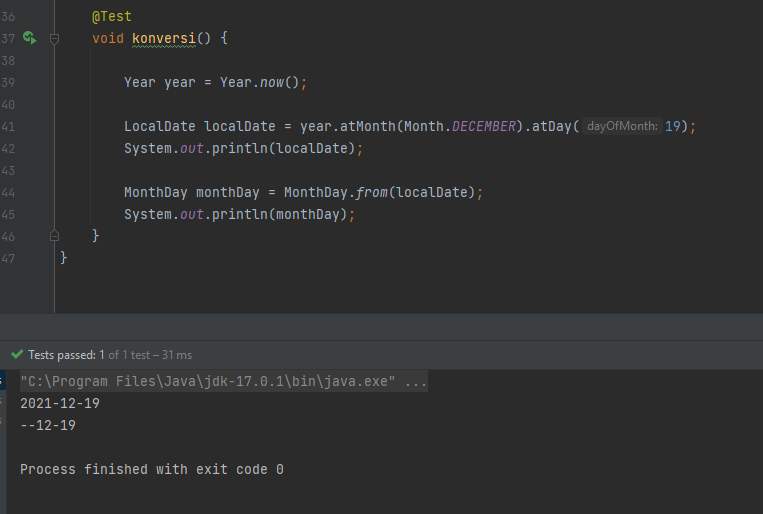


## **Year, YearMonth dan MonthDay**

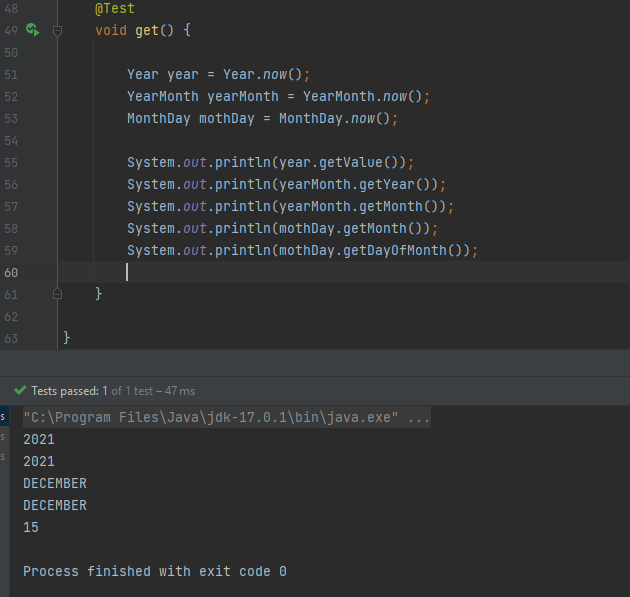
* Di Date & Time Api baru, terdapat class Year, YearMonth dan MonthDay
* Seperti dari nama class nya, Year digunakan untuk tanggal yang hanya berisi data tahun, dan YearMonth adalah tanggal yang berisi data tahun dan bulan, dan MonthDay adalah tanggal berisi bulan dan hari
* Kenapa menggunakan Year? Kenapa tidak langsung menggunakan angka saja? Misal 2020. Di Year, sudah banyak sekali method yang bis akita gunakan untuk manipulasi data tanggal dan konversi ke tipe lain seperti LocalDate misalnya
* Begitu juga dengan YearMonth dan MonthDay
* Format default untuk Year adalah yyyy dan format untuk YearMonth adalah yyyy-MM dan format untuk MonthDay adalah –MM—dd



**Konversi dari Year ke YearMonth**



**Mengambil Data Year dan YearMonth**

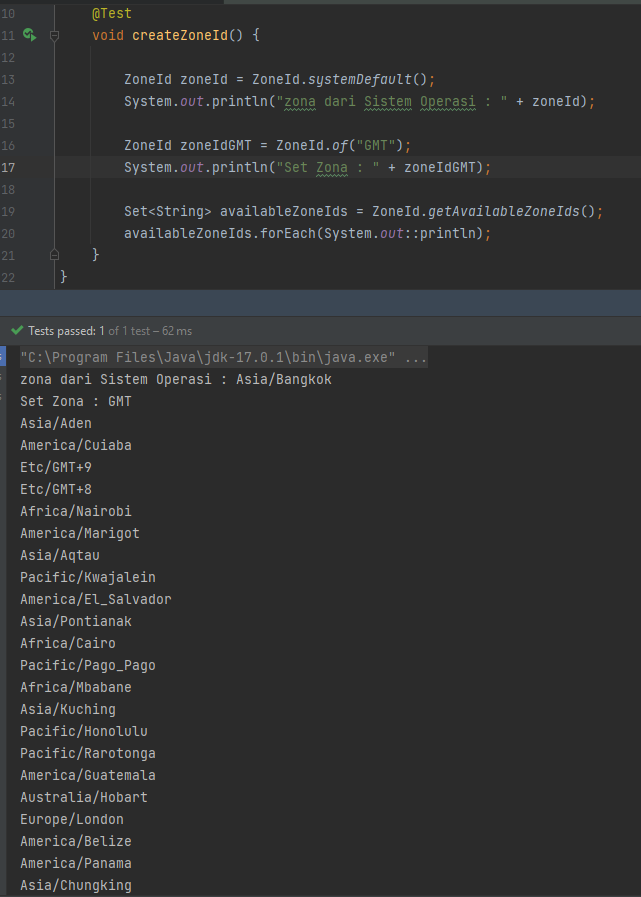


## **ZoneId dan ZoneOffset**

* Sebelum kita sudah tahu bahwa terdapat class TimeZone di Java sebagai representasi time zone
* Namun di Java Date & Time API terbaru, terdapat class baru untuk mendukung time zone, yaitu ZoneId dan ZoneOffset

**ZoneId**

* ZoneId mirip dengan TimeZone, dimana ini merupakan representasi time zone id
* Untuk mendapatkan default time zone, kita bisa menggunakan method ZoneId.systemDefault()
* Untuk membuat ZoneId, caranya bisa menggunakan method ZoneId.of(“Zone Id”)
* Dan untuk mendapatkan semua daftar time zone id yang didukung oleh Java, kita bisa menggunakan method getAvailableZoneIds()



