

JAVA 程序设计



第7章 工具类及常用算法



第7章 工具类及常用算法

- 7.1 Java语言基础类
- 7.2 字符串和日期
- 7.3 集合
- 7.4 排序与查找
- 7.5 泛型
- 7.6 常用算法

7.1 Java语言基础类

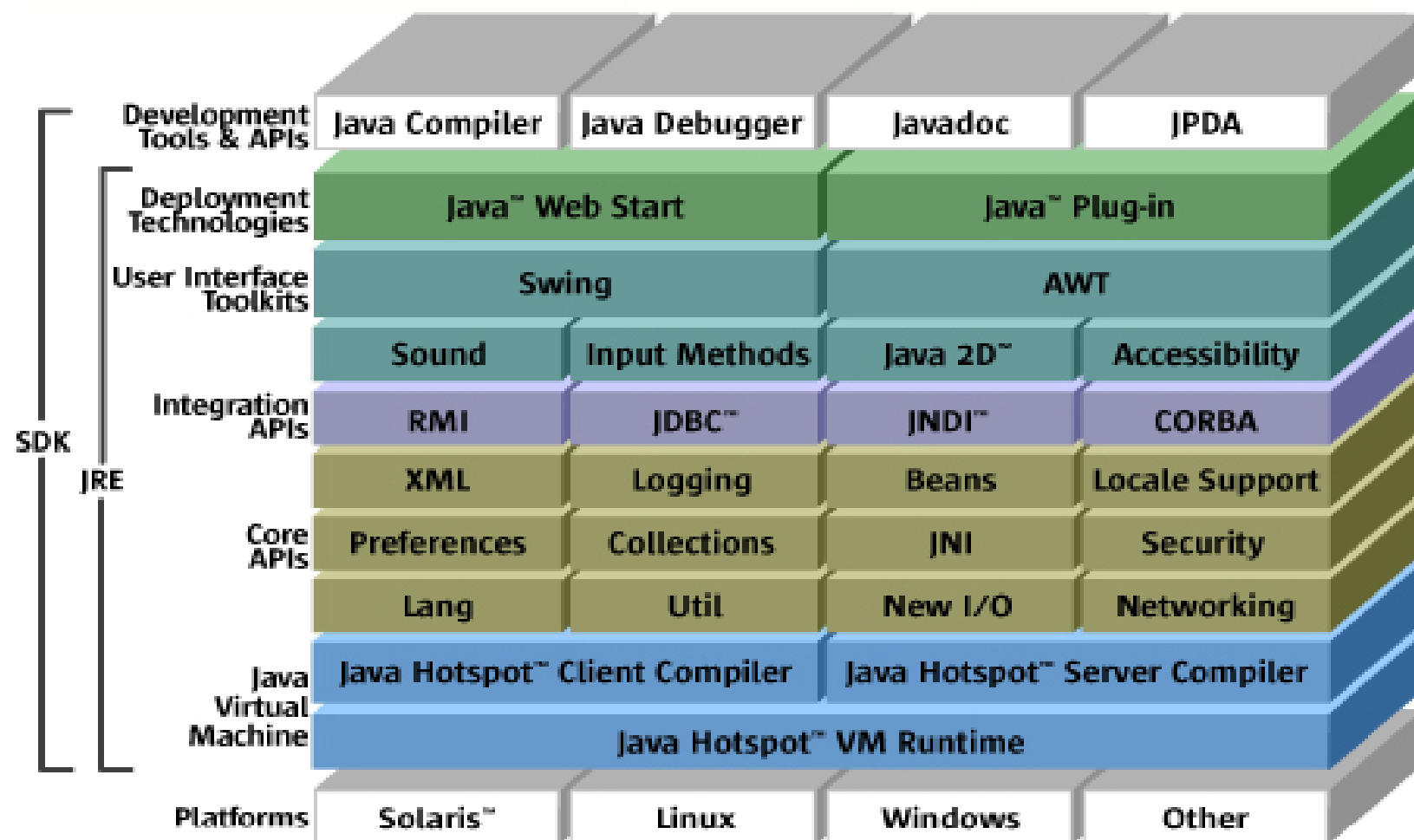




Java语言基础类

A horizontal line of white dots spanning the width of the slide, positioned below the title.

