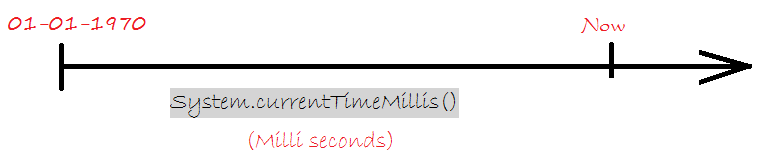
1- Các lớp Date, time, calendar trong Java

Java cung cấp một số class liên quan tới thời gian và lịch (Calendar), dưới đây là danh sách các class này:

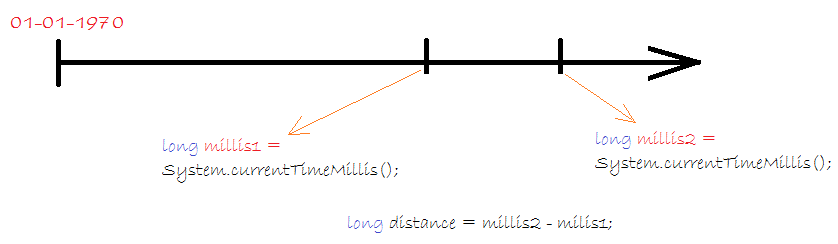
|  |  |
| --- | --- |
| Class | Mô tả |
| java.util.Date | Một lớp đại diện cho ngày tháng năm và thời gian. Tiếc là hầu hết các phương thức của nó đã bị lỗi thời, khuyến cáo là không nên sử dụng các phương thức đó, tuy nhiên lớp Date vẫn được sử dụng rỗng rãi. |
| java.util.concurrent.TimeUnit | TimeUnit là một Enum mô tả các đơn vị ngày tháng năm và thời gian. |
| java.sql.Date | Một lớp mô tả ngày tháng năm. Thông tin về thời gian bị cắt bỏ. Lớp này thường sử dụng trong JDBC. |
| java.sql.Time | Một lớp mô tả thời gian (Giờ phút giây, milli giây), và không chứa thông tin ngày tháng năm.Lớp này thường sử dụng trong JDBC. |
| java.sql.Timestamp | Một lớp mô tả ngày tháng năm và thời gian. Lớp này thường sử dụng trong JDBC. |
| java.util.Calendar | Lớp mô tả bộ Lịch. Nó có các phương thức toán học về thời gian, chẳng hạn thêm ngày, bớt ngày,... |
| java.util.GregorianCalendar | Là một lớp con trực tiếp của java.util.Calendar, mô tả ngày Dương Lịch, bộ lịch được sử dụng rộng rãi trên thế giới ngày nay. Nó có tất cả các method từ java.util.Calendar để thao tác toán học trên ngày tháng năm và thời gian. |
| java.util.TimeZone | TimeZone là lớp mô tả múi giờ, nó có ích khi bạn làm việc với Lịch trên múi giờ. |
| java.text.SimpleDateFormat | Lớp này giúp bạn chuyển một String có định dạng ngày tháng sang kiểu Date và ngược lại |

2- System.currentTimeMillis()

*currentTimeMillis()* là một method tĩnh của class System. Nó trả về khoảng thời gian bằng mili giây tính từ ngày 1-1-1970 cho tới thời điểm hiện tại.



*System.currentTimeMillis()* thường được sử dụng để đo khoảng thời gian làm một việc gì đó bằng cách gọi method này trước khi bắt đầu công việc, và sau khi hoàn thành công việc.



JobTimeDemo.java

**package** org.o7planning.tutorial.dt;

**public** **class** **JobTimeDemo** {

// Đây là phương thức tính tổng các số từ 1 tới 100.

**private** **static** int **sum**() {

int sum = 0;

**for** (int i = 0; i <= 100; i++) {

sum += i;

}

**return** sum;

}

**private** **static** **void** **doJob**(int count) {

// Gọi phương thức 'sum' với số lần cho bởi tham số.