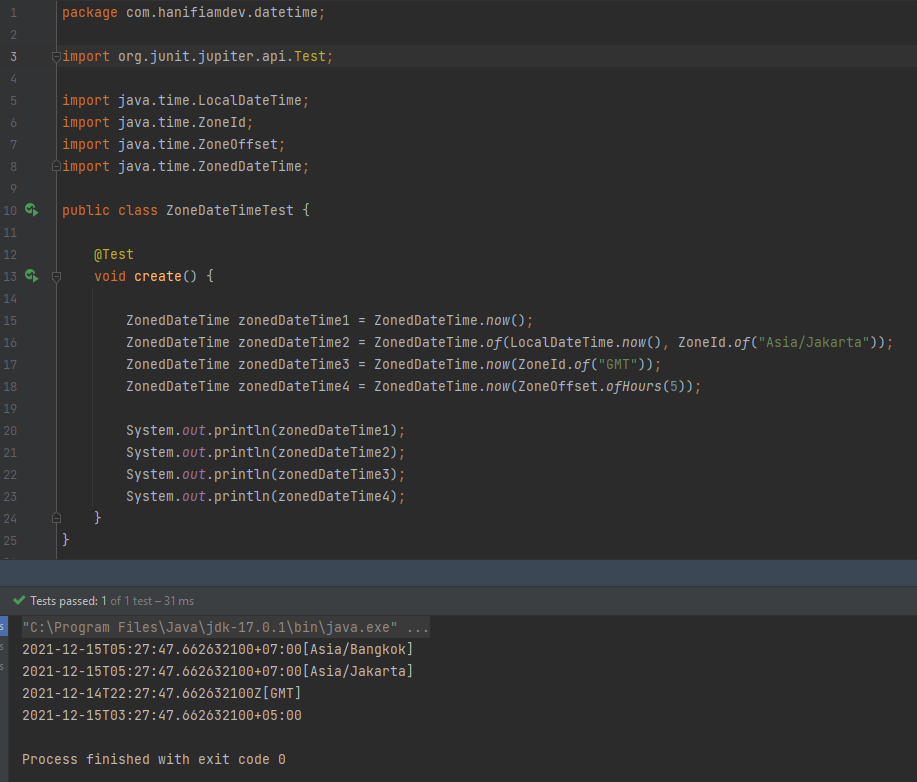
**ZoneOffset**

* Jika ZoneId merupakan representasi timezone menggunakan time zone id
* Pada class ZoneOffset, ini adalah format time zone dengan offset dari time zone Greenwich/UTC, misalnya +07:00 atau -01:00
* Untuk membuat ZoneOffset, kitab isa menggunakan method of
* ZoneOffset adalah turunan dari ZoneId, jadi semua parameter yang menerima object ZoneId, bis a kita isi dengan ZoneOffset

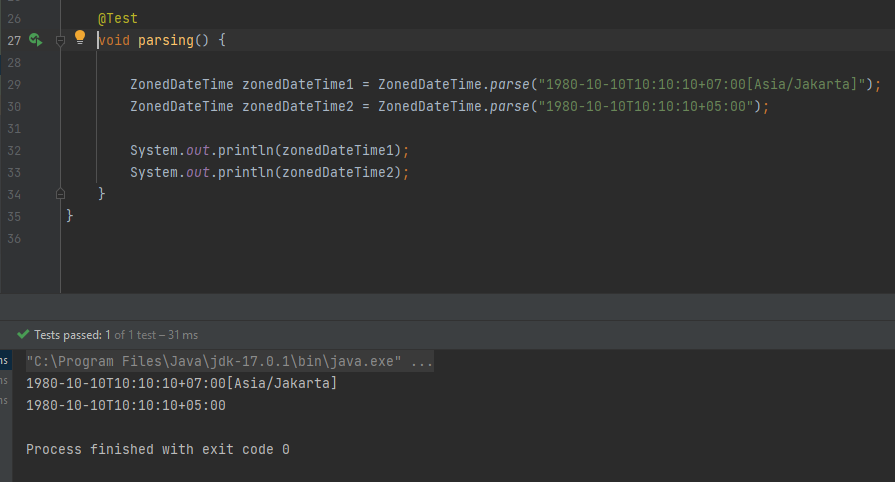


## **ZoneDateTime**

* Pada class LocalDateTime, secara default tidak ada informasi zone sama sekali pada class tersebut
* Jika kita ingin membuat tanggal dan waktu yang mendukung time zone, kita bisa menggunakan class ZonedDateTime
* Class ZonedDateTime sama seperti LocalDateTime, perbedaannya hanyalah mendukung time zone
* Format default untuk ZonedDate Time adalah yyyy-MM-ddTHH:mm:ss.nano(+/-)ZoneOffset[ZoneId], dimana ZoneId tidak wajib, dan jika ZoneId diisi, maka nilai ZoneOffset akan di hiraukan

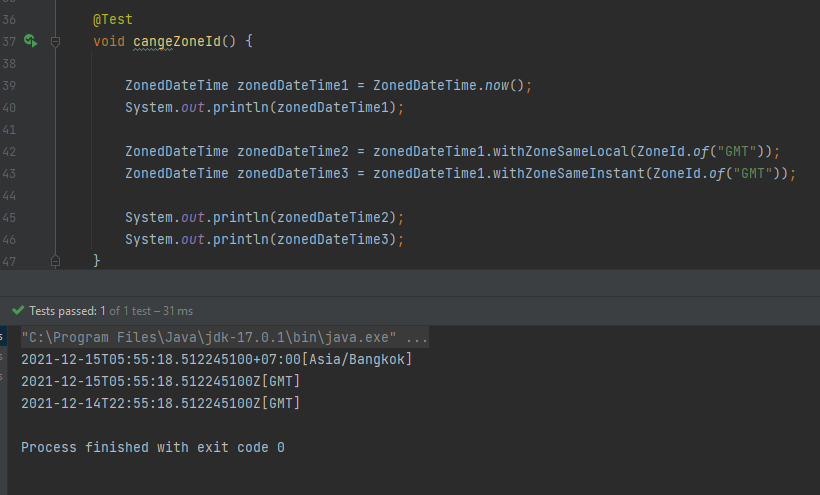


**Parsing ZonedDateTime**



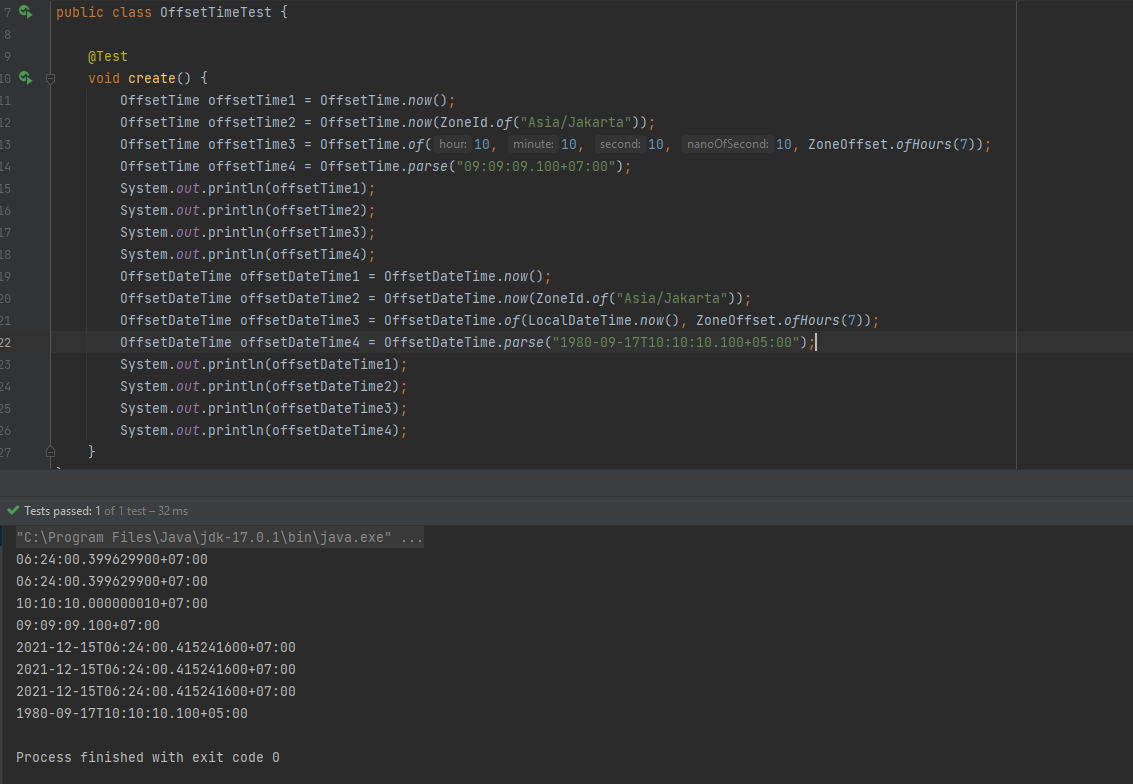
**Mengubah Time Zone**

* Untuk megnubah time zone pada ZonedDateTime, terdapat du acara
* Pertama, mengubah time zone, tanpa merubah tanggal dan waktu. Caranya dengan menggunakan method withZoneSameLocal(ZoneId)
* Kedua, mengubah time zone, sehingga tanggal dan waktu mengikuti timezone yang baru. Caranya dengan menggunakan method withZoneSameInstance(ZoneId)



