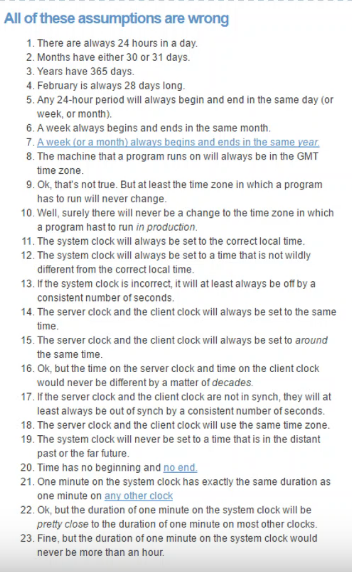
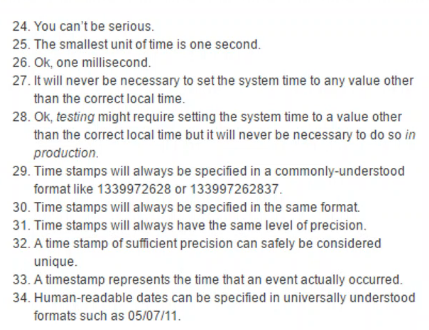
TECNICHE DI PROGRAMMAZIONE

15/04/19

GESTIONE DELLE DATA IN JAVA E IN SQL.

In questa lezione avremo a che fare con dati di tipo temporale (date, ore, ecc…). Lavorare con questi tipi di dati è una delle cose più difficili nella programmazione. Per risolvere questo tipo di problema, java mette a disposizione delle librerie che però o sono troppo semplici o sono fin troppo complicate e macchinose. I problemi dovuti alla gestione dei dati di tipo temporali, sono causati dalle seguenti assunzioni che a primo impatto possono sembrare vere, ma in realtà non lo sono.





Ci sono però dei modi per rappresentare il tempo in modo tale che sia uguale per tutti. Per esempio, se indichiamo determinati istanti di tempo, essi saranno uguali in qualsiasi parte del mondo; se dico “adesso” o “il giorno in cui sono nato” o “il giorno della scoperta dell’America” ecc…

Qui noi vogliamo rappresentare:

* Determinati istanti di tempo (Es: ora, il momento in cui l’uomo è atterrato sulla luna).
* Giorni (senza specificare il tempo. Es: oggi, il giorno in cui sono nato).
* Orari (senza specificare i giorni. Es: orario di lavoro dalle 9 alle 17).
* Date che si ripetono (senza specificare l’anno. Es: anniversario di matrimonio, giorno di Natale).
* Intervalli di tempo (Es: una settimana, 30 giorni lavorativi).
* Date relative (Es: prossimo giovedì).

Cosa prima cosa dobbiamo ben distinguere la differenza tra tempo **UMANO** e tempo **MACCHINA**.

Il primo è un tempo semplificato, lineare ed è quello che viene effettivamente misurato e gestito dal calcolatore. Viene definito un istante “0” e da li incominciamo a contare (in millisecondi, nanosecondi, secondi ecc…). Quindi esso è un numero intero che si incrementa in modo assoluto, a partire da una determinata data di riferimento arbitraria.

Il secondo invece si contraddistingue dal primo in quanto ci sono tanti circoli ricorrenti che si ripetono ogni volta.

Ci sono diversi metodi che ci consentono di operare con il tempo macchina:

* Parsing: converte una stringa il un oggetto di tipo data
* Formatting: opposto di Parsing
* Building: dato una “data” (es 02/08/17), la trasforma in tempo macchina.
* Analyzing: dato un determinato istante di tempo, ci permette di estrarne le componenti.
* Arithmetic: somma e sottrae una quantità dalle date.

Sulla Terra, per convenzione universale, viene misurato sempre rispetto ad un riferimento planetario, l’UTC. Esso è un fuso orario che non cambia mai di riferimento su tutto il Pianeta. Quindi uno dei nostri obiettivi e cercare di misurare sempre la data in UTC, però al tempo stesso non dobbiamo costringere l’utente a lavorare con un tempo che non è il suo. Quindi tutti gli elementi dell’interfaccia devono tenere conto dell’ora legale.

