A.一个未知类的如何学习使用；

A.JDK 查手册；

B.借助 Eclipse 的提示功能；

C.询问同事师友；

D.搜索引擎 //优先查看时间戳离现在最近的网页

B.JDK 手册中的类的使用方法：

A.先查看手册中的该类有什么静态方法和属性，并查看说明，这些属性和方法不需要ｎｅｗ对象就可以是用的；

B.查看这个类有哪些构造函数，

因为非静态的属性和方法只有通过对象来调用，而对象又要通过构造方法来创建；

C.满足某个构造函数的所有构造形参类型的参数；

D.如果一级形参类型又是一个复杂的对象，可以先考虑创建出形参类型的对象后，再传入目标构造函数中；

如果一级形参对象的构造函数的形参，也是一个复杂的对象依次类推，直至全部满足即可。

E.写Ｄｅｍｏ代码，调用对象的属相和方法测试，并理解运用之。

F.如果一个类可能并不是孤立的，和其他类是有关联的，直接上网查询，比如：多线程。

下面所写的所有类的详细信息都可以去查询 JDK-API 手册了解！

1.java.lang.Object

类 Object 是类层次结构的根类。每个类都使用 Object 作为超类。所有对象（包括数组）都实现这个类的方法。该类提供了面向对象编程技术的基本机制。

方法摘要	
protected Object	clone() 创建并返回此对象的一个副本。
boolean	equals(Object obj) 指示其他某个对象是否与此对象“相等”。
protected void	finalize() 当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾回收器调用此方法。
Class<?>	getClass()

	返回此 Object 的运行时类。
int	hashCode() 返回该对象的哈希码值。
void	notify() 唤醒在此对象监视器上等待的单个线程。
void	notifyAll() 唤醒在此对象监视器上等待的所有线程。
String	toString() 返回该对象的字符串表示。
void	wait() 在其他线程调用此对象的 notify() 方法或 notifyAll() 方法前，导致当前线程等待。
void	wait(long timeout) 在其他线程调用此对象的 notify() 方法或 notifyAll() 方法，或者超过指定的时间量前，导致当前线程等待。
void	wait(long timeout, int nanos) 在其他线程调用此对象的 notify() 方法或 notifyAll() 方法，或者其他某个线程中断当前线程，或者已超过某个实际时间量前，导致当前线程等待。

equals()

用来比较引用类型数据的等价性，Object 本身比较的是对象的引用。

equals()与==

A.equals()方法只能比较引用类型数据，而“==”可以比较引用类型及基本类型数据

B.对于引用类型：==比较的是对象的引用；equals()比较的是对象的内容

C.重写了 equals 方法后，一定要重写 hashCode 方法

为什么要重写 equals()方法

A.equals 方法本意是比较两个对象在”含义”上是否相等，但因为 Object 提供的实现只是机械地判断两个实体对象是否为同一对象，所以我们一般会改写 equals 方法以达到比较对象内容的目的.

B.集合类中不允许存在两个相同的对象实例，其判读的依据就是 equals()方法

equals()方法的约定规范

自反性：x.equals(x)为 true

对称性：x.equals(y)与 y.equals(x)结果相同

传递性：x.equals(y)且 y.equals(z)为 true，则 x.equals(z)为 true

一致性：当 x|y 对象没有改变，则多次调用 x.equals(y)返回的结果是一样的

非空性：对于非空引用 x，x.equals(null)返回一定是 false

equals()方法的重写步骤

A.使用==判断实参是否为指向当前对象的一个引用

B.使用 instanceof 检测实参对象是否为正确的类型

C.将实参转换为正确的类型

D.根据业务需求判断当前对象与实参对象是否相等

可以利用 Eclipse 自动生成后，自己再修改。

hashCode()

返回该对象的 hashCode 编码，用于区别不同的对象。不同对象 hashCode 值都是应该不一样的。

实际上，由 Object 类定义的 hashCode 方法确实会针对不同的对象返回不同的整数。（这一般是通过将该对象的内部地址转换成一个整数来实现的，但是 JavaTM 编程语言不需要这种实现技巧。）

hashCode()方法的约定规范

A.对象未改变，散列值应该相同

一个对象的生命周期中，用于 equals()方法中进行比较的属性值没有被修改过，那么多次调用 hashCode()返回的散列值也应该相同

B.相等的对象必须产生相等的散列值

如果两个对象根据 equals()方法判断的结果是相等的，那么调用这两个对象的 hashCode()方法返回的散列值是相同的

C.不相等的对象可以产生不相等的散列值

如果两个对象根据 equals()方法判断的结果是不相等的，那么调用这两个对象的 hashCode()方法返回的散列值可以是不同的。这种情况不强制要求

toString()