## **OffsetTime dan OffsetDateTime**

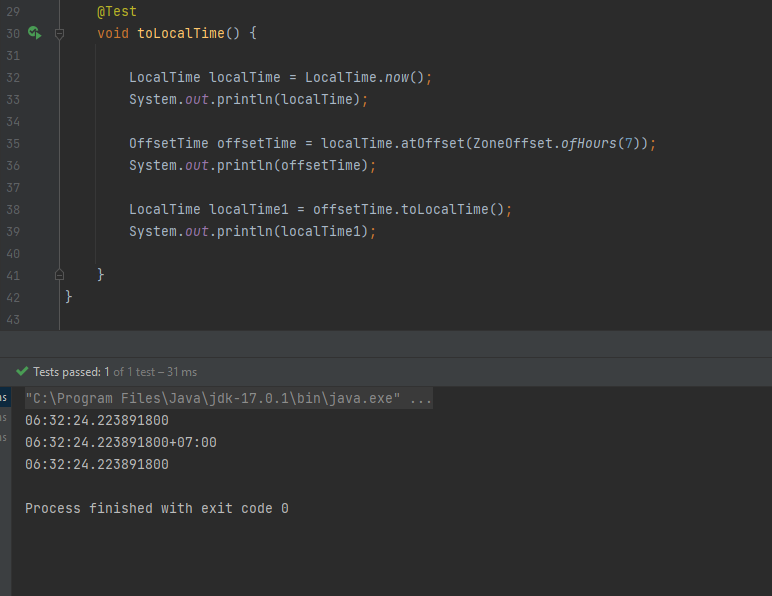
* Java Date & Time API memiliki class yang bernama OffsetTime dan OffsetDateTime
* Class OffsetTime adalah seperti LocalTime,namun memiliki time zone offset
* Dan Class OffsetDateTime adalah seperti LocalDateTime, namun memiliki time zone offset
* Sekilas OffsetDateTime mirip dengan ZonedDateTime, yang membedakan adalah kalo OffsetDate Time hanya bisa menggunakan ZoneOffset, tidak bisa menggunakan ZoneId
* Format default OffsetTime adalah HH:mm:ss(+/-)ZoneOffset
* Format default OffsetDate Time adalah yyyy-MM-ddTHH:mm:ss(+/-)ZoneOffset



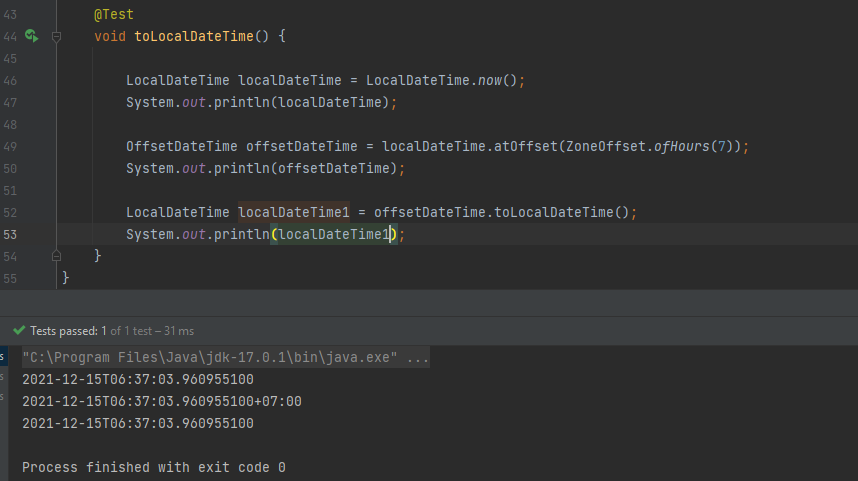
**Konversi ke dan dari Non Offset**

* Apa yang bisa kita lakukan di LocalTime dan LocalDateTime bisa juga dilakukan di OffsetTime dan OffsetDateTime
* Selain itu kita juga bisa konversi data dari dan ke non offset

LocalTime

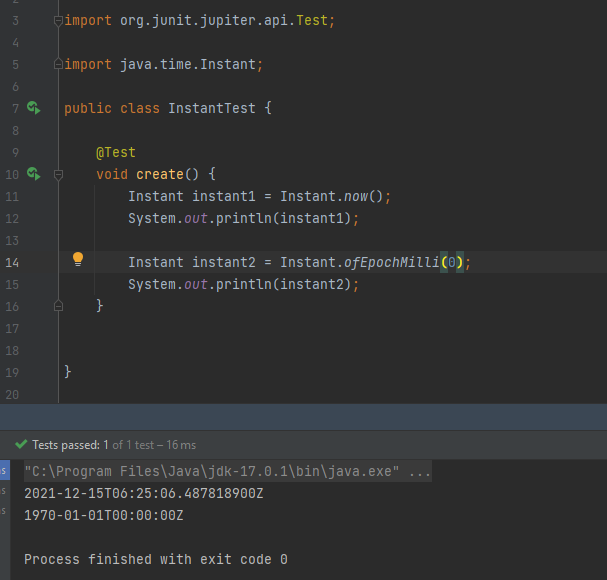


LocalDateTime

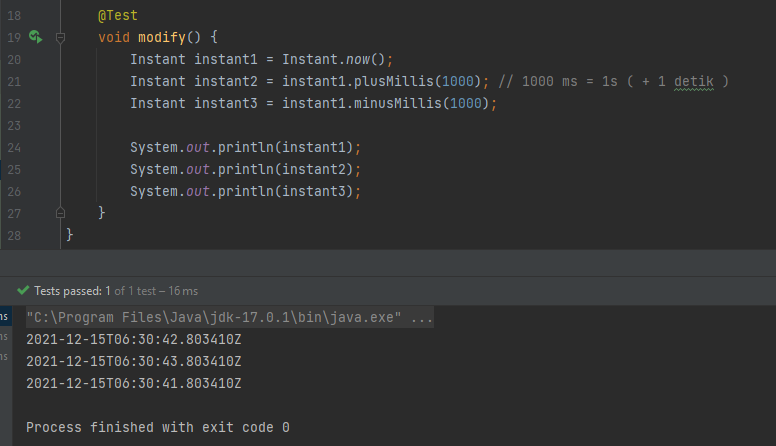


## **Instant**

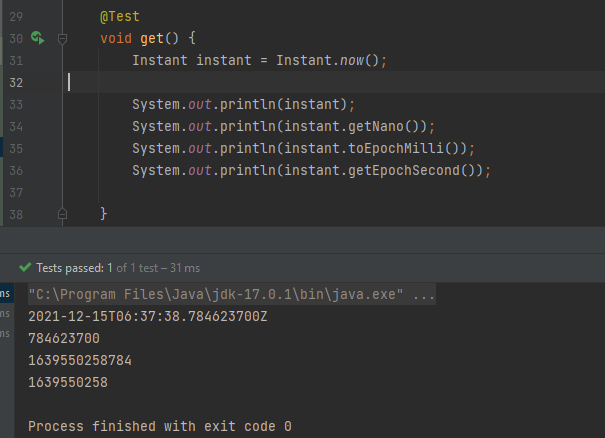
* Sebelumnya, milliseconds direpresentasikan dengan tipe data long, di Java Date & Time Api baru, implementasi milliseconds direpresentasikan dalam class instant
* Instant juga sama seperti class Date, dia menggunakan millisecond setelah Unix EPOCH 1970-01-01T00:00:00Z
* Artinya secara time zone, Instant menggunakan time zone UTC (00:00)