**for** (int i = 0; i < count; i++) {

sum();

}

}

**public** **static** **void** **main**(String[] args) {

long millis1 = System.currentTimeMillis();

doJob(10000);

long millis2 = System.currentTimeMillis();

long distance = millis2 - millis1;

System.out.println("Distance time in milli second: " + distance);

}

}

Kết quả chạy ví dụ:

Distance time in milli second: 3

3- TimeUnit

TimeUnit là một Enum, nó được đưa vào từ Java5. Nó có một số method chuyển đổi giữa các đơn vị thời gian, và thực sự có ích trong một số tình huống.

// Số phút.

int minute = 5;

// Chuyển số phút trên ra mili giây.

// Đây là cách truyền thống có thể bạn từng sử dụng.

int millisecond = minute \* 60 \* 1000;

// Sử dụng TimeUnit:

long millisecond = TimeUnit.MINUTES.toMillis(minute);

Một số method của TimeUnit

// Đổi sang nano giây (nanoseconds).

**public** long **toNanos**(long d);

// Đổi sang micro giây (microseconds).

**public** long **toMicros**(long d);

// Đổi sang mili giây (milliseconds)

**public** long **toMillis**(long d);

// Đổi sang giây (seconds).

**public** long **toSeconds**(long d);

// Đổi sang phút (minutes).

**public** long **toMinutes**(long d);

// Đổi sang giờ (hours).

**public** long **toHours**(long d);

// Đổi sang ngày (days).

**public** long **toDays**(long d) ;

// Chuyển đổi sang đơn vị thời gian khác, cho bởi tham số 'u'.

**public** long **convert**(long d, TimeUnit u);

Ví dụ:

TimeUnitConvertDemo.java

**package** org.o7planning.tutorial.timeunit;

**import** java.util.concurrent.TimeUnit;

**public** **class** **TimeUnitConvertDemo** {

**public** **static** **void** **main**(String[] args) {

// Số giây.

long second = 125553;

// Đổi số giây trên sang phút (Minute)

long minute = TimeUnit.MINUTES.convert(second, TimeUnit.SECONDS);

System.out.println("Minute " + minute);

// Đổi số giây trên sang giờ.

long hour = TimeUnit.HOURS.convert(second, TimeUnit.SECONDS);

System.out.println("Hour " + hour);

System.out.println("------");

// Chuyển đổi 3 ngày sang phút.

minute = TimeUnit.DAYS.toMinutes(3);

System.out.println("Minute " + minute);

// Chuyển đổi 3 ngày sang giờ.

hour = TimeUnit.DAYS.toHours(3);

System.out.println("Hour " + hour);

}

}

Kết quả chạy ví dụ:

Minute 2092

Hour 34

------

Minute 4320

Hour 72

4- java.util.Date

java.util.Date là một trong các class mô tả ngày tháng đầu tiên trong Java. Thật đáng tiếc là hầu hết các phương thức của nó đã lỗi thời, và thay vào đó là sử dụng các phương thức của java.util.Calendar. Nhưng bạn vẫn có thể sử dụng java.util.Date để mô tả ngày tháng.

Chỉ còn 2 Constructor có thể sử dụng:

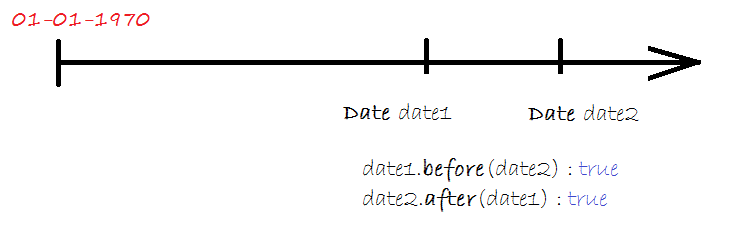
// Tạo một đối tượng Date mô tả thời điểm hiện tại.

Date date1 = **new** **Date**();

// Tạo một đối tượng Date theo thời gian là milli giây tính từ 1-1-1970.

long millis = .....;

Date date2 = **new** **Date**(millis);



DateDemo.java

**package** org.o7planning.tutorial.date;

**import** java.util.Date;

**import** java.util.concurrent.TimeUnit;

**public** **class** **DateDemo** {

**public** **static** **void** **main**(String[] args) **throws** InterruptedException {

// Tạo một đối tượng Date mô tả thời gian hiện tại.

Date date1 = **new** **Date**();

// Ngừng 1 khoảng thời gian 3 giây.