以字符串形式返回当前对象的有关信息，在 Object 类的原始定义中，所返回的是对象所属的类型名称及其哈希码。

Object 的 toString()方法返回的字符串：类名+@+对象的散列码

注意事项

A.当使用 System.out.println()方法直接输出引用类型变量时，println()方法中会先自动调用其 toString()方法，再将所返回的字符串信息输出。

B.在进行 String 与其它类型数据的连接操作时，自动调用 toString()方法，基本类型数据转换为 String 类型时，调用了对应封装类的 toString()方法。

C.可以根据需要在用户定义类型中重写 toString()方法。

finalize()

Java 运行时环境中的垃圾收集器在销毁一个对象之前，会自动调用该对象的 finalize()方法，然后才释放对象的内存空间。在实际应用中很少被重写，因为重写的意义并不是很大.

clone()

用于对象拷贝，要进行“ 克隆” 的对象所属类必须实现 java.lang.Cloneable 接口。

重写 clone()的方法步骤

A.克隆类实现 java.lang. Cloneable

B.重写 clone()方法

PS:通常情况下：XXXable 等以 able 结尾，表示的都是接口，具有 XXX 能力！

2.系统相关类：

java.lang.System

System 类是一个最终类，不能实例化，它的所有方法和属性都是静态的

System 提供了标准输入、标准输出和错误输出流；对外部定义的属性和环境变量的访问；加载文件和库；复制数组等。

public static void exit(int status)
终止正在运行的 JVM，0 表示正常退出，非 0 表示异常终止
public static void gc()
运行垃圾回收器，让虚拟机尽最大努力从所有丢弃的对象中收回空间
public static long currentTimeMillis()
返回以毫秒为单位的当前时间
public static Properties getProperties() / public static String getProperty(String key)
返回当前所有系统属性的 Properties 集合/返回相应的 key 对应的系统属性
public static void setProperties(Properties props) public static String setProperty(String key, String value)
设置当前系统属性

java.lang.Runtime

每个 Java 应用程序都有一个 Runtime 类实例，使应用程序能够与其运行的环境相连接。可以通过 getRuntime 方法获取当前运行时。**应用程序不能创建自己的 Runtime 类实例。也没有构造函数。**

方法摘要	
void	addShutdownHook(Thread hook) 注册新的虚拟机来关闭钩子。
int	**availableProcessors()** **向 Java 虚拟机返回可用处理器的数目。**
Process	exec(String command) 在单独的进程中执行指定的字符串命令。
Process	exec(String[] cmdarray) 在单独的进程中执行指定命令和变量。
Process	exec(String[] cmdarray, String[] envp) 在指定环境的独立进程中执行指定命令和变量。
Process	exec(String[] cmdarray, String[] envp, File dir) 在指定环境和工作目录的独立进程中执行指定的命令和变量。
Process	exec(String command, String[] envp) 在指定环境的单独进程中执行指定的字符串命令。
Process	exec(String command, String[] envp, File dir) 在有指定环境和工作目录的独立进程中执行指定的字符串命令。
void	exit(int status) 通过启动虚拟机的关闭序列，终止当前正在运行的 Java 虚拟机。
long	freeMemory() 返回 Java 虚拟机中的空闲内存量。

void	gc() 运行垃圾回收器。
static Runtime	getRuntime() 返回与当前 Java 应用程序相关的运行时对象。
void	halt(int status) 强行终止目前正在运行的 Java 虚拟机。
void	load(String filename) 加载作为动态库的指定文件名。
void	loadLibrary(String libname) 加载具有指定库名的动态库。
long	maxMemory() 返回 Java 虚拟机试图使用的最大内存量。
boolean	removeShutdownHook(Thread hook) 取消注册某个先前已注册的虚拟机关闭钩子。
void	runFinalization() 运行挂起 finalization 的所有对象的终止方法。
long	totalMemory() 返回 Java 虚拟机中的内存总量。
void	traceInstructions(boolean on) 启用 / 禁用指令跟踪。
void	traceMethodCalls(boolean on)