**Mengubah Instant**

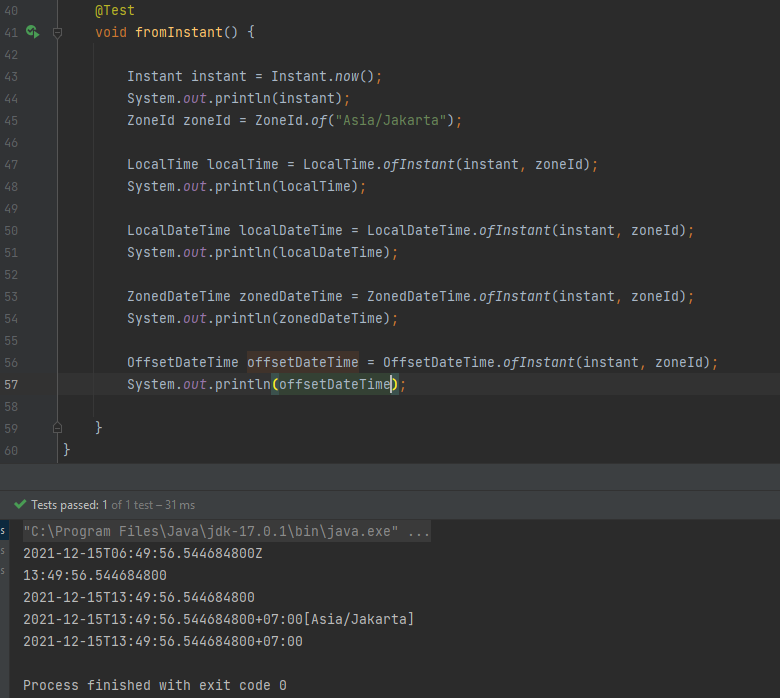


**Mengambil Data Instant**



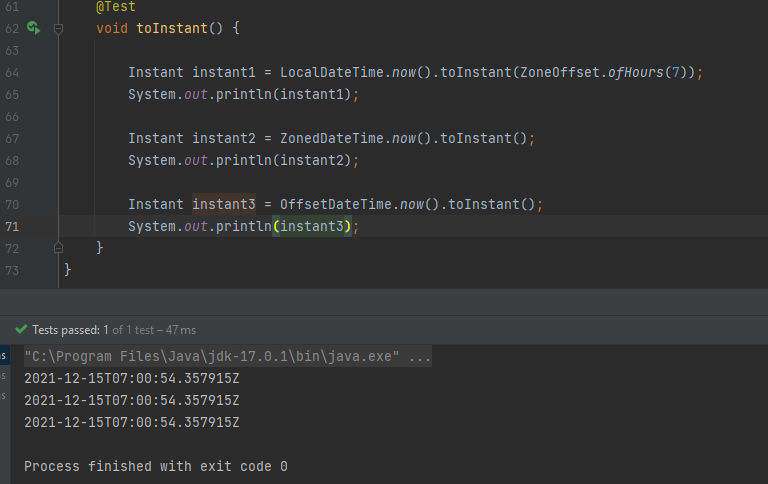
**Konversi dari Instant**

* Karena Instant berisikan millisecond, jadi kita bisa lakukan konversi ke tipe data lainnya, seperti LocalDate, LocalTime dan ZonedLocalTime
* Namun karena Instant menggunakan zona UTC, jadi kita perlu memberi tahu time zone apa yang akan kita gunakan ketika kita lakukan konversi
* Untuk melakukan konversi dari Instant ke tipe data lainnya, kitab isa menggunakan method ofInstant(instant, zone) ketika membuat object nya



**Konversi ke Instant**

* Selain itu konversi juga bisa dilakukan ke instant, dari tipe data LocalDateTime dan ZonedDateTime
* Namun karena instant menggunakan zone UTC, jadi kita perlu menambahkan ZoneOffset jika konversi dari tipe data yang tidak memiliki time zone

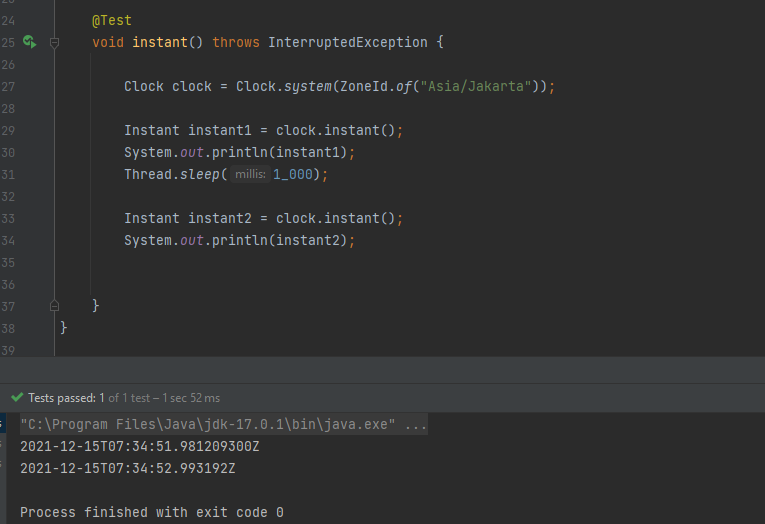


## **Clock**

* Seperti nama class nya, Clock adalah representasi tanggal dan waktu saat ini mengikuti time zone yang kita pilih
* Best practice nya sebenarnya jika kita ingin menggunakan tipe data di Date & Time API yang multi time zone adalah menggunakan Clock, sehingga jika kita ingin membuat data baru, kita bisa memanfaatkan Clock

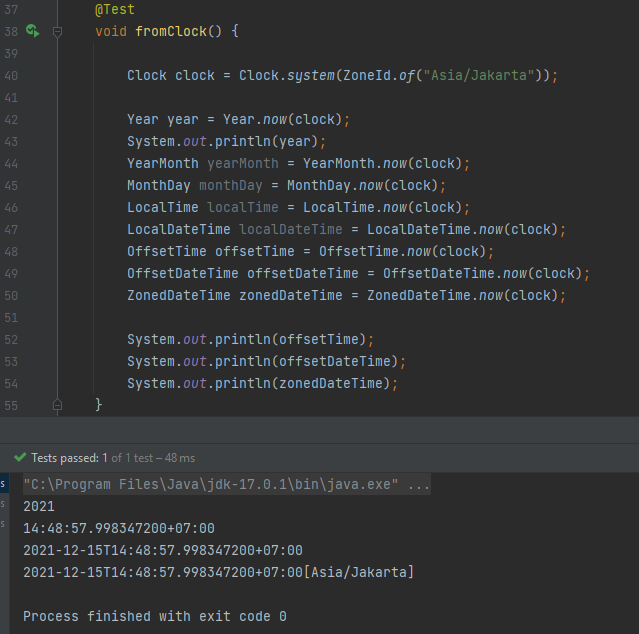
**Mendapatkan Instant di Clock**

* Kita sudah tahu bahwa Clock itu akan selalu berjalan, tidak pernah berhenti
* Dan representasi millisecond di Date & Time API baru adalah Instant
* Oleh karena itu, untuk mendapatkan tanggal dan waktu saat ini sesuai dengan time zone di Clock, kitab isa menggunakan method instant(), dan otomatis akan mengembalikan Instant saat ini sesuai dengan Clock nya
* Perlu diingat, karena Instant itu selalu menggunakan time zone UTC, jadi jika clock nya tidak menggunakan UTC, maka akan secara otomatis dikonversi ke time zone UTC



