本章内容：

·包装类

·Math：random() round()

·日期格式的处理：Date、Calendar、DateFormat

1. 包装类：java中提倡的思想一切皆对象，但是我们发现之前讲的java的8种基本数据类型并不是一个类，也就是说基本数据类的变量不是一个对象，那么和java提倡的思想是相矛盾，为了解决这些矛盾，java中提供了8中基本数据类型的包装类。

|  |  |  |
| --- | --- | --- |
| 编号 | 基本数据类型 | 包装类 |
| 1 | int | Integer |
| 2 | char | Character |
| 3 | double | Double |
| 4 | float | Float |
| 5 | boolean | Boolean |
| 6 | byte | Byte |
| 7 | long | Long |
| 8 | short | Short |

包装类的作用：

·进项装箱和拆箱操作

**·将字符串转换成基本数据类型**

装箱：将基本数据类型转换成包装类类型

拆箱：将包装类类型转换成基本数据类型

范例：进行装箱操作

|  |
| --- |
| **package** com.wanczy.packageDemo;  **public** **class** PackageDemo01 {  **public** **static** **void** main(String[] args) {  **int** i = 100;  Integer in = **new** Integer(i);//进行装箱操作  System.*out*.println(in);  }  } |

范例：进行拆箱操作：

|  |
| --- |
| **package** com.wanczy.packageDemo;  **public** **class** PackageDemo02 {  **public** **static** **void** main(String[] args) {  Integer in = **new** Integer(100);  **int** i = in.intValue();//进行拆箱操作  System.*out*.println(i);    Double dou = **new** Double(100.00001);  **double** d = dou.doubleValue();  System.*out*.println(d);  }  } |

以上的操作都是进行了手动的装箱和拆箱操作，其实在JDK1.5之后，装箱和拆箱操作都可以自动的进行，包装类对象本身是不能进行数学运算，但是经过拆箱操作之后可以进行数学运算，那么JDK1.5之后的包装类对象也可以进行数学运算了。

范例：自动进行装箱操作

|  |
| --- |
| **package** com.wanczy.packageDemo;  **public** **class** PackageDemo03 {  **public** **static** **void** main(String[] args) {  **int** i = 100;  Integer in = i;//进行自动装箱操作  System.*out*.println(in);  }  } |

范例：自动拆箱操作

|  |
| --- |
| **package** com.wanczy.packageDemo;  **public** **class** PackageDemo04 {  **public** **static** **void** main(String[] args) {  Integer in = **new** Integer(100);  **int** i = in;//进行自动拆箱操作  System.*out*.println(i);    Double dou = **new** Double(100.00001);  **double** d = dou;  System.*out*.println(d);  }  } |

**·将字符串转换成基本数据类型**

范例：将字符串“1122”转换成int类型：

|  |
| --- |
| **package** com.wanczy.packageDemo;  **public** **class** PackageDemo05 {  **public** **static** **void** main(String[] args) {  String s = "1122";  **int** x = Integer.*parseInt*(s);//将String对象转换成基本数据类型  **int** y = Integer.*valueOf*(s);//将String对象转换成基本数据类型  System.*out*.println(x+"\t"+y);  }  } |

将字符串转换成基本数据类型的时候要注意一点：就是在转换的时候要注意异常的发生，比如将字符串“abc”转换成int类型，肯定无法转换，在转换时就会出现数字转换异常。

1. Math:本身是一个类，里面几乎定义了所有的数学运算方法，我们只需要了解两个方法：

·random():取随机数，取出的随机数范围：0-1

·round()：四舍五入取整数

范例：取0-100之间的随机数，取50次

|  |
| --- |
| **package** com.wanczy.mathDemo;  **public** **class** MathDemo01 {  **public** **static** **void** main(String[] args) {  **for** (**int** i = 0; i < 50; i++) {  // double d = Math.random();//取随机数  // double d2 = d\*100;  // int x = (int) Math.round(d2);//四舍五入  // System.out.println(x);  System.*out*.println(Math.*round*(Math.*random*()\*100));  }  }  } |

