Data e Ora in Java

- Calendar e GregorianCalendar
- Controllo dell'input di data e ora
  - Comparazione di date e ore

Calendar

- In Java il tempo (data e ora) viene gestito tramite le classi `java.util.Calendar` e `java.util.GregorianCalendar`
- `Calendar` è una classe astratta e `GregorianCalendar` è una sottoclasse di `Calendar` che implementa tutti i metodi astratti di `Calendar` e aggiunge alcune particolarità del calendario Gregoriano
- L'uso della classe `java.util.Date` è deprecato fin dalla versione 1.1 della sdk, pertanto è sconsigliato

GregorianCalendar

- Un oggetto `GregorianCalendar` rappresenta un determinato istante nel tempo
- L'asse temporale è discreto e la distanza tra due punti consecutivi è di un millesimo di secondo (millisecondo)
- Lo zero dell'asse temporale è il primo gennaio 1970 a mezzanotte precisa, ora di Greenwich

GregorianCalendar

- Nello stato di un oggetto `GregorianCalendar` è presente un numero, positivo o negativo, corrispondente al numero di millisecondi su tale asse temporale
- Il calendario Gregoriano ha subìto un cambiamento di data il 4 ottobre 1582
- Per riallineare il calendario furono saltati 10 giorni e introdotte nuove regole più precise per gli anni bisestili

GregorianCalendar

- Tutti i cambiamenti e tutte le regole, spesso complicate, sono gestite dagli oggetti `GregorianCalendar`
- Lo scorrimento del tempo nell'oggetto tiene conto di tutto in modo che chi lo usa non si deve preoccupare di fare controlli
- Ciò rappresenta un grande vantaggio
- Vediamo ora come usare questi oggetti

Creazione

- Ci sono diversi costruttori per la classe
- Il costruttore senza parametri crea un `GregorianCalendar` che punta al millisecondo indicato dall'orologio di sistema al momento della creazione dell'oggetto
- Gli altri costruttori permettono di specificare la data, la data e l'ora o la data e l'ora precisa fino al millisecondo
- Attenzione ai numeri da passare....

Creazione

- Bisogna ricordare che il numero dei mesi, a differenza di quello che si fa di solito, parte da 0
- Quindi Gennaio è 0, Febbraio è 1 ecc.
- Può risultare comodo utilizzare le costanti pubbliche fornite dalla classe `Calendar`
- `Calendar.JANUARY`, `Calendar.FEBRUARY`, ecc.

Creazione

- Le ore sono da 0 a 23 e le ore 0:00:00:000 sono il primo millisecondo del giorno successivo a quello delle 23:59:59:999
- Benché sia impreciso, per motivi storici se si usa la visualizzazione dell'ora da 0 a 12 si ha che:
  - la mezzanotte può essere indicata come le 12:00 “am” < 12:01 “am” (mezzanotte e 1 minuto dello stesso giorno)
  - il mezzogiorno è indicato come le 12:00 “pm”

get/set

- Per accedere a un oggetto `GregorianCalendar` si usa una interfaccia con metodi get/set
- L'oggetto mette a disposizione un metodo `int get(int)` in cui il parametro da passare è un intero che corrisponde al tipo di informazione che si vuole avere
- In `Calendar` sono definite le costanti intere per tutte le informazioni che si possono ottenere sul millisecondo rappresentato dall'oggetto

get/set

- Ad esempio se si vuole ottenere il giorno del mese si usa la costante Calendar.DAY_OF_MONTH

```
GregorianCalendar cal = new
  GregorianCalendar(2000, 1,
  20,9,00);
int giorno =
  cal.get(Calendar.DAY_OF_MONTH);
// giorno vale 20 (il 20 di febbraio)
```


get/set

- Se si vuole sapere il giorno della settimana:

```
int giornoSett=
  cal.get(Calendar.DAY_OF_WEEK);
```

- `giornoSett` vale 1, cioè domenica, poiché il 20 di Febbraio del 2000 è domenica
- Le costanti: `Calendar.SUNDAY`=1, `Calendar.MONDAY`=2, ..., `Calendar.SATURDAY` = 7

get/set

- Il metodo `void set(int, int)` vuole come primo intero la costante del campo da modificare e come secondo intero il nuovo valore per la costante

```
cal.set(Calendar.YEAR, 2001);
giornoSett =
  cal.get(Calendar.DAY_OF_WEEK);
```

- `giornoSett` vale 3, cioè martedì, poiché il 20 febbraio 2001 è martedì

Costanti Calendar

- Consultare le API di `java.util.Calendar`
- Per conoscere tutti i campi del calendario leggibili e modificabili, e quindi le relative costanti intere
- Per esempi e per la formattazione di date a partire da un `GregorianCalendar` consultare il codice allegato `ProvaCalendar.java`

add/roll

- E' possibile modificare un `GregorianCalendar` aumentando di un certo numero un certo campo del calendario
- Ad esempio se voglio aggiungere 45 giorni al 20 febbraio 2001:

```
cal.add(Calendar.DAY_OF_MONTH,45);
```

- cal ora indicherà il 6 Aprile 2001, ore 9:00:000, esattamente 45 giorni dopo

add/roll

- I campi giorno, mese e anno vengono aggiornati consistentemente
- I campi ora, minuti, secondi e millisecondi invece non vengono toccati
- Si pensi al funzionamento di un contagiri: se si fa girare una rotella tutte le rotelle a sinistra cambiano di conseguenza, mentre quelle a destra rimangono invariate

Controllo dell'input di date

- Ogni oggetto `GregorianCalendar` ha un flag lenient/non-lenient
- Alla creazione il flag viene messo a `true`
- In questo stato se un campo del calendario viene assegnato con un valore fuori campo l'oggetto si aggiusta di conseguenza

```
cal.set(Calendar.DAY_OF_MONTH, 32);
```

- Il 32 aprile 2001 viene interpretato come il 2 maggio 2001

Controllo dell'input di date

- Ponendo a false il flag lenient la modifica con un valore fuori dal campo comporta il lancio di una eccezione `IllegalArgumentException`
- Si noti che l'eccezione non viene lanciata dal metodo `set` con il valore fuori campo, ma dalla prima chiamata al metodo `get`, di qualunque campo, successiva al `set` incriminato
- Questo meccanismo può essere usato per controllare l'input delle date

Controllo dell'input di date

- Per controllare se una data fornita in input è ben specificata, cioè il giorno, mese e anno indicati sono esistenti, si può
- Porre un `GregorianCalendar` a non-lenient
- Eseguire dei `set` con i valori che si hanno in input
- Eseguire un `get` che lancerà l'eccezione se i valori non sono corretti
- Si consulti il codice allegato InputDate.java

Confronto di date

- La classe `Calendar` offre un metodo `compareTo` che funziona nel solito modo

```
GregorianCalendar cal1 = new
  GregorianCalendar(2001, 4, 10);
int comp = cal1.compareTo(cal);
```

- Poiché il 10 maggio 2001 è successivo al 2 maggio 2001 `comp` avrà valore > 0
- Si consultino le API di `Calendar`

Confronto tra date

- Il metodo `compareTo` è preciso fino al millisecondo
- In pratica vengono testati il numero di millisecondi rappresentati da ogni oggetto per stabilire la loro posizione relativa prima/dopo