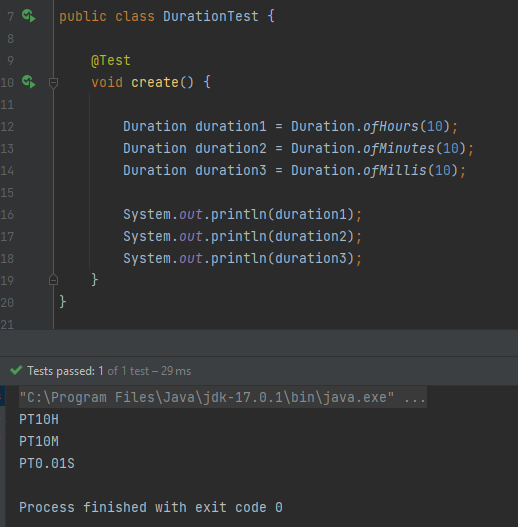
**Membuat Tanggal dan Waktu dari Clock**

* Karena jika menggunakan Instant kita harus melakukan konversi secara manual ke LocalDateTime atau ZonedDateTime, karena bisa saja time zonenya berbeda
* Untungnya, kita juga bisa membuat tipe data tanggal dan waktu langsung menggunakan Clock, cara menggunakan method now(Clock)
* Secara otomatis informasi time zone Clock akan dibawa di tipe data tanggal dan waktu yang kita buat

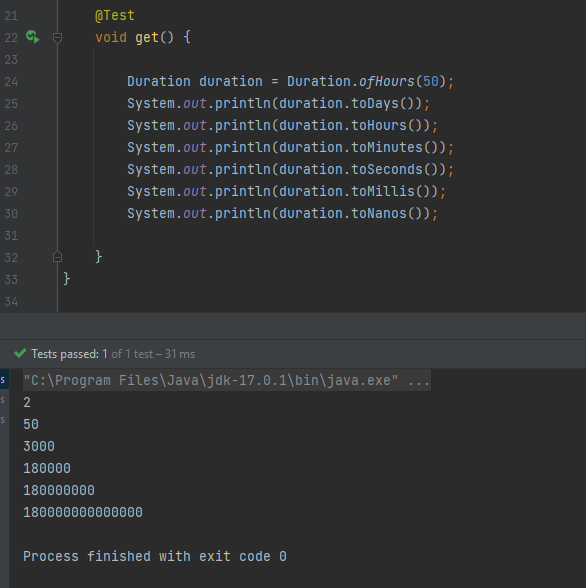


## **Duration**

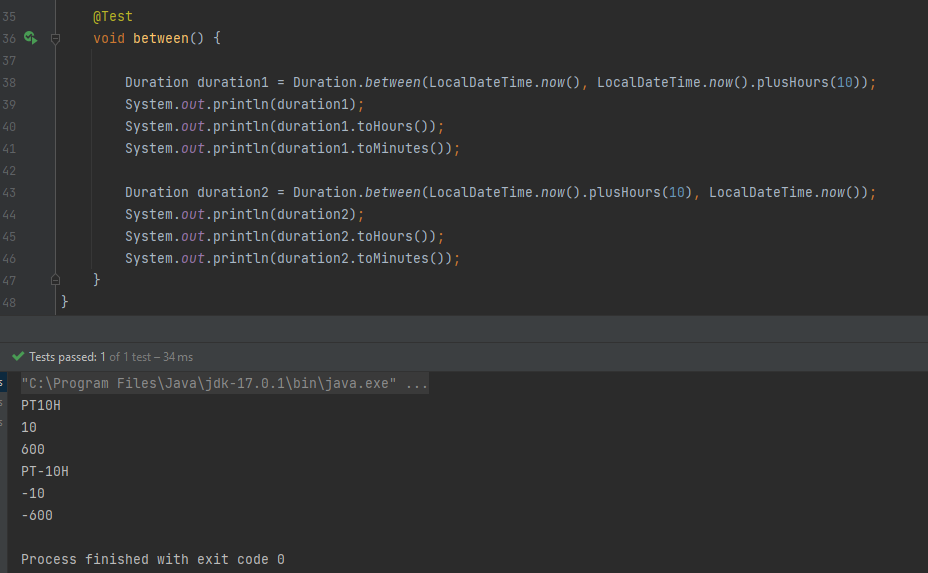
* Class Duration adalah representasi dari data durasi waktu
* Durasi waktu yang terdapat di class Duration
* Dengan menggunakan Duration, kita bisa dengan mudah mengkonversi durasi ke waktu yang kita inginkan, misal hour, minute, second, dan nano second



**Mengambil Data Duration**

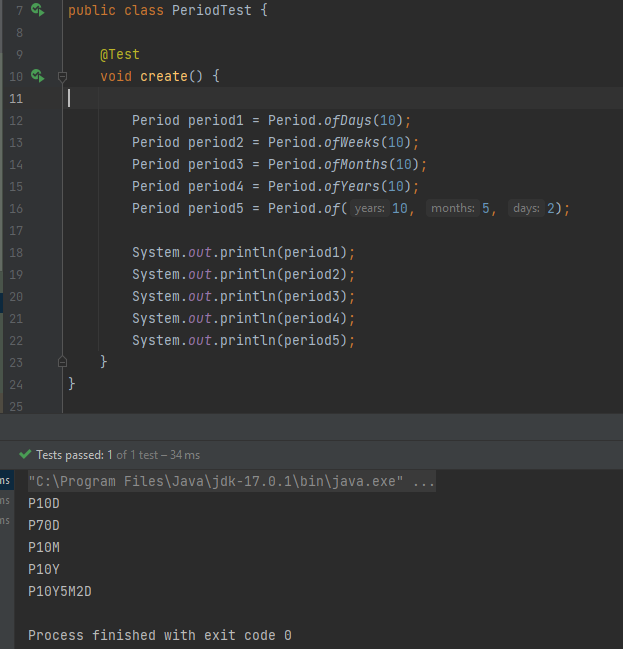


**Menghitung Duration**

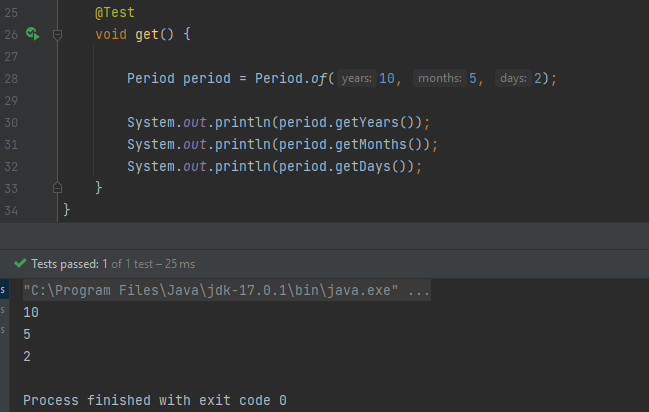


## **Period**

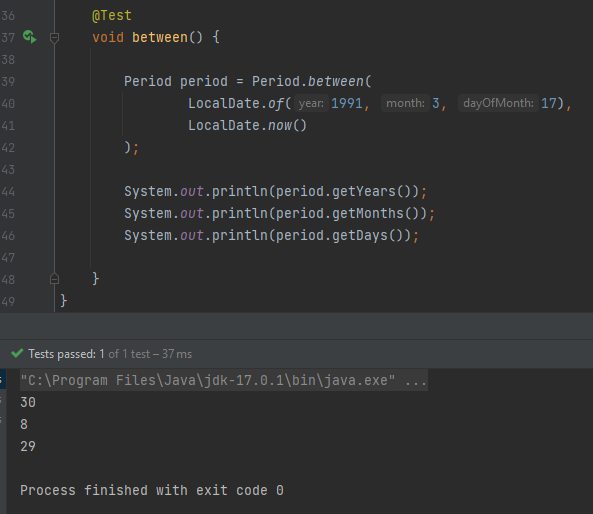
* Class Period mirip dengan Duration, yang membedakan adalah Period digunakan untuk tanggal
* Cara penggunaan Period hampir sama dengan duration