Quindi ogni volta che ci viene data una data dobbiamo sempre chiederci se questa è una data assoluta o una data locale rispetto ad un determinato fuso orario. Se così fosse dobbiamo sapere qual è il fuso orario.

Fino agli inizi degli anni 60, non c’era un ora legale comune, ogni città aveva il suo “orologio”.

Solo grazie alle linee ferroviarie si è capita la necessità di avere un orologio comune.

C’è una norma (ISO 8601) che ci definisce il modo in cui scrivere le date.

Un esempio di data scritta con questa norma è: 2017-05-28.

Per rappresentare data e ora insieme con lo standard UTC invece usiamo la seguente sintassi:

2017-05-28T16:37:18+00:00

Questa rappresentazione non è l’unica in quanto come abbiamo detto in precedenza, l’utente ha bisgonao di una rappresentazione più vicina alla sua localizzazione.

Java dalla versione 1 alla versione 7 ha sempre avuto delle classi che permettevano di gestire le date: java.util.Date e java.util.Calendar. Queste classi però fanno schifo in termini di implementazione e funzionalità.

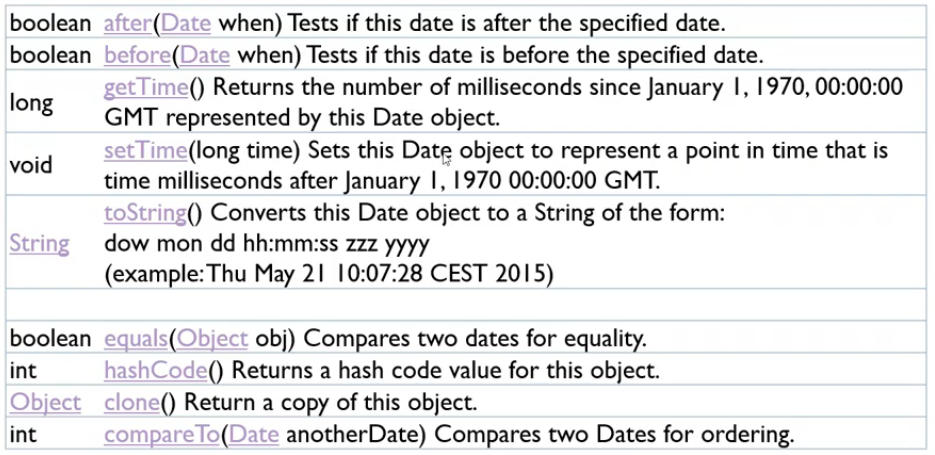
Dalla versione 8 java ha introdotto un nuovo package “java.time” cha ha un meccanismo di funzionamento più pratico e più logico.

Ma perché quelle di prima non andavano bene?

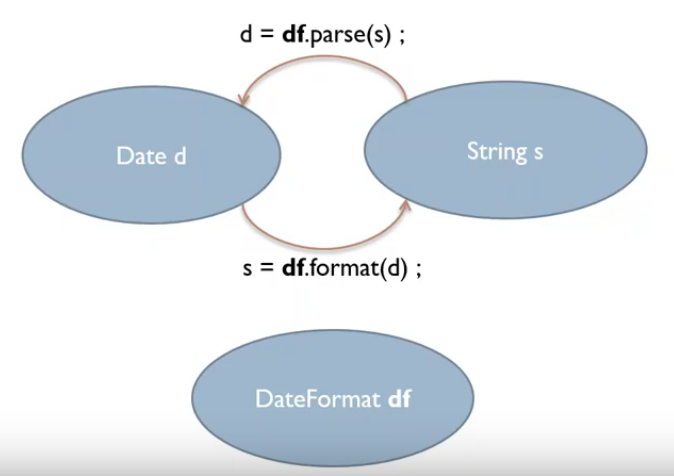
L’oggetto Date (la prima) rappresenta un “long” che è il tempo macchina, cioè il numero di millisecondi che sono passati dal 1° gennaio 1970. Questo ogetto rappresenta si auna data che un’ora. Inoltre, la maggior parte dei metodi di questo oggetto sono deprecati (non usati più).

Ci sono però ancora degli elementi che sopravvivono.