Thread.sleep(TimeUnit.SECONDS.toMillis(3));

// Số milli giây từ 01-01-1970 tới hiện tại.

long millis = System.currentTimeMillis();

Date date2 = **new** **Date**(millis);

// So sánh 2 đối tượng date1 và date2.

// i < 0 nghĩa là date1 < date2

// i = 0 nghĩa là date1 = date2

// i > 0 nghĩa là date1 > date2

int i = date1.compareTo(date2);

System.out.println("date1 compareTo date2 = " + i);

// Kiểm tra xem date1 có đứng trước date2 không.

boolean before = date1.before(date2);

System.out.println("date1 before date2 ? " + before);

// Kiểm tra xem date1 có đứng sau date2 không.

boolean after = date1.after(date2);

System.out.println("date1 after date2 ? " + after);

}

}

Kết quả chạy ví dụ:

date1 compareTo date2 = -1

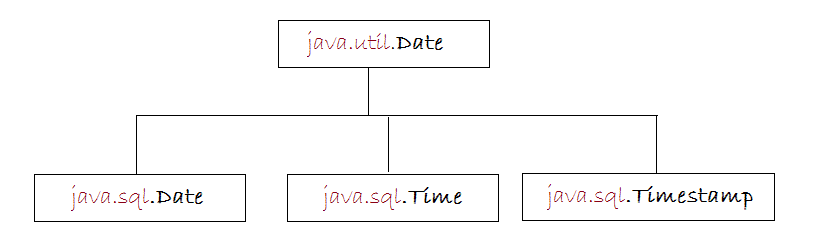
date1 before date2 ? true

date1 after date2 ? false

5- Date, Time, Timestamp (java.sql)

java.sql có 3 class liên quan tới ngày tháng và thời gian:

* java.sql.Date
* java.sql.Time
* java.sql.Timestamp



Cụ thể:

* java.sql.Date tương ứng với SQL DATE, nghĩa là nó chỉ lưu trữ năm, tháng, ngày và bỏ qua giờ phút giây, và milli giây. Thêm nữa java.sql.Date không chứa thông tin về múi giờ.
* java.sql.Time tương ứng với SQL TIME và chỉ chứa các thông tin về giờ, phút, giây, milli giây.
* java.sql.Timestamp tương ứng với SQL TIMESTAMP, nó chính xác tới nano giây (chú ý java.util.Date chỉ hỗ trợ tới milli giây!) với độ chính xác tùy biến.

Các class trên tham gia vào trong *PreparedStatement*trong JDBC API, chẳng hạn các method *setDate*, *setTime*, *setTimestamp*. Hoặc có thể lấy ra từ *ResultSet*.

6- java.util.Calendar

Sơ lược về các bộ lịch:

Gregorian Calendar: Đây chính là Dương Lịch, còn gọi lịch Thiên chúa giáo, là lịch quốc tế. Nó được sử dụng rộng rãi nhất được đặt tên theo Đức Giáo Hoàng Gregory XIII, người đã giới thiệu nó vào năm 1582.

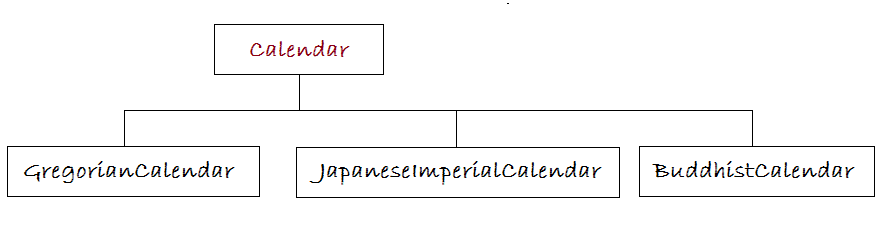
* [http://en.wikipedia.org/wiki/Gregorian\_calendar](https://en.wikipedia.org/wiki/Gregorian_calendar)

Buddhist Calendar: Đây là một bộ lịch phật giáo, thường được sử dụng tại một số nước Đông Nam Á trước kia như Thái Lan, Lào, Campuchia, cũng như Sri Lanka. Hiện nay lịch này được sử dụng trong các lễ hội phật giáo. Và không còn quốc gia nào sử dụng lịch này một cách chính thức, các quốc gia này đã đổi sang sử dụng Gregorian Calendar. Bạn có thể tham khảo thêm thông tin về lịch này tại:

* [http://en.wikipedia.org/wiki/Buddhist\_calendar](https://en.wikipedia.org/wiki/Buddhist_calendar)

Japanese Imperial Calendar: Đây là bộ lịch truyền thống của Nhật bản, hiện nay Nhật bản đã chuyển sang sử dụng dương lịch (Gregorian Calendar), tuy nhiên bộ lịch truyền thống vẫn được sử dụng một cách không chính thức.

