数据类型：

整形：byte(8 **-128（-2^7） 127（2^7-1） 0**)

short (16 **-32768（-2^15）32767（2^15 - 1）0**)

int(32 **-2,147,483,648（-2^31）2,147,483,647（2^31 - 1）0**)

long(64  **-2,147,483,648（-2^31） 2,147,483,647（2^31 - 1） 0**)

char( 16 位 Unicode 字符 **\u0000** 0 **\uffff** 65535 char 数据类型可以储存任何字符)

浮点类型:

Float:(单精度、32位、符合IEEE 754标准的浮点数 **0.0f**) 储存大型浮点数组的时候可节省内存空间

Double:( 双精度、64 位、符合 IEEE 754 标准的浮点数 **0.0d**) 浮点数的默认类型为 double 类型

布尔类型:

Boolean: (表示一位的信息 **false**)

基本语法：

访问控制：

public private protect



循环条件：if while for break continue

异常：基本异常类型，全局异常处理机制

反射：

原理及使用

获取内容 方法签名

构造器 Constructor<T> getConstructor(Class<?>... parameterTypes)

包含的方法 Method getMethod(String name, Class<?>... parameterTypes)

包含的属性 Field getField(String name)

包含的Annotation <A extends Annotation> A getAnnotation(Class<A> annotationClass)

内部类 Class<?>[] getDeclaredClasses()

外部类 Class<?> getDeclaringClass()

所实现的接口 Class<?>[] getInterfaces()

修饰符 int getModifiers()

所在包 Package getPackage()

类名 String getName()

简称 String getSimpleName()

判断内容 方法签名

注解类型? boolean isAnnotation()

使用了该Annotation修饰? boolean isAnnotationPresent(Class<? extends Annotation> annotationClass)

匿名类? boolean isAnonymousClass()

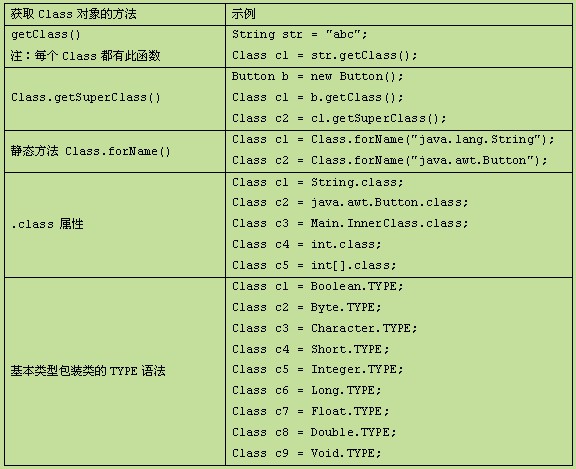
数组? boolean isArray()

枚举? boolean isEnum()

原始类型? boolean isPrimitive()

接口? boolean isInterface()

obj是否是该Class的实例 boolean isInstance(Object obj)



**获取某个类的构造方法**：

Constructor getConstructor(Class[] params)根据构造函数的参数，返回一个具体的具有public属性的构造函数

Constructor getConstructors() 返回所有具有public属性的构造函数数组

Constructor getDeclaredConstructor(Class[] params) 根据构造函数的参数，返回一个具体的构造函数（不分public和非public属性）

Constructor getDeclaredConstructors() 返回该类中所有的构造函数数组（不分public和非public属性）

**获取成员方法：**

Method getMethod(String name, Class[] params) 根据方法名和参数，返回一个具体的具有public属性的方法

Method[] getMethods() 返回所有具有public属性的方法数组

Method getDeclaredMethod(String name, Class[] params) 根据方法名和参数，返回一个具体的方法（不分public和非public属性）

Method[] getDeclaredMethods() 返回该类中的所有的方法数组（不分public和非public属性）

**获取类的成员变量:**

Field getField(String name) 根据变量名，返回一个具体的具有public属性的成员变量

Field[] getFields() 返回具有public属性的成员变量的数组

Field getDeclaredField(String name) 根据变量名，返回一个成员变量（不分public和非public属性）

Field[] getDelcaredFields() 返回所有成员变量组成的数组（不分public和非public属性）

面向对象：

类和对象：

创建与销毁 生命周期 容器等 equals()和hashcode()