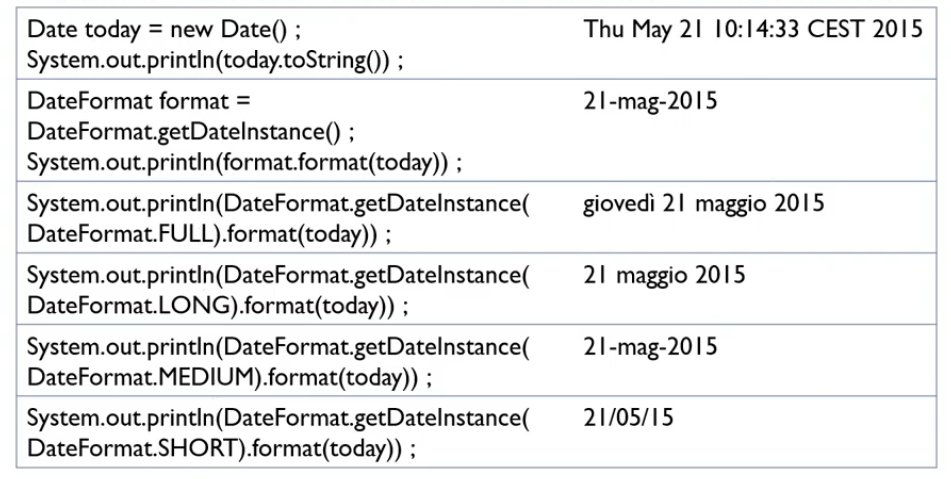
C’è un costruttore “Date ()” che crea un oggetto Date sulla base dell’istante attuale. Posso anche inserire un “long” dentro al costruttore e lui crea un oggetto Date che contiene quel numero long (millisecondi passati dal 1° gennaio 1970). Ci sono anche dei metodi che sono ancora non deprecati e sono i seguenti:



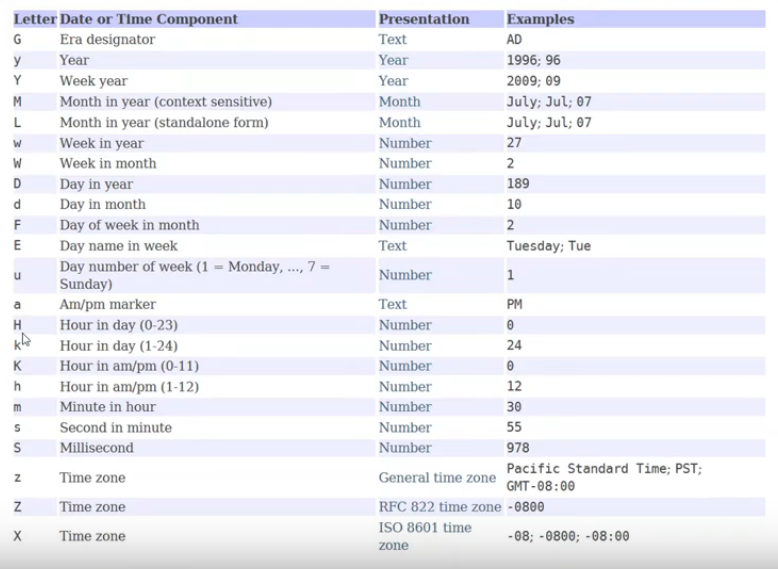
Inoltre abbiamo una classe che si chiama “java.text.DateFormat” che ha varie sotto classi e algoritmi di formattazione. Essa permette di trasformare una stringa in un oggetto Date e viceversa.

 Classe Format e il suo funzionamento.

Alcuni esempi per capire meglio quello detto fino ad adesso:



Ci sono inoltre una serie di patterns che permettono di visualizzare la data e l’ora in determinati modi:



I metodi e le classi elencate fino ad adesso mi permettono di leggere e scrivere una data e basta, non posso farci nessuna operazione (per esempio non posso chiedere che giorno è domani).

Per fare delle operazioni devo usare la classe “Calendar”. Essa ci permette di rappresentare il tempo attraverso una serie di campi flessibili, modificabili. **È una classe astratta quindi non posso fare “new” ma “getIstance”.** Così facendo otterrò un oggetto calendario impostato sull’istante attuale e poi se voglio impostare un valore che non sia quello attuale allora dovrò usare il metodo “setTime” che riceve un oggetto di tipo Date. Per stampare e leggere delle date, la classe “Calendar” non ha nessun metodo. Allora sono costretto a passare avanti e indietro tra Calendar e Date mediante dei “getTime” e “setTime”.