Calendar là class mô phỏng một hệ thống Lịch.



Calendar có một vài class con:

* GregorianCalendar
* JapaneseImperialCalendar
* BuddhistCalendar

Calendar là một class trừu tượng. Nghĩa là bạn không thể khởi tạo nó từ cấu tử (Constructor). Tuy nhiên có 2 method tĩnh để tạo ra đối tượng Calendar.

**public** **static** Calendar **getInstance**();

**public** **static** Calendar **getInstance**(TimeZone zone);

**public** **static** Calendar **getInstance**(Locale aLocale);

**public** **static** Calendar **getInstance**(TimeZone zone,Locale aLocale);

Ví dụ:

// Tạo đối tượng Calendar mô tả thời điểm hiện tại.

// Với Locale mặc định, và TimeZone (múi giờ) mặc định

// (Của máy tính đang chạy).

Calendar c = Calendar.getInstance();

Khi bạn sử dụng Calendar.getInstance(TimeZone,Locale) sẽ nhận được trả về là một trong các class con nói trên. Mà hầu hết là trả về GregorianCalendar.  
  
Gọi Calendar.getInstance() trả về đối tượng Calendar với tham số TimeZone theo máy tính của bạn và Locale mặc định.

Hãy xem code của class Calendar (JDK7):

Calendar (JDK7)

/\*\*

\* Gets a calendar using the default time zone and locale. The

\* <code>Calendar</code> returned is based on the current time

\* in the default time zone with the default locale.

\*

\* **@return** a Calendar.

\*/

**public** **static** Calendar **getInstance**()

{

Calendar cal = createCalendar(TimeZone.getDefaultRef(),

Locale.getDefault(Locale.Category.FORMAT));

cal.sharedZone = true;

**return** cal;

}

/\*\*

\* Gets a calendar using the specified time zone and default locale.

\* The <code>Calendar</code> returned is based on the current time

\* in the given time zone with the default locale.

\*

\* **@param** zone the time zone to use

\* **@return** a Calendar.

\*/

**public** **static** Calendar **getInstance**(TimeZone zone)

{

**return** createCalendar(zone, Locale.getDefault(Locale.Category.FORMAT));

}

/\*\*

\* Gets a calendar using the default time zone and specified locale.

\* The <code>Calendar</code> returned is based on the current time

\* in the default time zone with the given locale.

\*

\* **@param** aLocale the locale for the week data

\* **@return** a Calendar.

\*/

**public** **static** Calendar **getInstance**(Locale aLocale)

{

Calendar cal = createCalendar(TimeZone.getDefaultRef(), aLocale);

cal.sharedZone = true;

**return** cal;

}

/\*\*

\* Gets a calendar with the specified time zone and locale.

\* The <code>Calendar</code> returned is based on the current time

\* in the given time zone with the given locale.

\*

\* **@param** zone the time zone to use

\* **@param** aLocale the locale for the week data

\* **@return** a Calendar.

\*/

**public** **static** Calendar **getInstance**(TimeZone zone,

Locale aLocale)

{

**return** createCalendar(zone, aLocale);

}

**private** **static** Calendar **createCalendar**(TimeZone zone,

Locale aLocale)

{

Calendar cal = null;

String caltype = aLocale.getUnicodeLocaleType("ca");

**if** (caltype == null) {

// Calendar type is not specified.

// If the specified locale is a Thai locale,

// returns a BuddhistCalendar instance.