JDK API





- **java.lang** Java语言的核心类库
 - Java是自动导入java.lang.*的
- **java.util** 实用工具
- **java.io** 标准输入/输出类库
- **java.awt** **javax.swing** 图形用户界面(GUI)的类库
- **java.net** 网络功能的类库
- **java.sql** 数据库访问的类库
- 等等

阅读JDK API文档



- 在线查阅

- <http://docs.oracle.com/javase/8/docs/api/index.html>

- 文档下载

- <http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html>

- 更多文档

- <http://docs.oracle.com/javase/8/docs/index.html>

- 网上有chm格式的，有中文版



- JDK的源代码
 - ▣ 安装JDK后即有 **src.zip**
 - ▣ 例如：在 C:\Program Files\Java\jdk\下

Object类



- Object类是所有类的直接或间接父类
- 让所有的类有了一致性



(1) equals()

- 讲到了 “==” 与equals的区别
- 简单地说，==是引用是否相等，equals是内容（含义）相等
 - Integer one = new Integer (1);
 - Integer anotherOne = new Integer (1);
 - if(one==anotherOne) 。 。 。 //false
 - if (one.equals (anotherOne)) 。 。 。 //true
- 如果覆盖equals()方法，一般也要覆盖hashCode()方法



(2) getClass ()

- getClass ()方法是final方法，它不能被重载
- 它返回一个对象在运行时所对应的类的表示

```
void PrintClassName( Object obj ) {  
    System.out.println(“ The object’s class is “ + obj.getClass( ).getName( ) );  
}
```

```
Object creatNewInstanceOf (object obj) {  
    return obj.getClass( ).newInstance( );  
}
```



(3) toString()

- toString()方法用来返回对象的字符串表示
- 常用于显示
 - ▣ System.out.println (person);
- 另外，用于字符串的加号
 - ▣ “current person is “ + person
- 通过重载toString ()方法，可以适当地显示对象的信息以进行调试。



(4) finalize()

- 用于在垃圾收集前清除对象，前面已经讲述。



Object的其他方法

- (5) notify(), notifyAll(), wait()
与线程相关，以后讲解



基本数据类型的包装类

- Java的基本数据类型用于定义简单的变量和属性将十分方便，但为了与面向对象的环境一致，Java中提供了基本数据类型的包装类（wrapper），它们是这些基本类型的面向对象的代表。
- 与8种基本数据类型相对应，基本数据类型的包装类也有8种，分别是：
 - ▣ Character , Byte , Short , Integer , Long , Float , Double , Boolean。



包装类的特点

- (1) 这些类都提供了一些常数
 - ▣ 如Integer.MAX_VALUE (整数最大值) , Double.NaN(非数字) , Double. POSITIVE_INFINITY (正无穷) 等。
- (2) 提供了valueOf(String) , toString()
 - ▣ 用于从字符串转换及或转换成字符串。
- (3) 通过xxxxValue()方法可以得到所包装的值
 - ▣ Integer对象的intValue()方法。
- (4) 对象中所包装的值是不可改变的 (immutable) 。
 - ▣ 要改变对象中的值只有重新生成新的对象。
- (5) toString(), equals()等方法进行了覆盖。
- 除了以上特点外, 有的类还提供了一些实用的方法以方便操作。
 - ▣ 例如, Double类就提供了parseDouble(), max, min方法等。



包装与拆包

- JDK1.5以上，有包装 (boxing)及拆包(unboxing)
- Integer I = 5;
 - 即 I = Integer.valueOf(5);
- int i = I;
 - 即 i = I.intValue();



- Math类用来完成一些常用的数学运算
- `public final static double E;` // 数学常量e
- `public final static double PI;` // 圆周率常量
- `public static double abs(double a);` // 绝对值
- `public static double exp(double a);` // 参数次幂
- `public static double floor(double a);` // 不大于参数的最大整数
- `public static double IEEE remainder(double f1, double f2);` // 求余
- `public static double log(double a);` // 自然对数
- `public static double max(double a, double b);` // 最大值
- `public static float min(float a, float b);` // 最小值
- 例: TestMath.java