枚举：

enum和接口常量

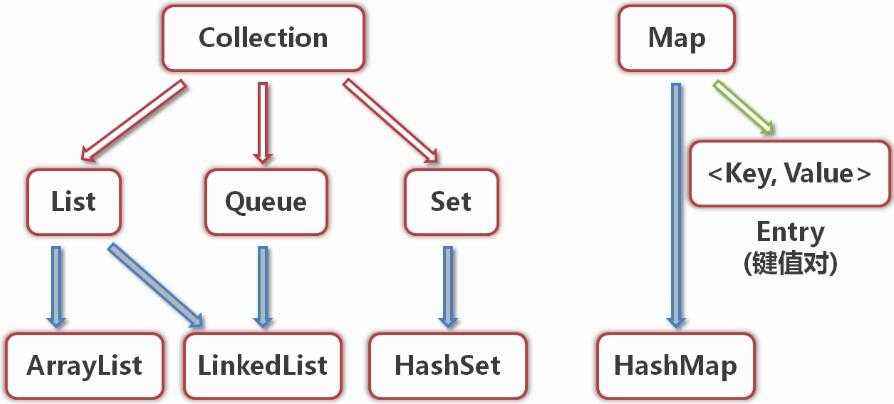
集合类：

List Map Set Queue 底层剖析

详解:



粗解：



List

Vector： Object[] 数组；

Arraylist： Object[] 数组；

LinkedList： 双向链表(JDK1.6 之前为双向循环链表，JDK1.7 取消了循环)

Set

HashSet（无序，唯一）: 基于 HashMap 实现的，底层采用 HashMap 来保存元素；

LinkedHashSet： LinkedHashSet 继承与 HashSet，并且其内部是通过 LinkedHashMap 来实现的，有点类似于 LinkedHashMap 的内部是基于 HashMap 实现；

TreeSet（有序，唯一）： 红黑树（自平衡的排序二叉树）

Map

HashMap： JDK1.8 前，HashMap 由数组+链表组成的，数组是 HashMap 的主体，链表则是主要为了解决哈希冲突而存在的（“拉链法”）。JDK1.8 以后在解决哈希冲突时有了较大的变化，当链表长度大于阈值（默认为8）时，会将链表转化为红黑树，以减少搜索时间；

LinkedHashMap： LinkedHashMap 继承自 HashMap，所以它的底层仍然是基于拉链式散列结构，即由数组和链表或红黑树组成。另外，LinkedHashMap 在上面结构的基础上，增加了一条双向链表，使得上面的结构可以保持键值对的插入顺序，同时通过对链表进行相应的操作，实现了访问顺序的相关逻辑。详细可以查看：《LinkedHashMap 源码详细分析（JDK1.8）》；

Hashtable： 数组+链表组成的，数组是 HashMap 的主体，链表则是主要为了解决哈希冲突而存在；

TreeMap： 红黑树（自平衡的排序二叉树）

继承和多态：

dddd

泛型：

泛型的本质是为了参数化类型（在不创建新的类型的情况下，通过泛型指定的不同类型来控制形参具体限制的类型）。也就是说在泛型使用过程中，操作的数据类型被指定为一个参数，这种参数类型可以用在类、接口和方法中，分别被称为泛型类、泛型接口、泛型方法。**泛型只在编译阶段有效**

参考：https://blog.csdn.net/s10461/article/details/53941091

内部类：dddd

接口：dddd

注解：

元注解: @Retention、 @Target、 @Document、 @Inherited和@Repeatable（JDK1.8加入）

**@Retention**

Retention英文意思有保留、保持的意思，它表示注解存在阶段是保留在源码（编译期），字节码（类加载）或者运行期（JVM中运行）。在@Retention注解中使用枚举RetentionPolicy来表示注解保留时期