**if** ("th".equals(aLocale.getLanguage())

&& ("TH".equals(aLocale.getCountry()))) {

cal = **new** **BuddhistCalendar**(zone, aLocale);

} **else** {

cal = **new** **GregorianCalendar**(zone, aLocale);

}

} **else** **if** (caltype.equals("japanese")) {

cal = **new** **JapaneseImperialCalendar**(zone, aLocale);

} **else** **if** (caltype.equals("buddhist")) {

cal = **new** **BuddhistCalendar**(zone, aLocale);

} **else** {

// Unsupported calendar type.

// Use Gregorian calendar as a fallback.

cal = **new** **GregorianCalendar**(zone, aLocale);

}

**return** cal;

}

Một số method quan trọng:

|  |  |
| --- | --- |
| Phương thức get(int) | Giá trị trả về |
| get(Calendar.DAY\_OF\_WEEK) | 1 (Calendar.SUNDAY) tới 7 (Calendar.SATURDAY). |
| get(Calendar.YEAR) | Năm (year) |
| get(Calendar.MONTH) | 0 (Calendar.JANUARY) tới 11 (Calendar.DECEMBER). |
| get(Calendar.DAY\_OF\_MONTH) | 1 tới 31 |
| get(Calendar.DATE) | 1 tới 31 |
| get(Calendar.HOUR\_OF\_DAY) | 0 tới 23 |
| get(Calendar.MINUTE) | 0 tới 59 |
| get(Calendar.SECOND) | 0 tới 59 |
| get(Calendar.MILLISECOND) | 0 tới 999 |
| get(Calendar.HOUR) | 0 tới 11, được sử dụng cùng với Calendar.AM\_PM. |
| get(Calendar.AM\_PM) | 0 (Calendar.AM) hoặc 1 (Calendar.PM). |
| get(Calendar.DAY\_OF\_WEEK\_IN\_MONTH) | DAY\_OF\_MONTH 1 tới 7 luôn luôn tương ứng với DAY\_OF\_WEEK\_IN\_MONTH 1;  8 tới 14 tương ứng với DAY\_OF\_WEEK\_IN\_MONTH 2, ... |
| get(Calendar.DAY\_OF\_YEAR) | 1 tới 366 |
| get(Calendar.ZONE\_OFFSET) | Giá trị GMT của múi giờ. |
| get(Calendar.ERA) | Biểu thị AD (GregorianCalendar.AD), BC (GregorianCalendar.BC). |

CalendarFieldsDemo.java

**package** org.o7planning.tutorial.calendar;

**import** java.util.Calendar;

**public** **class** **CalendarFieldsDemo** {

**public** **static** **void** **main**(String[] args) {

// Tạo một đối tượng Calendar (Lịch) mặc định.

// Với time zone (múi giờ) và locale mặc định.

Calendar c = Calendar.getInstance();

int year = c.get(Calendar.YEAR);

// Trả về giá trị từ 0 - 11

int month = c.get(Calendar.MONTH);

int day = c.get(Calendar.DAY\_OF\_MONTH);

int hour = c.get(Calendar.HOUR\_OF\_DAY);

int minute = c.get(Calendar.MINUTE);

int second = c.get(Calendar.SECOND);

int millis = c.get(Calendar.MILLISECOND);

System.out.println("Year: " + year);

System.out.println("Month: " + (month + 1));

System.out.println("Day: " + day);

System.out.println("Hour: " + hour);

System.out.println("Minute: " + minute);

System.out.println("Second: " + second);

System.out.println("Minute: " + minute);

System.out.println("Milli Second: " + millis);

}

}

Kết quả chạy ví dụ:

Year: 2021

Month: 5

Day: 15

Hour: 20

Minute: 34

Second: 52

Minute: 34

Milli Second: 382

Một số method khác của Calendar:

**void** **set**(int calendarField, int value)

**void** **set**(int year, int month, int date)

**void** **set**(int year, int month, int date, int hour, int minute, int second)

// Thêm hoặc trừ một khoảng thời gian trên một trường (field) của Calendar.

// Dựa trên quy tắc của bộ Lịch.

**void** **add**(int field, int amount)

// Cuộn (lên/xuống) một trường (field) của Calendar.

// roll(): Không làm ảnh hưởng tới các trường khác của Calendar.

**void** **roll**(int calendarField, boolean up)

// Cuộn lên (roll up) một trường của Calendar.

// roll(): Không làm ảnh hưởng tới các trường khác của Calendar.