- `public static double pow(double a, double b);`
// 乘方
- `public static double random();` // 产生0和1(不含1)之间的伪随机数
- `public static double rint(double a);` // 四舍五入
- `public static double sqrt(double a);` // 平方根
- `public static double sin(double a);` // 正弦
- `public static double cos(double a);` // 余弦
- `public static double tan(double a);` // 正切
- `public static double asin(double a);` // 反正弦
- `public static double acos(double a);` // 反余弦
- `public static double atan(double a);` // 反正切



- 在Java中，系统属性可以通过环境变量来获得
 - ▣ `System.getProperty(String name)`方法获得特定的系统属性值
 - ▣ `System.getProperties()`方法获得一个 `Properties`类的对象，其中包含了所有可用的系统属性信息
- 在命令行运行Java程序时可使用-D选项添加新的系统属性
 - ▣ 如 `java -Dvar=value MyProg`
- 示例：`SystemProperties.java`

7.2 字符串和日期





字符串和日期

.....



- 字符串可以分为两大类

- String类

- 创建之后不会再做修改和变动，即 immutable

- StringBuffer、StringBuilder类

- 创建之后允许再做更改和变化
 - 其中 StringBuilder是JDK1.5增加的 它是非线程安全的

- 特别注意

- 在循环中使用String的+=可能会带来效率问题

- 示例：[StringAndStringBuffer.java](#)



- String 类对象保存不可修改(immutable)的Unicode字符序列
 - ▣ String类的下述方法能创建并返回一个新的String对象实例: concat, replace, replaceAll, substring, toLowerCase, toUpperCase, trim, toString.
 - ▣ 查找: endsWith, startsWith, indexOf, , lastIndexOf.
 - ▣ 比较: equals, equalsIgnoreCase,
 - ▣ 字符及长度: charAt , length.
- 例 : TestStringMethod.java
- Jdk1.5 增加了format函数
 - ▣ %1\$,8.5f %序号\$ 标识 宽度及精度 转换方式



- 除了immutable特点外，还要注意String常量的内部化（interned）问题
- 即同样的字符串常量是合并的（是指向同一个引用的）
- 以保证 “abc”==“abc”
 - 但是 “abc” != new String(“abc”)



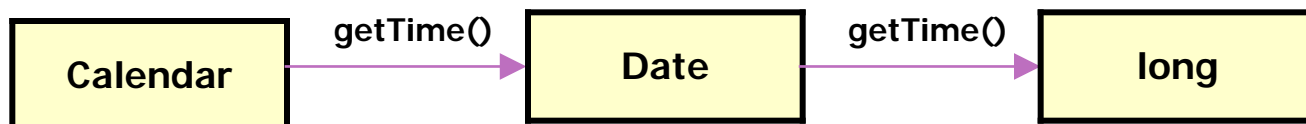
StringBuffer类

- StringBuffer类对象保存可修改的Unicode字符序列:
- StringBuilder类似，它效率更高，不考虑线程安全性
- 构造方法
 - StringBuffer()
 - StringBuffer(int capacity)
 - StringBuffer(String initialString)
- 实现修改操作的方法:
 - append, insert, reverse, setCharAt, setLength.



字符串的分割

- java.util.StringTokenizer类提供了对字符串进行分割的功能。
- 构造
 - ▣ StringTokenizer(String str, String delim);
- 该类的重要方法有：
 - ▣ public int countTokens() ; // 分割串的个数
 - ▣ public boolean hasMoreTokens() ; // 是否还有分割串
 - ▣ public String nextToken() ; // 得到下一分割串
- 例：TestStringTokenizer.java
- 另String类的 matches, replaceAll, split可以使用正则表达式（以后讲）



- Calendar
 - ❑ 得到一个实例 `Calendar.getInstance() //Locale.ZH`
 - ❑ `.get(DAY_OF_MONTH)` `.getDisplayName(DAY_OF_WEEK)`
 - ❑ `.set` `.add(HOUR,1)` `.roll(MONTH, 5),`
 - ❑ `.setTime(date), .getTime()`
- Date
 - ❑ `new Date(), new Date(System.currentTimeMillis())`
 - ❑ `.setTime(long), .getTime()`
- `SimpleDateFormat("yyyy-MM-dd HH:mm:ss")`
 - ❑ `.format,` `.parse`
- 例 `CalendarDate.java`

