1 Friendly Java Date API

I wrote a hopefully friendly API on top of the java.util.Date and java.util.Calendar classes. The intention is to be able to do a number of things more easily

- adding time spans to a date
- substracting to dates
- comparing date, time or both
- calculating work days and even add filter to take of specific holidays

1.1 Edition

February 2009 by Sebastian Hennebrueder

Library versions

- API version 0.1, February 2009
- Java 1.5 or later

1.2 Download

PDF of this tutorial

http://www.laliluna.de/download/friendly-java-date-api.pdf

Source code

http://www.laliluna.de/download/friendly-java-date-api.zip

1.3 Charges and License

The Friendly Date API is published under the Open Source Robin Hood License. The license details can be found here.

http://www.laliluna.de/open-source-robin-hood-license.html

Depending on your richness you might need to pay charges to use the API.

Poverty limits for the Friendly Date API

If you use the API privately, your average monthly income should not be more then 1500 EUR.

If you are a organization or a company your budget or total sales should not exceed 100.000 EUR.

2 Using the Friendly API

The Date class is the central element. It is a mutable class which encapsulates access to a *java.util.Calendar*.

2.1 Creating dates

```
// the current date and time
    date = new Date();
    date = Date.now();

// the current date and time set to 00:00:00
    date = Date.today();

// defining a timezone different to the one of your computer
    date = new Date(TimeZone.EUROPE_BELFAST);
    date = Date.now(TimeZone.EUROPE_BELFAST);

// creating explicitly 2009, February 12th
   date = new Date(2009, 2, 12);
// creating date using enums
    date = new Date(2009, Months.APR, 12);
// 15:30 on 2009, Feb 12 th
    date = new Date(2009, 2, 12, 15, 30);

// creating (cloning) from another date
    Date other = new Date();
    date = new Date(other);
    date = new Date(GregorianCalendar.getInstance());
    date = new Date(new java.util.Date());
```

2.2 Comparing dates

The API allows to compare for date and time (default), only date or only time. There are methods which follow the behavior of the *Comparable* interface but you may use the *before / after / equal* methods for a better human readable impression.

```
    Date date = Date.now().setHour(14);
    Date later = Date.now().addHours(2);
// true
    date.equalDate(later);
// false
    date.equalDateTime(later);
// false
    date.equalTime(later);

// true
    date.before(later);
// false
    date.after(later);

// false, we have the same date
    date.beforeDate(later);
// true, the time is earlier
```

```
    date.beforeTime(later);
```

2.3 Changing a date or time zone

```
    Date date = Date.now();

// You can add or set values for year, month, day,
// hour, minute, second, millisecond
    date.addHours(5);
    date.addMonths(2);
    date.setMinute(30);

// Just change the time zone
    date.setTimeZone(TimeZone.AMERICA_NEW_YORK);
// Translate the current time to a different time zone
    date.translate(TimeZone.AFRICA_BAMAKO);
```

2.4 Calculating with dates and fractions

A very nice concept is taking from Ruby. Ruby has native support for fractions. You might remember them from school.

```
50 % = ½
one third is 1/3
```

Fractions are calculated from a day perspective. Fractions are not mutable.

```
1/1 = a day
1/24 = an hour
25/24 = a day and an hour
2/1440 = 2 minutes
```

Here is the Java code

```
// 5 hours and 30 minutes
new Fraction(5, 24).plus(30, 60 * 24);

// 2 days, 3 hours and 20 minutes
new Fraction(2,1).plus(3, 24).plus(20, 60*24);
```

The date class has a couple of ready made fractions like

```
// an hour
Fraction anHour = Date.HOUR;
// a minute
Fraction aMinute = Date.MINUTE;
// a second
Fraction aSecond = Date.SECOND;

// five hours
Date.HOUR.times(5);
// 20 minutes
Date.MINUTE.times(20);
// 5 seconds
Date.SECOND.times(5);
```

Fractions can be used to add or remove time from a date.

```
/* Add 5.5 hours to the current date and time */
Fraction _55 = Date.HOUR.times(5).plus(Date.MINUTE.times(30));
Date.now().add(_55);

// substract 600 minutes from the current date
Date.now().substract(Date.MINUTE.times(600));
```

A fraction is the result of a subtraction from two dates as well.

```
// calculatge the difference between two dates
Date now = Date.now();
Date later = new Date(now).addDays(5);
Fraction delta = later.minus(now);
// the difference is 5 days, so the following is true
delta.equals(new Fraction(5,1));
```

2.5 Calculating working days

Working days are normally Monday to Friday. The basic methods doesn't know about holidays and don't want to know it.

In Germany every Bundesland (same as state in other countries) has its own special holidays and even some cities have a special holiday. I don't want to implement this for the whole world.

As a consequence you can define a DayFilter which can filter the working days by your special holiday filter.

```
Date now = Date.now();
Date later = new Date(now).addDays(5);
from.workingDaysBetween(to)

Date now = Date.now();
Date later = new Date(now).addDays(5);
now.workingDaysBetween(later);

// working days with a day filter
DayFilter filter = new MyBadVilbelGermanyHolidayFilter();
now.workingDaysBetween(later, filter);
```

2.6 Printing dates

Printing is very limited. Basically you can only extract the values from the date class and get the month and day names in a short and a long version.

But you can always fall back to Java formatting.

```
Date date = Date.today();
// time
String.format("%02d:%02d:%02d", date.hour(), date.minute(), date.second());
// time in english format
String.format("English: %02d:%02d:%02d %s", date.hour12(), date.minute(),
date.second(),date.am() ? "AM" : "PM");
// printing English names for day and month
String.format("%s %s, %d %d", date.dayShort(), date.monthName(), date.day(),
```

```
date.year());

// fall back to Java formatting
SimpleDateFormat dateFormat = new SimpleDateFormat();
dateFormat.format(Date.today().toJavaDate());
```

I hope you enjoy the API.

Best Regards / Viele Grüße

Sebastian Hennebrueder