Oggi abbiamo il package “java.time” il quel è più ordinato. Tutti gli oggetti di questa libreria sono immutabili, quindi non esiste nessun metodo per modificare la data. Quindi, devo sempre creare nuovi oggetti e, la loro creazione avviene sempre con metodi di “Factory” (non si creeranno mai oggetti con New ma con una serie di metodi).

Il metodo più frequentemente usato è il metodo “of” che crea un oggetto usando un’informazione che gli passo. Inoltre, abbiamo oggetti separati per rappresentare date e per rappresentare date e ore messe insieme. Alla fin fine quindi questo package contiene le seguenti classi:

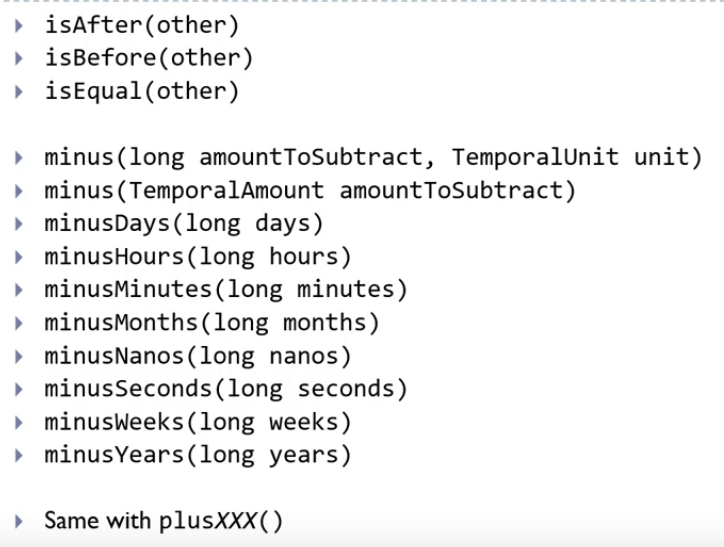


La classe “Istant”, per esempio, rappresenta il tempo solamente in secondi, quindi rappresenta il tempo macchina. Tutte le altre classi invece rappresentano il tempo umano.   
La classe “LocalDate” ci permette di rappresentare una data (anno,mese,giorno).

Se ci interessa anche l’ora usiamo invece la classe “LocalDateTime”. Se invece vogliamo anche specificare in quale fuso orario il nostro evento avviene, allora userò la classe “ZonedDateTime”. E così via…

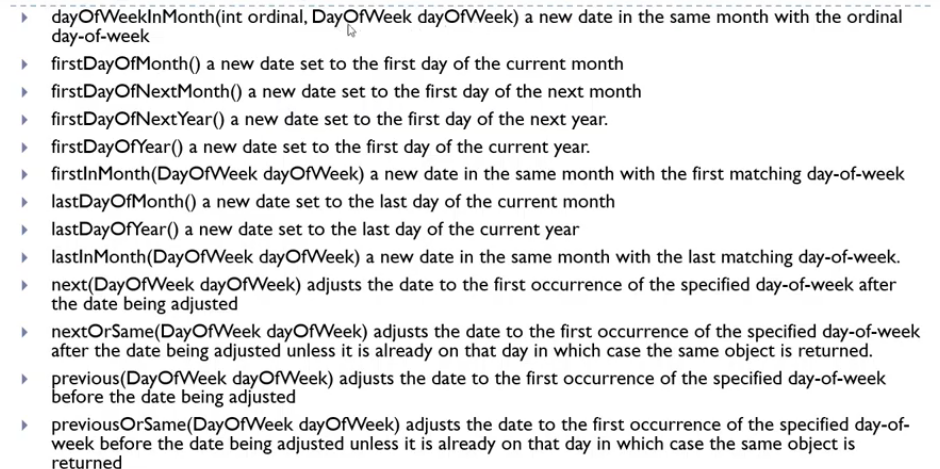
Per costruire nuovi oggetti ho 3 metodi di factory. Uno di questi è “parse” che, data una stringa, costruisce l’oggetto Date o Time o DateTime. Gli altri due sono “of” e “from”. La differenza è che il primo prendere dei valori esistenti e li mette nell’oggetto, mentre il secondo fa delle conversioni (per esempio da Date a DateTime).