Java8中的time api



- `java.time.*`
- `java.time.format.*`
- 主要的类
 - Instant 时刻 Clock 时区 Duration 时间段
 - 常用的类 LocalDateTime LocalDate LocalTime
 - `.of` `.parse` `.format` `.plus` `.minus`
 - DateTimeFormatter
- 示例 [CalendarDate8.java](#)

7.3 集合类





集合类



Collection API



- Collection API提供“集合”“收集”的功能
- Collection API包含一系列的接口和类



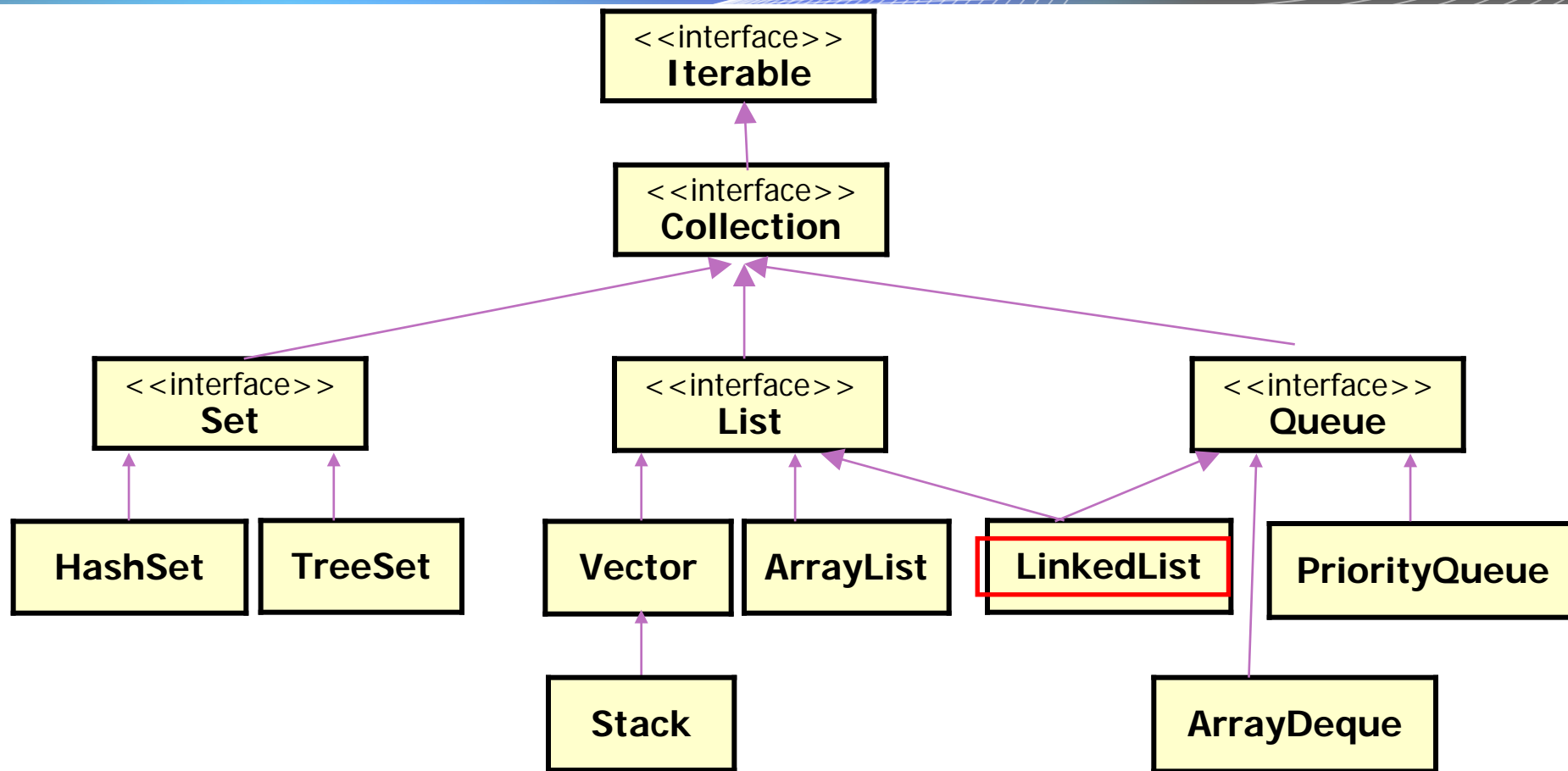
Collection API包含三大类

- Collection接口：有两个子接口
 - List: (Collection的子接口)记录元素的保存顺序，且允许有重复元素
 - Set: (Collection的子接口)不记录元素的保存顺序，且不允许有重复元素
- Map接口，即映射
 - 键-值对 (key-value pair) 的集合