	启用 / 禁用方法调用跟踪。

3. 字符串相关类：

java.lang.String

public final class String extends Object

implements Serializable, Comparable<String>, CharSequence

String 类对象表示不可修改的 Unicode 编码字符串

构造方法
String()|String(String value)|String(char [ ] value)|String(byte [ ] value)
连接、转换、截断
concat()|replace()|substring()|toLowerCase()|toUpperCase()|trim()
检索、查找
charAt()|startsWith()|indexOf()|lastIndexOf()|length()
内容比较
equals()|equalsIgnoreCase()|compareTo()

java.lang.StringBuffer

该类对象保存可修改的 Unicode 字符序列。

方法列表

构造方法
StringBuffer()|StringBuffer(int capacity)|StringBuffer(String initialString)
长度、容量
length()|capacity()|ensureCapacity()
功能方法
append()|insert()|setCharAt()|reverse()|delete()|deleteCharAt()|toString()

java.lang.StringBuilder

JDK5.0 以后引入， 该类能够提供与 StringBuffer 相同的功能

StringBuilder 与 StringBuffer 比较：

A. StringBuffer 类是线程安全的，而 StringBuilder 则不是，即不保证其对象的同步性，在多线程环境中是不安全的。

B. StringBuilder 在性能上要比 StirngBuffer 好一些。

简单记忆：StringBuilder 性能好但是线程不安全！

4. 封装类

将原始数据类型进行包装以 java 类的形式提供给用户使用，这样的包装了原始数据类型的类我们称做类型包装类。

原始数据类型	包装类

byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

原始类型数据和包装类对象的转换

A. 原始类型数据→包装类对象

使用包装类的构造方法

使用包装类的静态方法：Type.valueOf(type parameter)

B. 包装类对象→原始类型数据

使用包装类对象的 typeValue()方法

C. 自动装箱与自动拆箱:**Java 自动装箱机制和自动拆箱机制**

http://blog.csdn.net/sun_wangdong/article/details/43281105

字符串对象与包装类对象之间的转换

A. 字符串对象→包装类对象

使用包装类的字符串参数构造方法

使用包装类字符串参数静态方法：valueOf(type parameter)

可能引发 NumberFormatException 异常

B. 包装类对象→字符串对象

使用包装类对象的 toString()方法

5. 时间相关类：

java.util.Date

public class Date extends Object implements Serializable, Cloneable, Comparable<Date>

用距格林威治标准时间 1970 年 1 月 1 日的毫秒偏移量表示日期和时间，该类用于表示特定的时间点。Date 类最佳应用之一是获取系统当前时间

注意

但不支持日期的国际化和分时区显示。java1.1 发布后，很多方法的实现被移到了 Calendar 和 DateFormat 类中实现，Date 类中的很多方法建议不再使用

方法列表

构造方法
public Date() public Date(long date)
功能方法

public int compareTo(Date anotherDate) public boolean equals(Object obj) public long getTime() public String toString()

java.util.Calendar

用距格林威治标准时间 1970 年 1 月 1 日的毫秒偏移量表示日期和时间。该类是抽象类，提供了常规的日期修改功能和国际化支持。

获取实例的静态方法
public static Calendar getInstance()
日历常量
YRAR/MONTH/DATE/HOUR/MINUTE/SECOND……
功能方法
public int get(int field) public abstract void add(int field, int amount) public void set(int field, int value) public final void set(int year, int month, int date) public final void set(int year,int month,int date,int hourOfDay,int minute, int second) public final Date getTime() public String toString()

java.util.Locale

该类描述特定的地理、政治/文化上的地区， Locale 对象主要封装了“地区”和“语言种类”两方面的信息。通常用于在国际化/本地化程序中以地区/语言相关的方式显示日期、数字或文本信息等。

方法列表

构造方法
public Locale(String language) public Locale(String language, String country)
功能方法
public static Locale getDefault() public String getCountry() public String getLanguage() public final String getDisplayName() public static Locale[] getAvailableLocales()

java.util.TimeZone

该类被定义为抽象类，用来描述时区信息

方法列表

获取 TimeZone 对象
public static TimeZone getDefault() public static TimeZone getTimeZone(String ID)