**Mengambil Data Period**



**Menghitung Period**



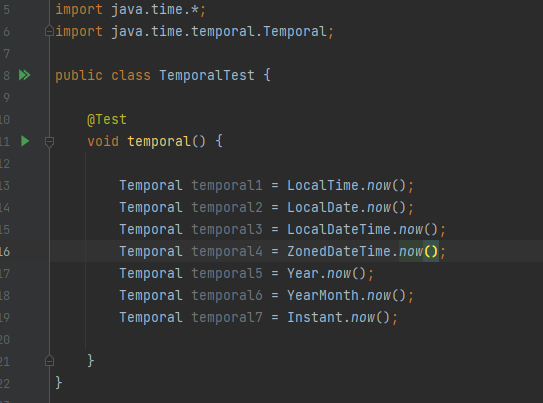
## **Temporal**

* Di dalam package java.time, sebenarnya ada package java.time.temporal
* Di dalam package java.time, temporal terdapat banyak sekali interface yang merupakan kontrak dari Java Date & Time API

|  |  |
| --- | --- |
| **Interface** | **Keterangan** |
| Temporal | Interface untuk temporal object, seperti date, time , dan lain - lain |
| TemporalAmount | Interface untuk jumlah waktu, seperti duration dan period |
| TemporalUnit | Interface untuk unit satuan temporal, seperti jam, menit, hari |
| TemporalField | Interface untuk field dalam temporal data |
| TemporalQuery | Interface untuk query data dari TemporalAccessor ( super interface Temporal) |
| TemporalAdjuster | Strategi untuk menyesuaikan objek temporal |

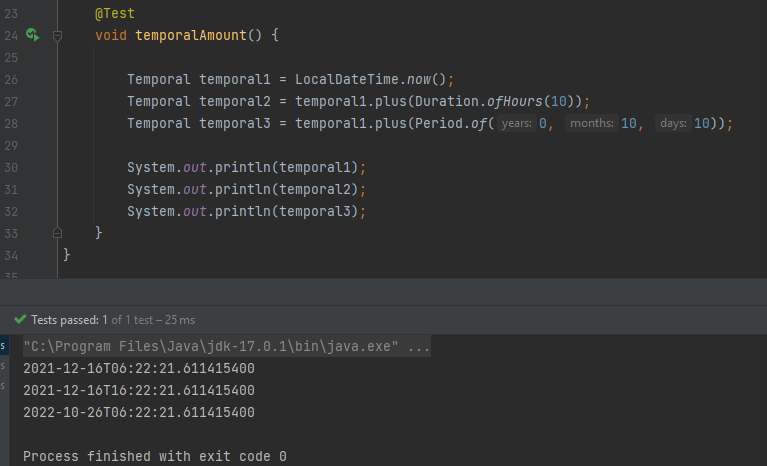
**Temporal**

* Hampir semua tipe data tanggal dan waktu di Java Date & Time adalah implementasi dari interface Temporal
* Maka dari itu, jika diperhatikan, hampir semua tipe datanya memiliki method method yang hampir sama



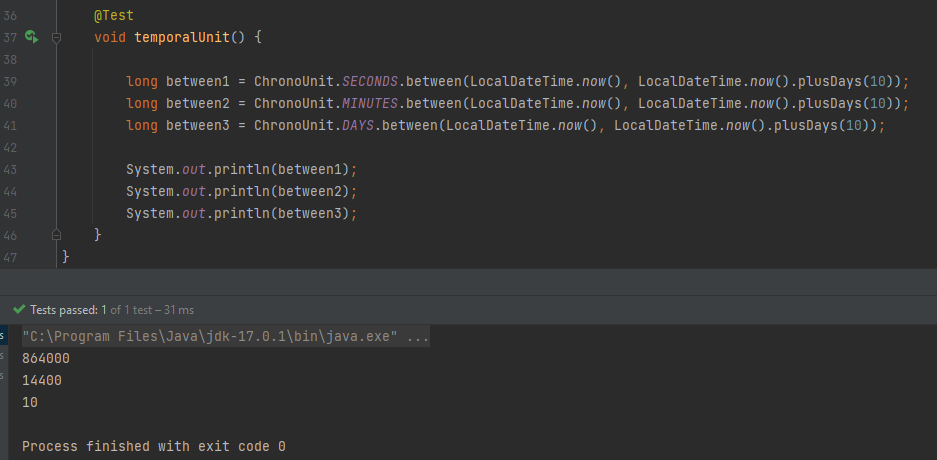
**Temporal Amount**

* Duration dan Period adalah implementasi dari interface TemporalAmount
* Salah satu method yang menggunakan TemporalAmount di Temporal adalah plus() dan minus()
* Artinya, dengan ini kitab isa menggunakan object TemporalAmount untuk melakukan penambahan/pengurangan tanggal dan waktu



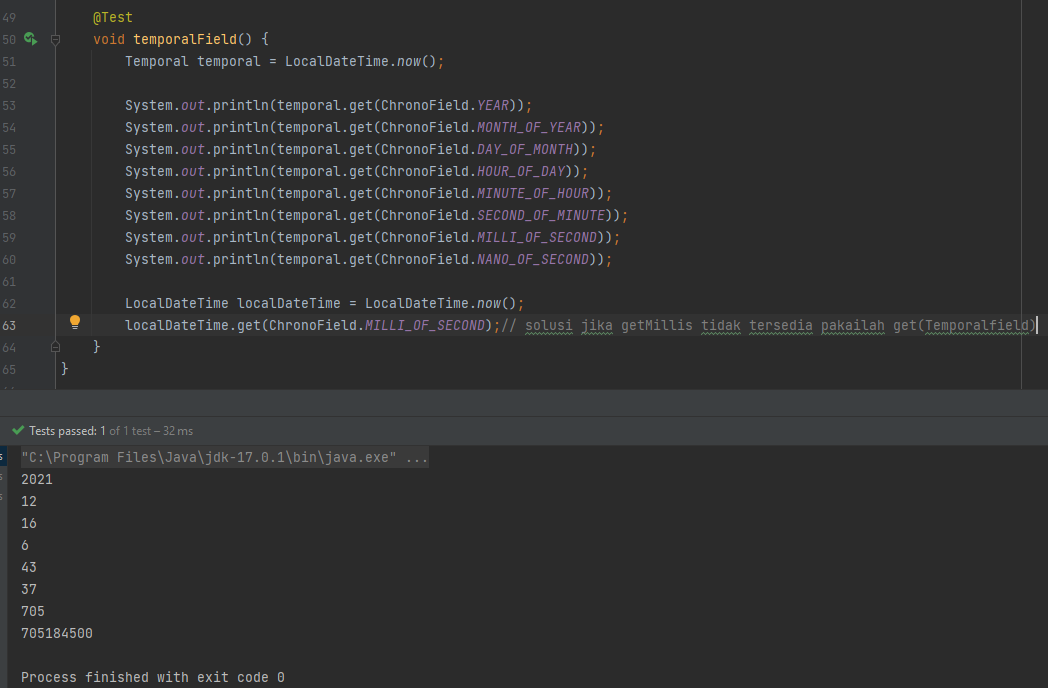
**TemporalUnit**

* TemporalUnit adalah implementasi dari unit satuan waktu
* Implementasi TemporalUnit adalah sebuah enum ChronoUnit
* TemporalUnit selain sebagai informasi satuan waktu, bisa juga digunakan untuk menghitung durasi waktu