Collection 接口

<<interface>> Collection
<div data-bbox="784 504 963 604" style="border: 2px solid red; width: 70px; height: 70px; display: flex; align-items: center; justify-content: center; margin-bottom: 5px;">+</div> <div data-bbox="784 504 1516 554" style="display: inline-block; margin-bottom: 5px;">add(element : Object) : boolean</div> <div data-bbox="784 554 1503 604" style="display: inline-block; margin-bottom: 5px;">+remove(element : Object) : boolean</div> <div data-bbox="784 604 1019 646" style="display: inline-block; margin-bottom: 5px;">+size() : int</div> <div data-bbox="784 646 1192 689" style="display: inline-block; margin-bottom: 5px;">+isEmpty() : boolean</div> <div data-bbox="784 689 1516 739" style="display: inline-block; margin-bottom: 5px;">+contains(element : Object) : boolean</div> <div data-bbox="784 739 1177 782" style="display: inline-block;">+iterator() : Iterator</div>

Collection 层次结构(简化)





- List接口：线性表 (linear list)

- ▣主要的实现类是 ArrayList, LinkedList , 以及早期的Vector

- List接口

- public interface List<E> extends Collection<E> {
 - E get(int index);
 - E set(int index, E element);
 - void add(int index, E element);
 - E remove(int index);
 - int indexOf(Object o);
 -
 - }



- 迭代器 Iterator (所有的Collection都能产生)

- Iterator iterator = iterable.iterator();

- while(iterator.hasNext()) doSomething(iterator.next());



增强的for语句

- 在JDK1.5以后，增强的for语句(enhanced for)或叫for-each
 - `for(Element e : list) doSomething(e);`
 - `for (Photo photo : album){`
 - `System.out.println(photo.toString());`
 - `}`
 - 编译器生成了Iterator的while(hasNext ()) {...next() }
- 示例：[TestList.java](#)



- 是遵循“后进先出” (Last In First Out, LIFO)原则
- 重要线性数据结构
- 包含三个方法
 - `public Object push(Object item)` : 将指定对象压入栈中。
 - `Public Object pop()` : 将 栈最上面的元素从栈中取出, 并返回这个对象。
 - `public boolean empty()` : 判断栈中没有对象元素。
- 示例 : [TestStack.java](#)



- 队列(Queue)，也是重要的线性数据结构。
 - ▣ 队列遵循“先进先出”(First In First Out , FIFO)的原则
 - ▣ 固定在一端输入数据(称为入队)，另一端输出数据(称为出队)。
- 重要的实现是LinkedList类，示例：[TestQueue.java](#)

	可抛出异常的	返回元素的
Insert（插入）	add(e)	offer(e)
Remove（移除）	remove()	poll()
Examine（检查）	element()	peek()