现在所有的公司发工资或者银行存储卡中的余额都是取两位小数的，现在我们有一段double 的数据，需要四舍五入保留两位小数。

|  |
| --- |
| **package** com.wanczy.mathDemo;  **public** **class** MathDemo02 {  **public** **static** **void** main(String[] args) {  **double** d = 1876.34672783461;  System.*out*.println((Math.*round*(d\*100))/100.0);  }  } |

1. 重点，日期格式的处理

·Date：java.util包中的Date类，一般只会使用无参数的构造方法去取得系统当前的时间，精确到毫秒，其他的方法基本都已过时。

|  |
| --- |
| **package** com.wanczy.dateDemo;  **import** java.util.Date;  **public** **class** DateDemo01 {  **public** **static** **void** main(String[] args) {  Date date = **new** Date();  System.*out*.println(date);  }  } |

那么我现在要去取日期中的年月日、时分秒的详细信息，怎么办呢？我们知道Date类中有此种方法，但是已经过时，这个时候我们可以去使用到Calendar类中的方法，当然Calendar类也可以取得系统当前的时间。

通过API发现Calendar类是一个抽象类，其类的定义格式如下：

|  |
| --- |
| public abstract class Calendarextends Objectimplements Serializable, Cloneable, Comparable<Calendar> |

通过面向对象的多态性，如果说要实例化此类的话比需要依靠子类，现在来观察Calendar的子类：GregorianCalendar，此类的构造方法我们一般只会使用无参数的构造，用于取得Calendar的对象。

观察Calendar中的常用方法：

·public abstract void add(int field, int amount)：给指定的日历字段添加一个值

·public final void clear()：清除日期的内容

·public int get(int field)：取得指定日历字段的值

·public final Date getTime()：将Calendar类对象转换成Date类型对象

·public void set(int field, int value)：给指定的日历字段设置指定的值

·public final void setTime(Date date)：给Calendar对象设置时间

范例1：取得当前日期的年月日时分秒毫秒

|  |
| --- |
| **package** com.wanczy.dateDemo;  **import** java.util.Calendar;  **import** java.util.Date;  **import** java.util.GregorianCalendar;  **public** **class** DateDemo02 {  **public** **static** **void** main(String[] args) {  Date date = **new** Date();  Calendar c = **new** GregorianCalendar();  c.setTime(date);//给Calendar对象设置时间  System.*out*.println("年:"+c.get(Calendar.*YEAR*));  System.*out*.println("月:"+(c.get(Calendar.*MONTH*)+1));  System.*out*.println("日:"+c.get(Calendar.*DAY\_OF\_MONTH*));  System.*out*.println("时:"+c.get(Calendar.*HOUR\_OF\_DAY*));  System.*out*.println("分:"+c.get(Calendar.*MINUTE*));  System.*out*.println("秒:"+c.get(Calendar.*SECOND*));  System.*out*.println("毫秒:"+c.get(Calendar.*MILLISECOND*));  }  } |

范例2：取得的当前日期的前五天的日期

|  |
| --- |
| package com.wanczy.dateDemo;  import java.util.Calendar;  import java.util.Date;  import java.util.GregorianCalendar;  public class DateDemo03 {  public static void main(String[] args) {  Date date = new Date();  Calendar c = new GregorianCalendar();  c.setTime(date);//给Calendar对象设置时间  c.add(Calendar.DAY\_OF\_MONTH, -5);//给日期减5天  System.out.println("年:"+c.get(Calendar.YEAR));  System.out.println("月:"+(c.get(Calendar.MONTH)+1));  System.out.println("日:"+c.get(Calendar.DAY\_OF\_MONTH));  System.out.println("时:"+c.get(Calendar.HOUR\_OF\_DAY));  System.out.println("分:"+c.get(Calendar.MINUTE));  System.out.println("秒:"+c.get(Calendar.SECOND));  System.out.println("毫秒:"+c.get(Calendar.MILLISECOND));  }  } |