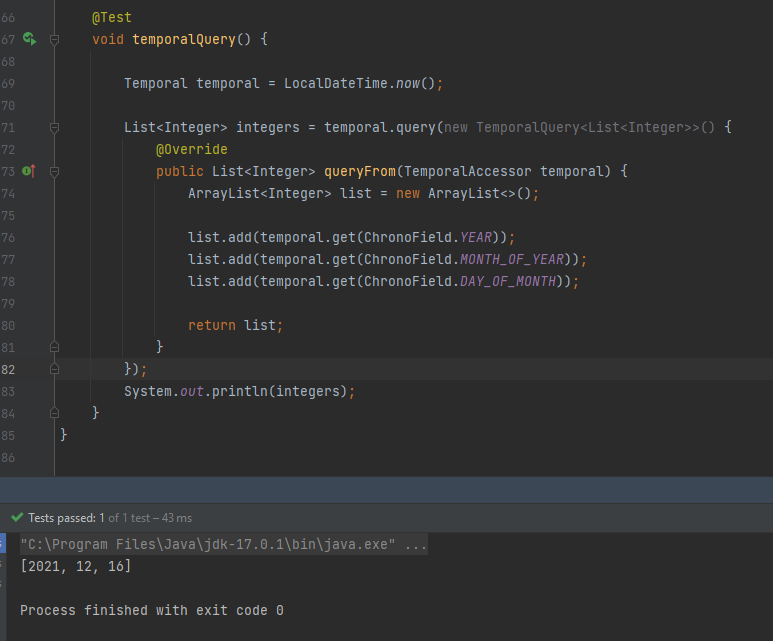
**TemporalField**

* TemporalField adalah informasi field yang terdapat dalam sebuah tipe data
* Semua object Temporal memiliki method bernama get(TemporalField) atau getLong(TemporalField) untuk mendapatkan seputar field pada object tersebut, sesuai dengan Field yang kita inginkan
* Implementasi TemporalField adalah enum bernama ChronoField



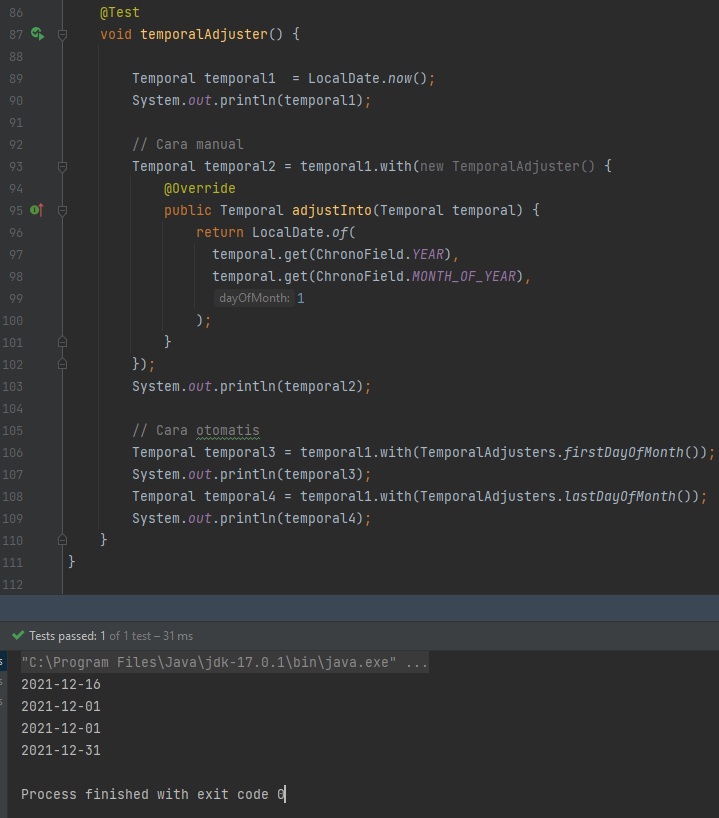
**TemporalQuery**

* TemporalQuery merupakan lambda interface yang bisa kita gunakan untuk mengambil informasi dari data TemporalAccessor
* TemporalQuery adalah generic type, jadi kita bisa mengembalikan tipe data apapun pada hasil query yang kita lakukan di TemporalAccessor



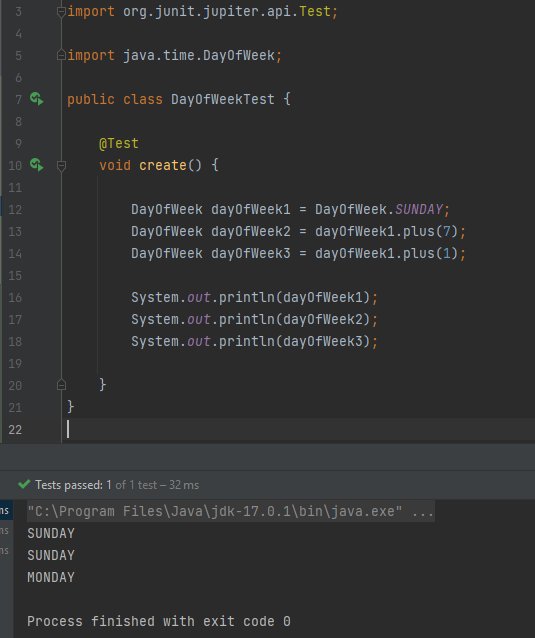
**TemporalAdjuster**

* TemporalAdjuster adalah strategi untuk menyesuaikan objek temporal
* Kita bisa melakukan implementasi penyesuaian object sendiri, atau kita juga bisa menggunakan helper class benama TemporalAdjusters
* Terdapat banyak static method di TemporalAdjusters yang bis akita gunakan untuk mempermudah melakukan objek temporal



## **DayofWeek**

* Yang menarik di Java Date & Time adalah, memiliki implementasi data hari
* Data hari ini berupa Enum DayOfWeek, namun dia adalah turunan dari Temporal, sehingga operasi yang bisa dilakukan di Temporal bis akita lakukan disini, seperti menambah atau mengurangi



## **Parsing dan Formatting**

* Sebelumnya di awal - awal kita sering sekali menggunakan method parse untuk melakukan parsing data String menjadi objek Temporal
* Sebenarnya proses parsing dan formatting tersebut dilakukan oleh class java.time.format.DateTimeFormatter

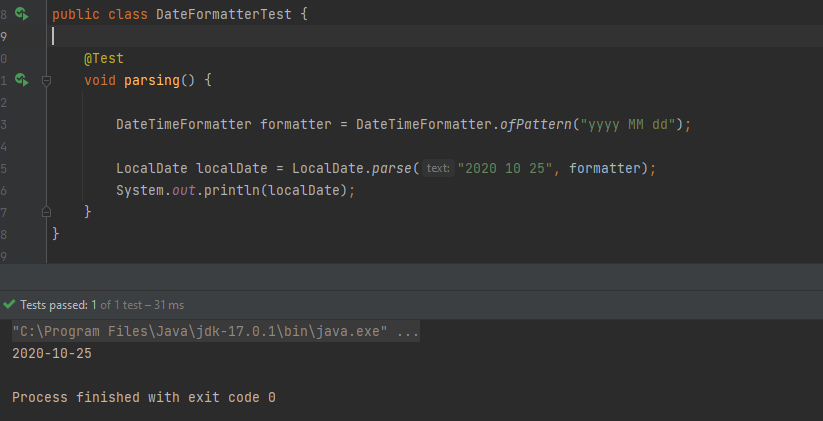
**Pattern**

* Pattern untuk membuat DateTimeFormatter hampir mirip dengan pattern yang sudah pernah dibahas di course tentang Java Internationalization
* Atau lebih detail bisa dilihat di halaman ini :