**void** **roll**(int calendarField, int amount):

// Trả về đối tượng Date dựa trên giá trị của Calendar.

Date **getTime**()

**void** **setTime**(Date date)

// Trả về số mili giây của đối tượng Calendar này.

long **getTimeInMills**():

**void** **setTimeInMillis**(long millis)

**void** **setTimeZone**(TimeZone value)

CalendarDemo.java

**package** org.o7planning.tutorial.calendar;

**import** java.util.Calendar;

**public** **class** **CalendarDemo** {

**public** **static** **void** **showCalendar**(Calendar c) {

int year = c.get(Calendar.YEAR);

// Trả về giá trị từ 0 - 11

int month = c.get(Calendar.MONTH);

int day = c.get(Calendar.DAY\_OF\_MONTH);

int hour = c.get(Calendar.HOUR\_OF\_DAY);

int minute = c.get(Calendar.MINUTE);

int second = c.get(Calendar.SECOND);

int millis = c.get(Calendar.MILLISECOND);

System.out.println(" " + year + "-" + (month + 1) + "-" + day //

+ " " + hour + ":" + minute + ":" + second + " " + millis);

}

**public** **static** **void** **main**(String[] args) {

// Tạo đối tượng Calendar, mô tả thời điểm hiện tại.

// Với time zone (múi giờ) và locale mặc định.

Calendar c = Calendar.getInstance();

System.out.println("First calendar info");

showCalendar(c);

// roll(..): Cuộn một trường (field) của Calendar.

// roll(..): Không làm thay đổi các trường khác.

// Ví dụ: Cuộn lên thêm một giờ (boolean up = true)

c.roll(Calendar.HOUR\_OF\_DAY, true);

System.out.println("After roll 1 hour");

showCalendar(c);

// roll(..): Không làm thay đổi các trường khác của Calendar.

// Cuộn xuống một giờ (boolean up = false)

c.roll(Calendar.HOUR\_OF\_DAY, false);

System.out.println("After roll -1 hour");

showCalendar(c);

// add(..): Có thể làm thay đổi các trường khác của Calendar.

// Tăng thêm một giờ (boolean up = true)

c.add(Calendar.HOUR\_OF\_DAY, 1);

System.out.println("After add 1 hour");

showCalendar(c);

// roll(..): Không làm thay đổi các trường khác của Calendar.

// Cuộn xuống 30 ngày.

c.roll(Calendar.DAY\_OF\_MONTH, -30);

System.out.println("After roll -30 day");

showCalendar(c);

// add(..): Có thể làm thay đổi các trường khác của Calendar.

// Thêm 30 ngày.

c.add(Calendar.DAY\_OF\_MONTH, 30);

System.out.println("After add 30 day");

showCalendar(c);

}

}

Kết quả chạy ví dụ:

First calendar info

2021-5-15 20:35:27 395

After roll 1 hour

2021-5-15 21:35:27 395

After roll -1 hour

2021-5-15 20:35:27 395

After add 1 hour

2021-5-15 21:35:27 395

After roll -30 day

2021-5-16 21:35:27 395

After add 30 day

2021-6-15 21:35:27 395

7- Chuyển đổi giữa Date và Calendar

* Date ==> Calendar

Date now = **new** **Date**();

Calendar c = Calendar.getInstance();

c.setTime(now);

* Calendar ==> Date

Calendar c = Calendar.getInstance();

Date date = c.getTime();

CalendarDateConversionDemo.java

**package** org.o7planning.tutorial.calendar;

**import** java.util.Calendar;

**import** java.util.Date;

**import** java.util.concurrent.TimeUnit;

**public** **class** **CalendarDateConversionDemo** {

**public** **static** **void** **showCalendar**(Calendar c) {

int year = c.get(Calendar.YEAR);

// Trả về giá trị từ 0 - 11

int month = c.get(Calendar.MONTH);

int day = c.get(Calendar.DAY\_OF\_MONTH);

int hour = c.get(Calendar.HOUR\_OF\_DAY);

int minute = c.get(Calendar.MINUTE);

int second = c.get(Calendar.SECOND);

int millis = c.get(Calendar.MILLISECOND);

System.out.println(year + "-" + (month + 1) + "-" + day //

+ " " + hour + ":" + minute + ":" + second + " " + millis);