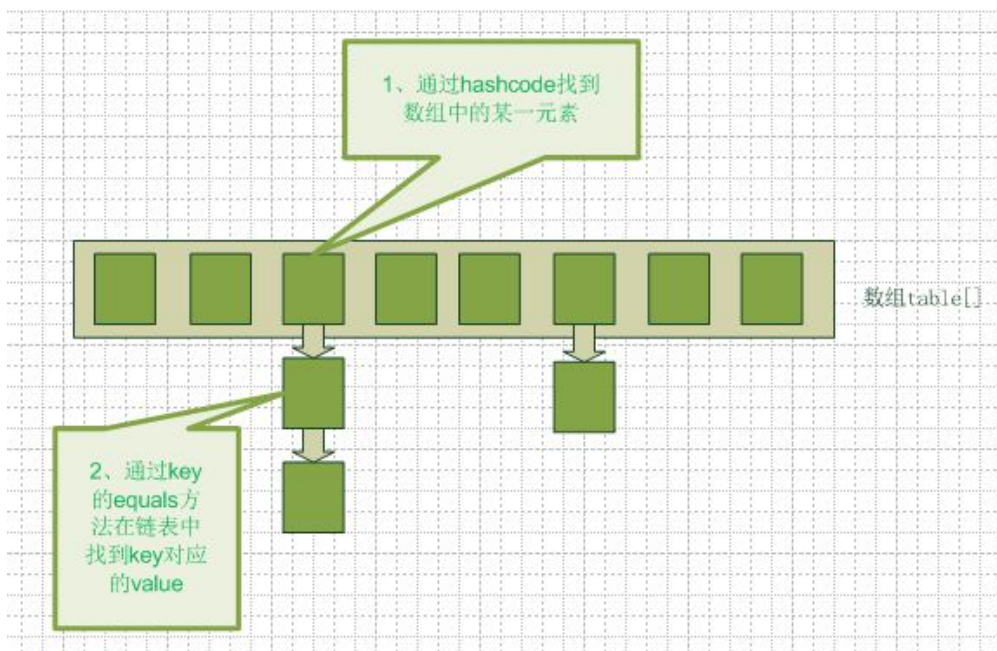


几个早期的类或接口

- Vector , 现多用 ArrayList
 - 相当于动态数组(比JDK1.0中的 ArrayList好), elementAt,
- Stack , 现多用 LinkedList
 - Stack是Vector的子类, push, pop, peek
- Hashtable , 现多用 HashMap
 - Hashtable实现Map接口, 参见Properties类
- Enumeration , 现多用Iterator
 - Enumeration用另一种方式实现Iterator的功能
 - 如Vector可以得到枚举器
 - Enumeration<E> e = v.elements();
 - while(e.hasMoreElements()) doSomething(e.nextElement())



- Set 集
 - ▣两个重要的实现 **HashSet**及**TreeSet**
 - ▣其中TreeSet的底层是用TreeMap来实现的
- Set中对象不重复，即：
 - ▣hashCode()不等
 - ▣如果hashCode()相等，再看equals或==是否为false
-
- 例：[TestSet.java](#)