Un altro metodo molto importante è il “with” che dato un oggetto, mi restituisce una copia di questo ma con un campo modificato. E’ molto simile ad un Setter nel mondo delle classi immutabili. Quindi, questo metodo prende una data e la modifica in qualche modo. Questa modifica avviene usando vari tipi di criteri di modifica, attraverso una famiglia di classi di tipo “TemporalAdjuster”.



I metodi in figura ci permettono di eseguire operazioni con le date. MAI fare l’aritmetica con le date per conto nostro. Per fare la data di domani, per esempio, farò la data di oggi “plusDays (1)”.

Se somma e sottrazione non ci bastano ci sono i seguenti metodi, i quali derivano dalla classe “TemporalAdjuster”.

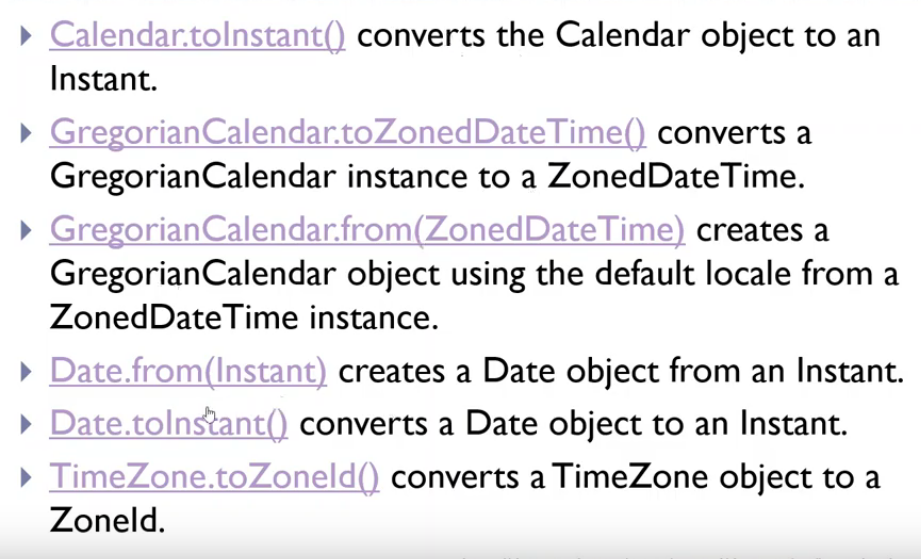


Ci sono ancora altre 2 classi che rappresentano un aspetto diverso del tempo. Abbiamo la classe “Duration” che come appunto ci suggerisce il nome, ci indica una durata (es: il compito dura 2 ore). In questo caso quindi abbiamo la differenza tra due istanti.

Nel caso in cui avessimo la differenza fra 2 “LocalDate[Time]”, usiamo la classe “Period”. Quindi nel caso in cui non ci limitiamo al tempo, ma mettiamo in mezzo anche le date (es: il lavoro dura 4 settimane e 9 giorni) utilizziamo questa classe.

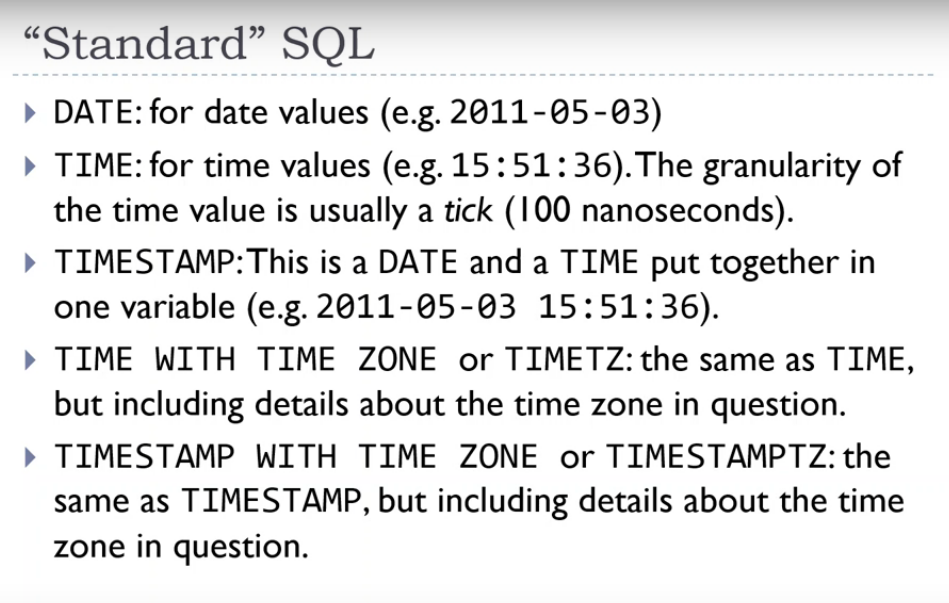
Tutte e due le classi ci rappresentano quindi la differenza tra 2 cose. Con la classe “Duration” otterremo la quantità di tempo tra due intervalli descritti in tempo Macchina, mentre con la classe “Period” otterremo la quantità di tempo tra due intervalli descritti in tempo Umano.