}

**public** **static** **void** **main**(String[] args) {

Calendar c = Calendar.getInstance();

// year, month, day

c.set(2000, 11, 24);

Date date = c.getTime();

System.out.println("Date " + date);

long timeInMillis = System.currentTimeMillis();

// Trừ đi 24 giờ.

timeInMillis -= TimeUnit.HOURS.toMillis(24);

Date date2 = **new** **Date**(timeInMillis);

Calendar c2 = Calendar.getInstance();

c2.setTime(date2);

showCalendar(c2);

}

}

Kết quả chạy ví dụ:

Date Sun Dec 24 20:36:10 KGT 2000

2021-5-14 20:36:10 608

8- DateFormat & SimpleDateFormat

* Date ==> String

Date date = **new** **Date**();

DateFormat df = **new** **SimpleDateFormat**("dd-MM-yyyy HH:mm:ss");

String dateString = df.format(date);

* String ==> Date

String dateString = "23/04/2005 23:11:59";

DateFormat df = **new** **SimpleDateFormat**("dd/MM/yyyy HH:mm:ss");

Date date = df.parse(dateString);

DateFormatDemo.java

**package** org.o7planning.tutorial.dateformat;

**import** java.text.DateFormat;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** **DateFormatDemo** {

**public** **static** **void** **main**(String[] args) **throws** ParseException {

**final** DateFormat df1 = **new** **SimpleDateFormat**("dd/MM/yyyy HH:mm:ss");

String dateString1 = "23/04/2005 23:11:59";

System.out.println("dateString1 = " + dateString1);

// String ==> Date

Date date1 = df1.parse(dateString1);

System.out.println("date1 = " + date1);

**final** DateFormat df2 = **new** **SimpleDateFormat**("dd-MM-yyyy HH:mm:ss");

// Date ==> String.

String dateString2 = df2.format(date1);

System.out.println("dateString2 = " + dateString2);

}

}

Kết quả chạy ví dụ:

dateString1 = 23/04/2005 23:11:59

date1 = Sat Apr 23 23:11:59 KGST 2005

dateString2 = 23-04-2005 23:11:59

8.1- Tùy biến định dạng Date, Time

Tùy biến định dạng Date. Hãy xem một số ví dụ định dạng và kết quả trả về.

|  |  |
| --- | --- |
| **Khuôn mẫu** | **Kết quả** |
| dd.MM.yy | 30.06.09 |
| yyyy.MM.dd G 'at' hh:mm:ss z | 2009.06.30 AD at 08:29:36 PDT |
| EEE, MMM d, ''yy | Tue, Jun 30, '09 |
| h:mm a | 8:29 PM |
| H:mm | 8:29 |
| H:mm:ss:SSS | 8:28:36:249 |
| K:mm a,z | 8:29 AM,PDT |
| yyyy.MMMMM.dd GGG hh:mm aaa | 2009.June.30 AD 08:29 AM |

Các mẫu cú pháp định dạng Date

|  |  |  |  |
| --- | --- | --- | --- |
| **Ký hiệu** | **Ý nghĩa** | **Trả về** | **Example** |
| G | Kỷ nguyên | Text | AD |
| y | Năm | Number | 2009 |
| M | Tháng trong năm | Text & Number | July & 07 |
| d | Ngày trong tháng | Number | 10 |
| h | Giờ am/pm (1-12) | Number | 12 |
| H | Giờ trong ngày (0-23) | Number | 0 |
| m | Phút trong giờ | Number | 30 |
| s | Giây trong phút | Number | 55 |
| S | millisecond | Number | 978 |
| E | Ngày trong tuần | Text | Tuesday |
| D | Ngày trong năm | Number | 189 |
| F | Ngày của tuần trong tháng | Number | 2 (2nd Wed in July) |
| w | Tuần trong năm | Number | 27 |
| W | Tuần trong tháng | Number | 2 |
| a | am/pm | Text | PM |
| k | Giờ trong ngày (1-24) | Number | 24 |
| K | Giờ am/pm (0-11) | Number | 0 |
| z | Múi giờ (time zone) | Text | Pacific Standard Time |
| ' | escape for text | Delimiter | (none) |
| ' | single quote | Literal | ' |