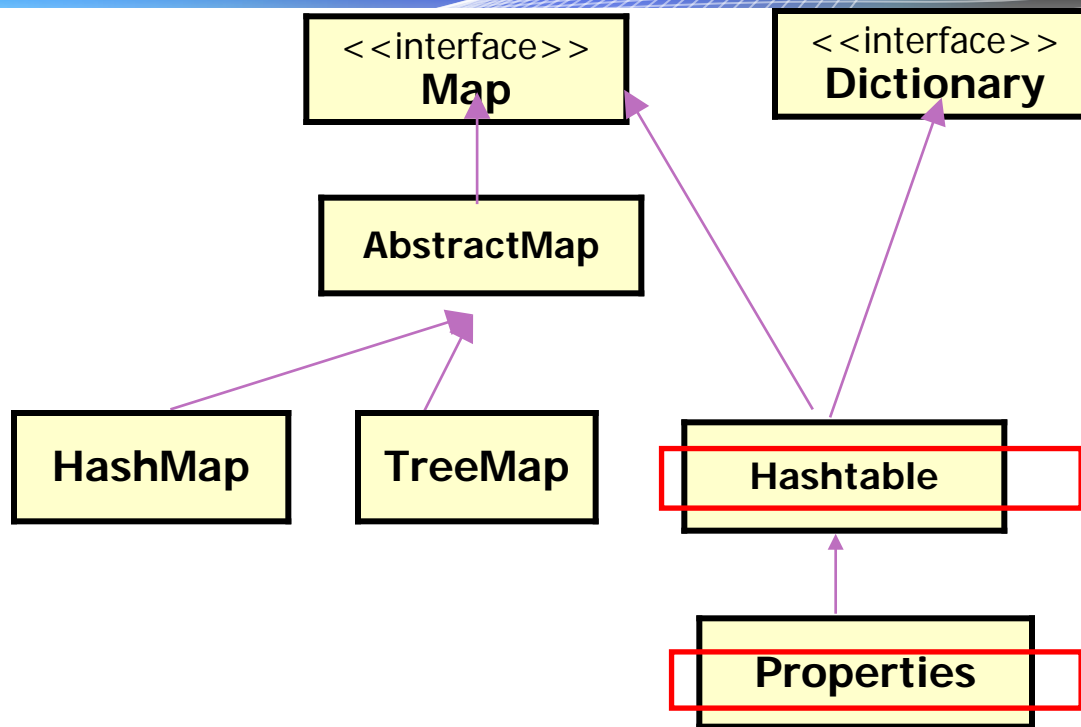


- 注：
 - String 对象的哈希码根据以下公式计算：
$$s[0] * 31^{(n-1)} + s[1] * 31^{(n-2)} + \dots + s[n-1]$$
 - 使用 int 算法，这里 $s[i]$ 是字符串的第 i 个字符， n 是字符串的长度， $^$ 表示求幂。（空字符串的哈希值为 0。）
- 一般在覆盖时，要同时覆盖 hashCode、equals 方法



- Map是键-值对的集合
 - ▣ 其中可以取到entrySet()、keySet()、values()、
 - ▣ Map.Entry是一个嵌套接口
- Map类的重要实现
 - ▣ HashMap类
 - ▣ TreeMap类：用红黑树的算法
- 例：TestMap.java

Map层次结构(简化)



7.4 排序与查找





排序与查找





- 自编程序排序与查找
 - 如冒泡排序、选择排序、快速排序等
- 系统已有的排序与查找
 - 如 Arrays类及Collections类



- Arrays类是用于对数组进行排序和搜索的类。
 - ▣ Arrays.asList(10, 7, 6, 5, 9) 方法可以直接得到一个List对象
- Arrays类提供了 **sort()** 和 **binarySearch()**
- **执行binarySearch()之前应调用sort()**
 - ▣ public static void sort(List list);
 - ▣ public static void sort(List list, Comparator c);
 - ▣ public static int binarySearch(List list, Object key);
 - ▣ public static int binarySearch(List list, Object key, Comparator c);
- 例：TestArraysSort.java



- 要么对象是java.lang.Comparable
 - 实现方法
 - `public int compareTo(Object obj){
 return this.price - ((Book)obj).price;`
 - `}`
- 要么提供一个java.lang. Comparator
 - 实现方法 `public int compare(T o1, T o2)`
 - 这些方法的含义要与equals不冲突



- 此类完全由在 collection 上进行操作静态方法组成.
- 如 `sort`, `binarySearch`, `reverse`等
- 例 `TestCollectionsSort.java`
- 更一般地, 使用 `Lambda表达式` (Java8以上)
- 例 `TestCollectionsSortByLambda.java`

7.5 泛型





泛型





- 泛型 (Generic) 是JDK1.5增加的最重要的Java语言特性。
- 使用泛型可以针对不同的类有相同的处理办法
 - `Vector<String> v = new Vector<String> ();`
 - `v.addElement("one");`
 - `String s = v.elementAt(0);`
- 使用泛型的好处
 - 类型更安全
 - 适用更广泛，针对不同的类有相同的处理办法，但这些类之间不一定有继承关系。



自定义泛型

- 自定义泛型类
 - GenericTreeClass.java
- 自定义泛型方法
 - GenericMethod.java
 - 注意：<>要写到方法名字的前面



- 使用?

- 如Collections的reverse方法
- `reverse(List<?> list)`

- 使用**extends**

- 如Set的addAll方法
- `addAll(Collection<? extends E> col)`

- 使用**super**

- 如Collections的fill方法
- `fill(List<? super T> list, T obj)`



有时泛型写起来比较复杂

- Arrays.sort方法
 - `public static <T> void sort(T[] a, Comparator<? super T> c)`
- Stream.map方法
 - `public <R> Stream<R> map(Function<? super T, ? extends R> mapper)`
- Collections.max方法
 - `public static <T extends Object & Comparable<? super T>>T`
 - `max(Collection<? extends T> coll)`



- 协变(Covariance) ? extends T

- GenericCovariance.java

- 原因 : ListArray<Apple>不是ListArray<Fruit>的子类
 - 但又想让<Apple>当作<Fruit>
 - 就声明ListArray<? extends Fruit>

- 逆变(Contravariance) ? super T

- GenericContrvariance.java

- 原因 : Basket<Apple>不是Basket<Fruit>的子类
 - 但又想让Comparator<Fruit>用于Comparator<Apple>
 - 就声明Comparator<? super Fruit>

- 注 : ? 实际上是去类型化(变成Object), 只在编译时检查

- 总之: 协变 用于获取, 用于out, 用于Producer

- 逆变 用于加入, 用于in, 用于Consumer

- 关于能否赋值, 请见 TCovarContravar.java

7.6 常用算法





常用算法

A horizontal dotted line consisting of 30 white dots on a blue background, spanning the width of the slide.