@Retention(RetentionPolicy.SOURCE)，注解仅存在于源码中，在class字节码文件中不包含

@Retention(RetentionPolicy.CLASS)， 默认的保留策略，注解会在class字节码文件中存在，但运行时无法获得

@Retention(RetentionPolicy.RUNTIME)， 注解会在class字节码文件中存在，在运行时可以通过反射获取到

如果我们是自定义注解，则通过前面分析，我们自定义注解如果只存着源码中或者字节码文件中就无法发挥作用，而在运行期间能获取到注解才能实现我们目的，所以自定义注解中肯定是使用 @Retention(RetentionPolicy.RUNTIME)

**@Target**

Target的英文意思是目标，这也很容易理解，使用@Target元注解表示我们的注解作用的范围就比较具体了，可以是类，方法，方法参数变量等，同样也是通过枚举类ElementType表达作用类型

@Target(ElementType.TYPE) 作用接口、类、枚举、注解

@Target(ElementType.FIELD) 作用属性字段、枚举的常量

@Target(ElementType.METHOD) 作用方法

@Target(ElementType.PARAMETER) 作用方法参数

@Target(ElementType.CONSTRUCTOR) 作用构造函数

@Target(ElementType.LOCAL\_VARIABLE)作用局部变量

@Target(ElementType.ANNOTATION\_TYPE)作用于注解（@Retention注解中就使用该属性）

@Target(ElementType.PACKAGE) 作用于包

@Target(ElementType.TYPE\_PARAMETER) 作用于类型泛型，即泛型方法、泛型类、泛型接口 （jdk1.8加入）

@Target(ElementType.TYPE\_USE) 类型使用.可以用于标注任意类型除了 class （jdk1.8加入）

一般比较常用的是ElementType.TYPE类型

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.TYPE)

public @interface MyTestAnnotation {

}

**@Documented**

Document的英文意思是文档。它的作用是能够将注解中的元素包含到 Javadoc 中去。

**@Inherited**

Inherited的英文意思是继承，但是这个继承和我们平时理解的继承大同小异，一个被@Inherited注解了的注解修饰了一个父类，如果他的子类没有被其他注解修饰，则它的子类也继承了父类的注解。

/\*\*自定义注解\*/

@Documented

@Inherited

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.TYPE)

public @interface MyTestAnnotation {

}

/\*\*父类标注自定义注解\*/

@MyTestAnnotation

public class Father {

}

/\*\*子类\*/

public class Son extends Father {

}

/\*\*测试子类获取父类自定义注解\*/

public class test {

public static void main(String[] args){

//获取Son的class对象

Class<Son> sonClass = Son.class;

// 获取Son类上的注解MyTestAnnotation可以执行成功

MyTestAnnotation annotation = sonClass.getAnnotation(MyTestAnnotation.class);

