Time & Date Representation in Java
Date class

• An object of type Date represents an instance in time
• Part of java.util.* (requires import statement)
• A new Date object is automatically set to the time of its creation, to the millisecond
• For example:
  Date now = new Date();  // object now holds value
                          // of the current millisecond
Converting to String

• Like many other Java standard classes, Date includes a toString() method, which allows us to output the value of a Date object in the form: **Day Mon dd hh:mm:ss TMZ yyyy**
  – Day is a three-letter abbreviation for day of week – e.g. Sat, Sun, etc.
  – Mon is a three-letter abbreviation for month of year – e.g. Jan, Feb, etc.
  – dd is the one- or two-digit date
  – hh:mm:ss is the current time
  – TMZ is the time zone – for example, a Date object created at Kirkwood would have a CDT or CST stamp
  – yyyy is the 4-digit year
SimpleDateFormat class

• The java.text.* package contains a class that allows us to output Date objects using other formats – the SimpleDateFormat class interacts with Date objects much like the DecimalFormat class interacts with floating-point numbers

• A Date object is created, then a SimpleDateFormat object is created to describe its output appearance
public class DemoDate {
    public static void main (String [] args) {
        Date now;
        SimpleDateFormat fmt;
        now = new Date();
        fmt = new SimpleDateFormat ("dd MMM yyyy");
        System.out.println ("Today's date: " +
            fmt.format(now);
    }
}
SimpleDateFormat patterns – a partial list

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;yyyy.MM.dd G 'at' HH:mm:ss z&quot;</td>
<td>2006.02.01 AD at 12:08:56 CST</td>
</tr>
<tr>
<td>&quot;hh 'o'clock' a, zzzz&quot;</td>
<td>12 o'clock PM, Central Standard Time</td>
</tr>
<tr>
<td>&quot;yyyyy.MMMMMM.dd hh:mm aaa&quot;</td>
<td>02006.February.01 12:08 PM</td>
</tr>
<tr>
<td>&quot;EEE, MMM d, &quot;yy&quot;</td>
<td>Wed, Feb 4, '06</td>
</tr>
</tbody>
</table>

More examples: http://java.sun.com/j2se/1.5.0/docs/api/
GregorianCalendar class

• The architects of Java changed their minds about how the Date class should work during the evolution of the language
• Several methods that were previously part of the Date class have since moved to other classes that describe this kind of data, notably the GregorianCalendar class
GregorianCalendar class

• Also from java.util.*
• More friendly than Date because it uses named constants
• Incorporates a Date object, accessible via the `getTime()` method
• Like a Date object, a GregorianCalendar object will have the current time as its value by default; it is relatively easy to create an object using a different moment in time
Example

GregorianCalendar myDay = new GregorianCalendar(1978, 4, 16);

• This creates an object that contains the Date value May 16, 1978 (my 18th birthday)

• Note that the month of May is represented by a 4 rather than a 5; this is because the first month, January, is represented by a 0
Example

• We can use a predefined constant from the Calendar class to represent the month, as follows:
  
  ```java
  GregorianCalendar myDay =
  new GregorianCalendar(1978, Calendar.MAY, 16);
  ```

• Calendar.MAY is another example of a class constant, like Math.PI
Extracting Date information from a GregorianCalendar object

• Using other predefined constants from the Calendar class and the GregorianCalendar object’s get() method, we can extract various pieces of information about the underlying Date value

• The Calendar class constants include Calendar.DAY_OF_WEEK, Calendar.DATE, Calendar.MONTH, Calendar.DAY_OF_YEAR and Calendar.YEAR
Example

GregorianCalendar myDay =
    new GregorianCalendar(1978, 4, 16);
System.out.println((myDay.get(Calendar.MONTH)+1) + "/
    + myDay.get(Calendar.DATE) + "/
    + myDay.get(Calendar.YEAR));

// prints 5/16/1978