La classe “java.time” ha introdotto anche dei metodi per trasformare gli oggetti di tipo “Calendar” o “Date” in oggetti di tipo “Istant” o “ZonedDateTime”. Notare come gli oggetti di tipo “Calendar” vengano sempre trasformati in oggetti di tipo “ZonedDateTime” mentre gli oggetti di tipo “Date” vengano sempre trasformati in oggetti di tipo “Isntant”.

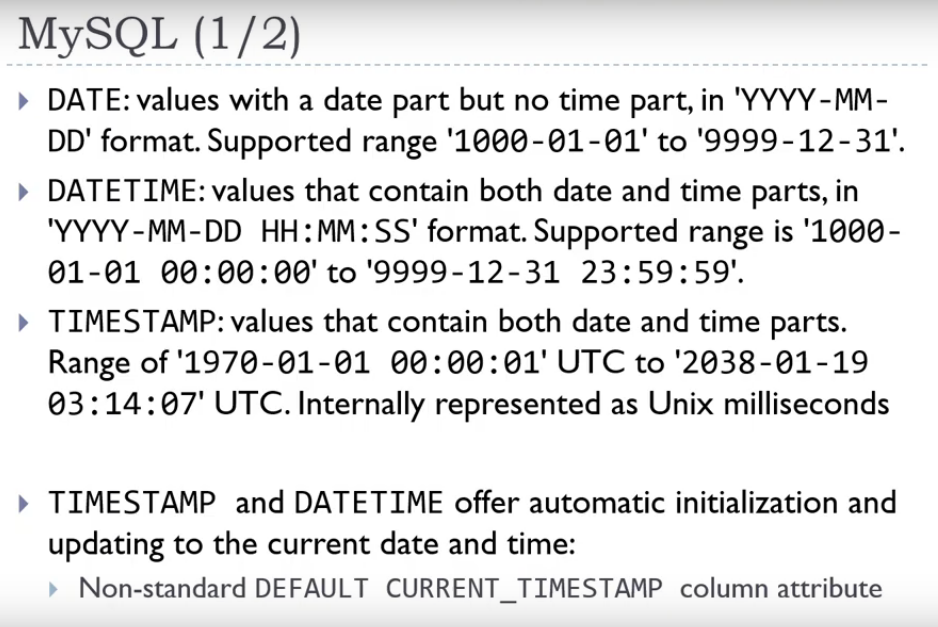


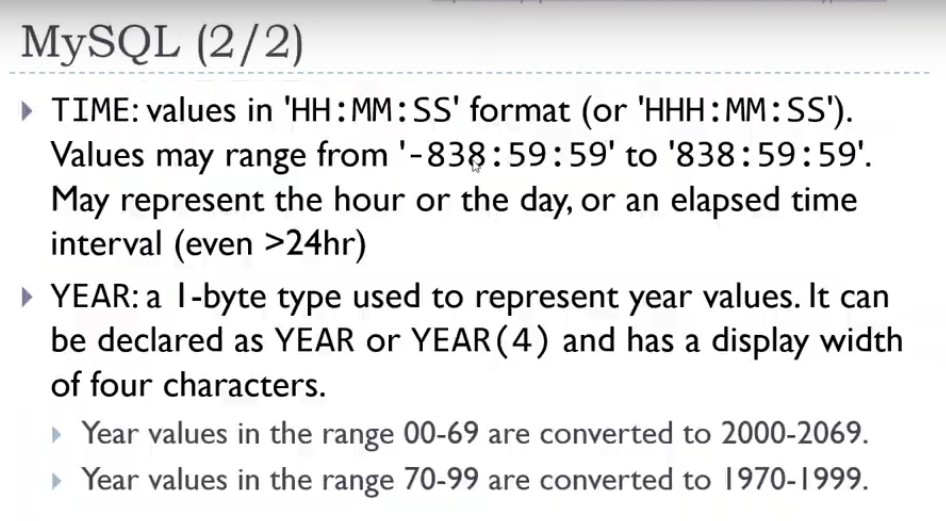
Ora abbiamo capito come gestire le date in Java, ma come dobbiamo fare nel caso in cui ci trovassimo nello standard SQL? E nel caso in cui fossimo in MySQL? Come vengono convertite le date e i tempi in JDBC?

Per lo standard SQL abbiamo 5 colonne temporali:



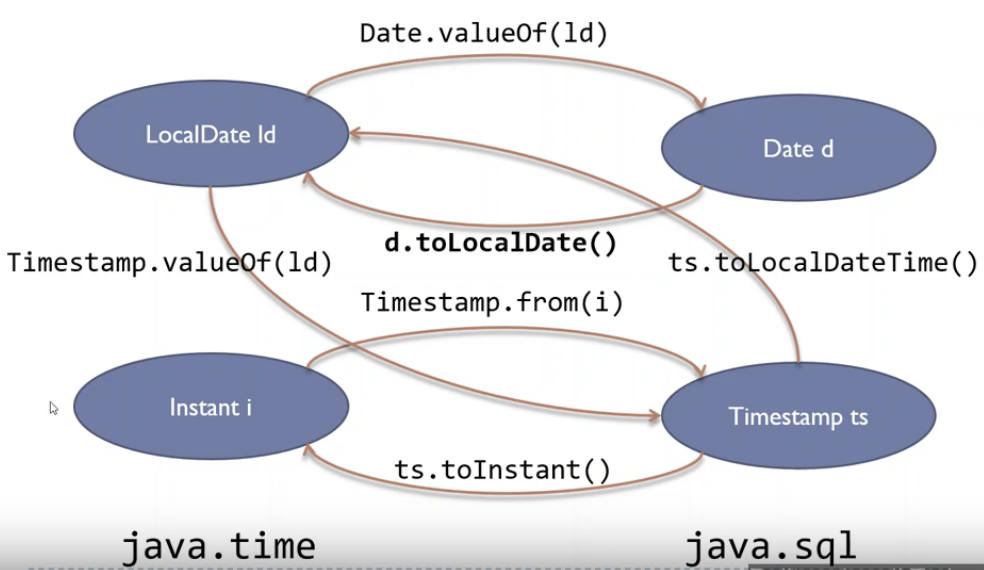
Per mySQL invece avremo i seguenti campi (niente da spiegare):





Per JDBC vedere slide apposite molto chiare.

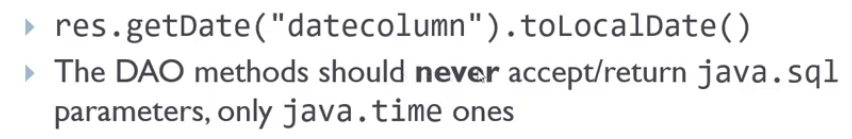
La seguente immagine invece riassume tutti i passaggi possibili tra i tipi di data “Date” e “Timestamp” di Java SQL e i tipi di data “LocalDate” e “Istant” di java.time :



Questo lavoro lo facciamo fare al DAO. Quindi nei metodi del DAO avrò oggetti di tipo java.time.

Suggerimenti:

* Evitiamo sempre java.Date e java.Calendar-
* Usiamo sempre le classi java.time
* In SQL facciamo sempre attenzione alle colonne (se sono Data o DateTime o solo Time).
* Quando leggiamo i dati dal Database, convertire sempre in classi java.time.
* I calcoli facciamoli nel Database e non in java.



Metodo per convertire.

**MAI trattare le date come stringhe o numeri interi (vedi errori sulle slide).**

* I