获取有效的时区
public static String[] getAvailableIDs()

java.util.GregorianCalendar

该类是 Calendar 的子类，该类提供了世界上大多数国家/地区使用的标准日历系统，并添加判断闰年的功能。

方法列表

构造方法
public GregorianCalendar() public GregorianCalendar(TimeZone zone) public GregorianCalendar(Locale locale) public GregorianCalendar(TimeZone zone, Locale locale) public GregorianCalendar(int year, int month, int dayOfMonth) public GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute, int second)
功能方法
public boolean isLeapYear(int year)　　//判断闰年的方法

java.text.DateFormat

该类提供了将日期/时间信息进行格式化处理的功能

方法列表

构造方法
public static final DateFormat getDateInstance() public static final DateFormat getTimeInstance() public static final DateFormat getDateTimeInstance()
功能方法
public final String format(Date date)

java.text.SimpleDateFormat

该类是 DateFormat 类的子类，它允许用户自己定义日期/时间信息的表现格式， 并提供更灵活的 Date<=>String 信息转换和解析的功能。

注意

在创建 SimpleDateFormat 实例时应指定所转换的字符串格式（模式串），否则系统会缺省采用与所在语言环境相关的默认格式。

方法列表

构造方法
public SimpleDateFormat() public SimpleDateFormat(String pattern) public SimpleDateFormat(String pattern, Locale locale)
功能方法
public String format(Date date) public Date parse(String source)

以下示例显示了如何在美国语言环境中解释日期和时间模式。给定的日期和时间为美国太平洋时区的本地时间 2001-07-04 12:08:56。

日期和时间模式	**结果**
"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 AD at 12:08:56 PDT
"EEE, MMM d, ''yy"	Wed, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o''clock' a, zzzz"	12 o'clock PM, Pacific Daylight Time
"K:mm a, z"	0:08 PM, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.July.04 AD 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"	Wed, 4 Jul 2001 12:08:56 -0700
"yyMMddHHmmssZ"	010704120856-0700
"yyyy-MM-dd'T'HH:mm:ss.SSSZ"	2001-07-04T12:08:56.235-0700

字母	**日期或时间元素**	**表示**	**示例**
G	Era 标志符	Text	AD
y	年	Year	1996; 96
M	年中的月份	Month	July; Jul; 07
w	年中的周数	Number	27
W	月份中的周数	Number	2
D	年中的天数	Number	189
d	月份中的天数	Number	10

F	月份中的星期	Number	2
E	星期中的天数	Text	Tuesday; Tue
a	Am/pm 标记	Text	PM
H	一天中的小时数（0-23）	Number	0
k	一天中的小时数（1-24）	Number	24
K	am/pm 中的小时数（0-11）	Number	0
h	am/pm 中的小时数（1-12）	Number	12
m	小时中的分钟数	Number	30
s	分钟中的秒数	Number	55
S	毫秒数	Number	978
z	时区	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	时区	RFC 822 time zone	-0800

6.数学相关类：

java.lang.Math

public **final** class **Math** extends Object

该类是最终类，提供常用数学功能的静态方法和数学常量

方法列表

常量
PI、E

数据截断
ceil()|floor()|round()
最大、最小、绝对值
max()|min()|abs()
三角函数
sin()|cos()|tan()|asin()|acos()|atan()|toDegrees()|toRadians()
幂运算、对数运算
pow()|exp()|sqrt()|log()|log10()
随机数
random()

java.util.Random

该类是基于“线性同余”算法的一种伪随机数序列生成器。之所以伪是因为当随机数种子相同时它只是一个均匀分布的数值序列

方法列表

构造方法
public Random() public Random(long seed)
功能方法
public int nextInt() public long nextLong() public float nextFloat() public double nextDouble() public int nextInt(int n)

public void setSeed(long seed)

java.math.BigInteger/java.math.BigDecimal

这两种类型可以分别提供任意长度/精度的整数和浮点数运算功能

方法列表

构造方法
public BigInteger(String val)
功能方法
public static BigInteger valueOf(long val)
public BigInteger add(BigInteger val)
public BigInteger subtract(BigInteger val)
public BigInteger multiply(BigInteger val)
public BigInteger divide(BigInteger val)
public int compareTo(BigInteger val)
public BigInteger remainder(BigInteger val)
public BigInteger pow(int exponent)
public String toString()

java.text.NumberFormat/java.text.DecimalFormat

这两种类型提供了将数字格式化为语言环境相关字符串以及逆向解析字符串为数字的功能。

方法列表

获取对象的静态方法
public static final NumberFormat getInstance()
public static NumberFormat getInstance(Locale inLocale)
public static final NumberFormat getCurrencyInstance()
public static NumberFormat getCurrencyInstance(Locale inLocale)

public static final NumberFormat getPercentInstance() public static NumberFormat getPercentInstance(Locale inLocale)
功能方法
public final String format(double number)