- 常用的几种算法
- 这些算法属于“通用算法”
 - 它们在解决许多问题中都有应用。
 - 遍试、迭代、递归和回溯



- 遍试（穷举，exhaust algorithm）
- 在有限的范围内，可以对所有的值都进行试验和判断，从而找到满足条件的值
 - All_153.java 求三位的水仙花数
 - All_628.java 求9999以内的完全数
 - All_220.java 求9999以内的“相亲数”
- 遍试算法基本的模式
 - `for(;;){ if(); }`



- 迭代 (iterative algorithm)
- 是多次利用同一公式进行计算，每次将计算的结果再代入公式进行计算，从而逐步逼近精确解
 - Sqrt.java 自编一个函数求平方根
- 迭代的基本模式
 - `while() { x = f(x); }`



- 递归(recursive)就是一个过程调用过程本身。
 - 在递归调用中，一个过程执行的某一步要用到它的上一步(或上几步)的结果
- 示例
 - Fac.java 用递归方法求阶乘
 - CayleyTree.java 画出树
- 递归算法的基本模式
 - $f(n)\{ \quad f(n-1); \}$



- 回溯 (back-track)
- 回溯法也叫试探回溯法
 - 先选择某一可能的线索进行试探，每一步试探都有多种方式，将每一方式都一一试探，如果不符合条件就返回纠正，反复进行这种试探再返回纠正，直到得出全部符合条件的答案或是问题无解为止。
- 示例 Queen8.java 八皇后问题
- 回溯法的其本模式
 - `x++;` `if(...)` `x--;`



IDE的使用





Eclipse常用编辑功能

- **智能提示**（完成代码）：Alt+/
 - sysout + Alt+/ //生成System.out.print
 - main Alt+/ //生成main
- 代码自动完成：Tab 或回车
- 快速修复：Ctrl+1
- 导入所需包：Ctrl+Shift+O
- **代码自动插入**：Alt-Insert
 - 这个可以自动插入Getter-Setter方法的代码
 - 可以插入构造方法，可以插入override方法
- 自动



导航与书签

- Ctrl+O 打开类型
- F3 转至定义
- Ctrl+鼠标指向 转向定义或实现
- 显示成员：Ctrl+O
- Ctrl+F 查代、Ctrl+H 替换
- 【Alt+←】、【Alt+→】 上下位置
- 使用书签
- 格式化文档 Ctrl+Shift+F



修改与重构

- Alt+Shift+R 变量统一修改
- Ctrl+/, 注释间切换



NetBeans常用编辑功能

- 智能提示（完成代码）：ctrl+\
- 代码自动完成：Tab 或回车
 - `sout` + Tab //生成`System.out.print`
 - `psvm` + Tab //生成`main`
- 显示错误提示：alt + enter
- 导入所需包：ctrl+shift+i
- 代码自动插入：Alt-Insert
 - 这个可以自动插入Getter-Setter方法的代码
 - 可以插入构造方法，可以插入override方法



- Ctrl+O 打开类型
- Ctrl+B 转至源
- Ctrl-1 显示“项目”窗口
- 显示 Javadoc : Alt-F1
- Ctrl+F 查代、 Ctrl+H 替换
- Ctrl+G 上一编辑位置
- 【Alt+←】、【Alt+→】 上下位置
- 使用书签
- 格式化文档 Alt+Shift+F



修改与重构

- Ctrl+R 变量统一修改，取消用esc
- Ctrl+Shift+C，注释间切换
- 显示文档，Ctrl+Shift+Space
- 安装JDK源代码 工具--Java平台--源及javaDoc



- F5 开始调试主项目
- 运行主项目：F6
- 运行文件：Shift-F6
- * 设置断点：Ctrl-F8
- * 调试主项目：Ctrl-F5
- * 逐步调试：F7
- Application应用程序的参数args的设置，在Build->Set Main Projects Configuration
- 程序运行快捷键F6