<https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/time/format/DateTimeFormatter.html>

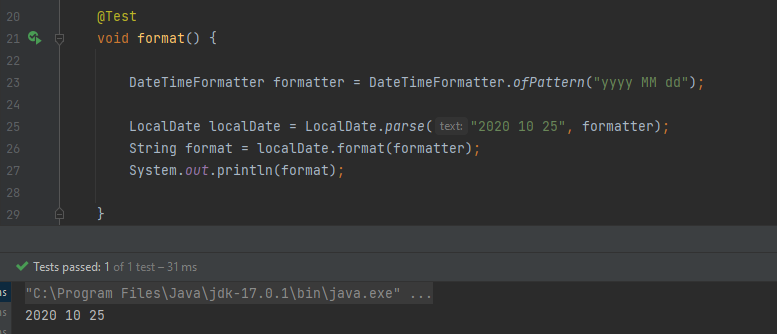
**Parsing**

* Sebelumya kita sudah bahas tentang default pattern untuk parsing semua tipe objek temporal
* Jika kita ingin mengubah format pattern nya, kita bisa menggunakan DateTimeFormatter pada saat melakukan parsing, misal :
* LocalDate.parse(“yyyy MM hh”, dateTimeFormatter



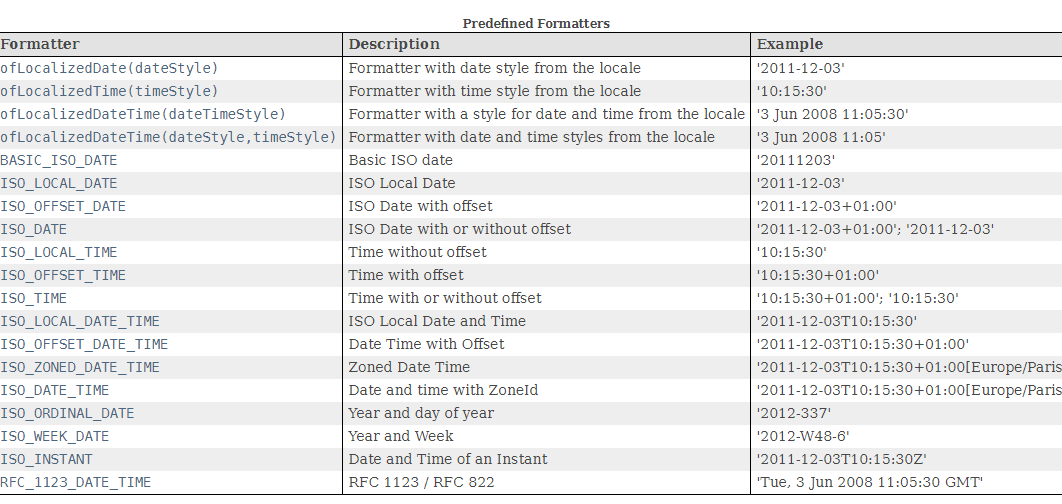
**Formatting**

* Untuk formatting juga kita bisa menggunakan DateTimeFormatter
* Untuk melakukan formatting, kitab isa menggunakan method format(Date TimeFormatter) milik objek temporal



**Default Formatter**

* Selain kitab isa membuat formatter sendiri menggunakan pattern
* Sudah disediakan juga formatter default, sehingga jika kita ingin menggunakan formatter misal yang sudah standar internasional, kita tidak perlu membuat ulang menggunakan pattern





**Internationalization**

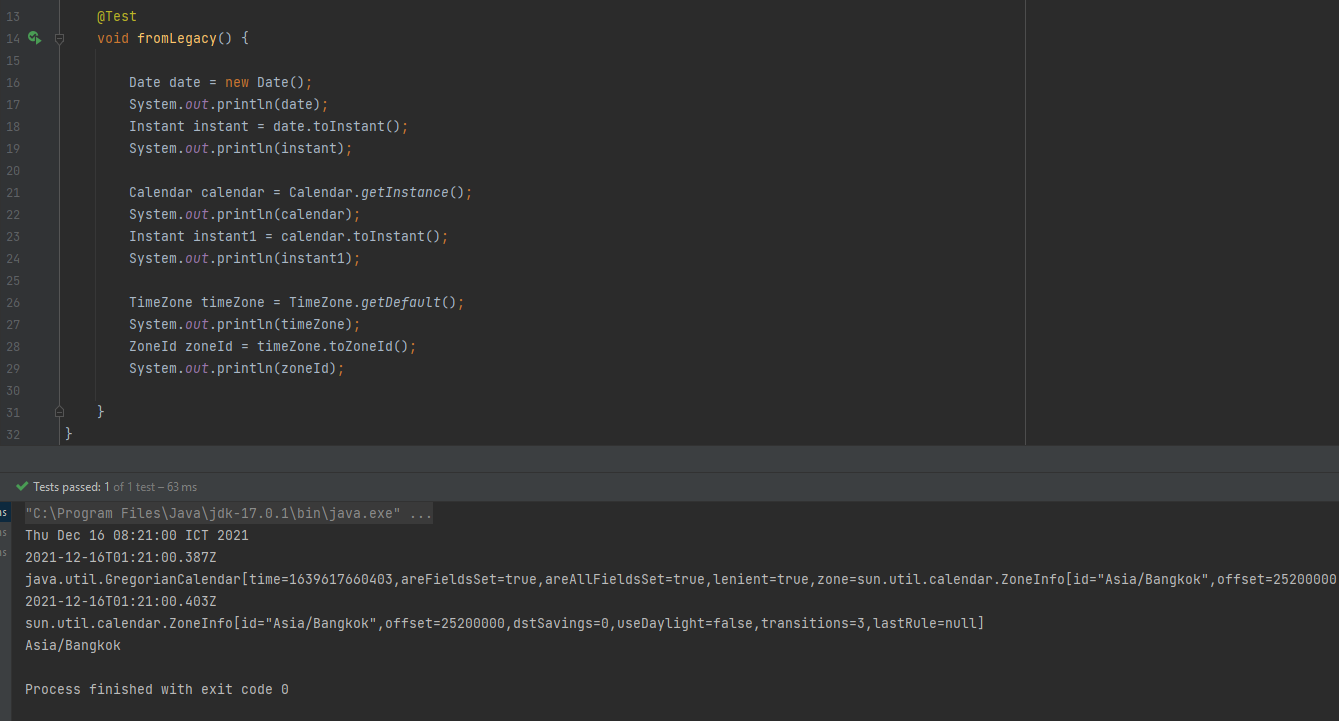
* DateTimeFormatter juga mendukung internationalization
* Caranya saat membuat formatter, kita perlu menambahkan Locale



## **Legacy Date dan Time**

* Saat fitur Java Date & Time API baru keluar, Java juga menambahkan integrasi dengan fitur legacy date dan time yng menggunakan Date, Calendar dan TimeZone
* Kita bisa melakukan konversi dari tipe temporal legacy ke tipe temporal baru ataupun sebaliknya

**Konversi Legacy ke Baru**



**Konversi New ke Legacy**