·日期格式的转换：我们发现现在不管是Date还是Calendar打印出的时间都不符合我们中国人看时间的习惯，我们的习惯是“2008-08-08”，“2008/08/08 23:10:10”,这个时候怎么办呢？java中提供了一个类，专门用于日期格式的转换，这个类就是DateFormat，现在我们来找到此类：public abstract class DateFormatextends Format发现此类的定义仍然是一个抽象类，所有实例化此类的时候必须依靠其子类：SimpleDateFormat，观察其构造方法：public SimpleDateFormat(String pattern)参数是指定的日期格式。

DateFormat中常用的方法：

**·public final String format(Date date)：将Date类型转换成字符串类型**

**·public Date parse(String source) throws ParseException：将字符串转换成Date日期格式**

那么现在我们怎么转换成中国人习惯的日期格式呢？这个时候需要在SimpleDateFormat中定义一个日期格式，那么这个格式怎么去定义呢？

|  |  |  |  |
| --- | --- | --- | --- |
| **字母** | **日期或时间元素** | **表示** | **示例** |
| y | 年 | [Year](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#year) | 1996; 96 |
| M | 年中的月份 | [Month](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#month) | July; Jul; 07 |
| w | 年中的周数 | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 27 |
| W | 月份中的周数 | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 2 |
| D | 年中的天数 | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 189 |
| d | 月份中的天数 | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 10 |
| F | 月份中的星期 | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 2 |
| E | 星期中的天数 | [Text](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#text) | Tuesday; Tue |
| a | Am/pm 标记 | [Text](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#text) | PM |
| H | 一天中的小时数（0-23） | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 0 |
| k | 一天中的小时数（1-24） | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 24 |
| K | am/pm 中的小时数（0-11） | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 0 |
| h | am/pm 中的小时数（1-12） | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 12 |
| m | 小时中的分钟数 | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 30 |
| s | 分钟中的秒数 | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 55 |
| S | 毫秒数 | [Number](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#number) | 978 |
| z | 时区 | [General time zone](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#timezone) | Pacific Standard Time; PST; GMT-08:00 |
| Z | 时区 | [RFC 822 time zone](mk:@MSITStore:E:\DH02\课堂笔记\JAVASE\JDK_API_1_6_zh_CN.CHM::/java/text/SimpleDateFormat.html#rfc822timezone) | -0800 |

如果说要转换成2008-08-08 23:30:30.300此种日期，那么我们怎么去定义日期格式呢？

yyyy-MM-dd HH:mm:ss.SSS

范例：将Date类型转换成字符串

|  |
| --- |
| **package** com.wanczy.dateDemo;  **import** java.text.DateFormat;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **public** **class** DateDemo04 {  **public** **static** **void** main(String[] args) {  Date date = **new** Date();  DateFormat df = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss.SSS");  String s = df.format(date);//进行转换  System.*out*.println(s);  }  } |

范例2：将字符串转换成Date

|  |
| --- |
| **package** com.wanczy.dateDemo;  **import** java.text.DateFormat;  **import** java.text.ParseException;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **public** **class** DateDemo04 {  **public** **static** **void** main(String[] args) {  String str = "2008-08-08 10:10:10.200";  DateFormat df = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");  // String s = df.format(date);//进行转换  Date date = **null**;  **try** {  date = df.parse(str);  } **catch** (ParseException e) {  e.printStackTrace();  }//将字符串转换成Date类型  System.*out*.println(date);//  }  } |

当然在转换的时候需要注意异常的发生，ParseException ，DateFormatException，如果说格式定义错误就必须会参数这个异常。

总结：

·使用包装类将字符串转换成基本数据类型

·会使用Math中的random()和round()方法

·学会使用Calendar类中的一些方法

·学会使用DateFormat进行日期格式的转换

练习：取得当前时间前一个月的第一天和最后一天，并使用“200-08-08“的形式打印出来。