}

}

**@Repeatable**

Repeatable的英文意思是可重复的。顾名思义说明被这个元注解修饰的注解可以同时作用一个对象多次，但是每次作用注解又可以代表不同的含义。

**注解的本质就是一个Annotation接口**

参考：https://www.jianshu.com/p/9471d6bcf4cf

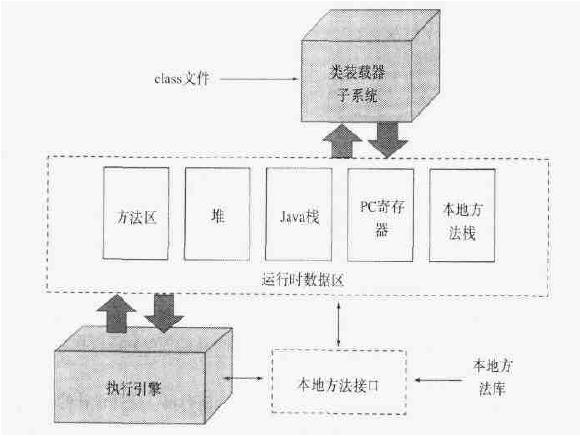
常用工具类：

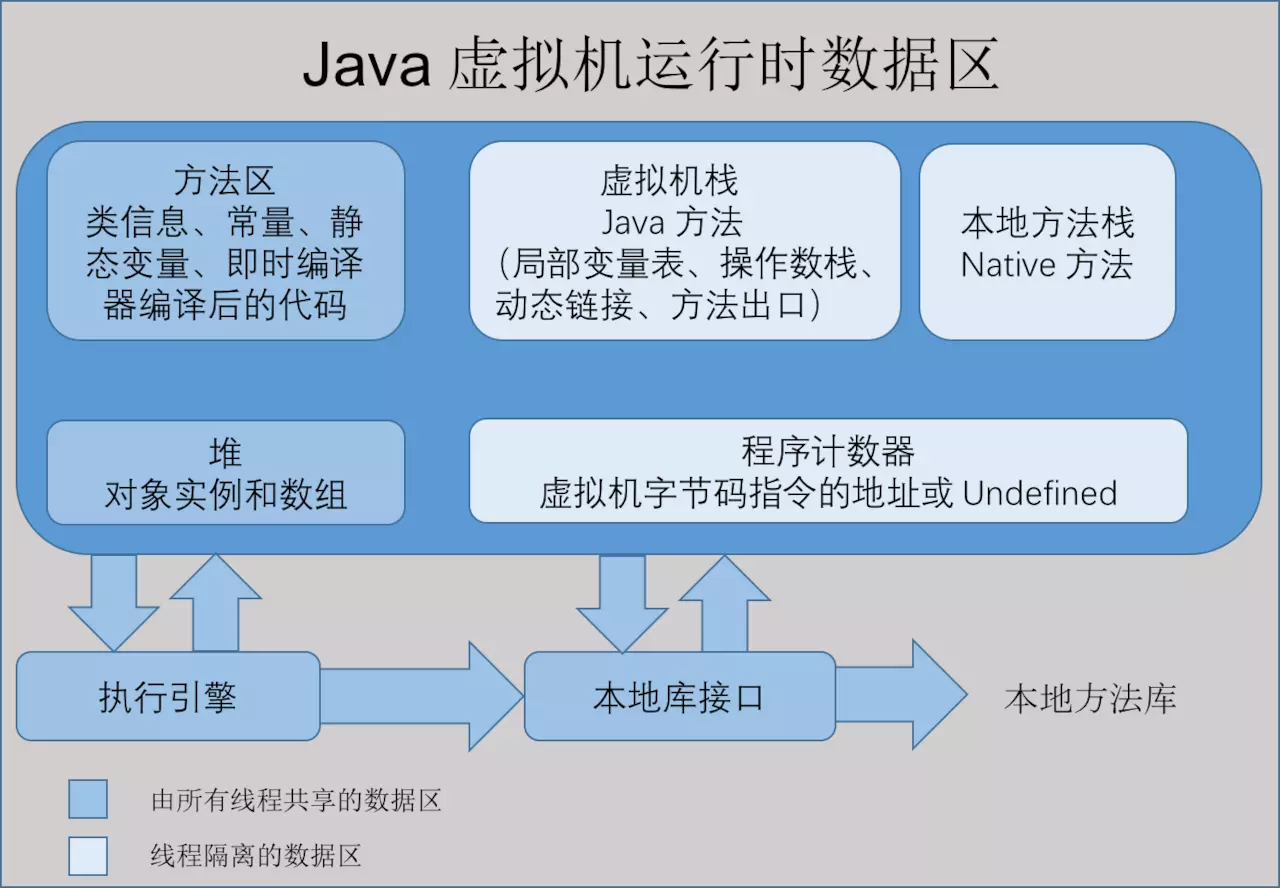
String,Date 以及相关优化类 <https://www.cnblogs.com/donleo123/p/14140627.html>

正则表达式：

手册：https://tool.oschina.net/uploads/apidocs/jquery/regexp.html

内存和JVM&GC：





类加载机制

加载class文件到内存：

加载：导入class文件

验证：检查class文件的正确性

准备：为类的静态变量分配内存

解析：

初始化：对静态变量和静态代码初始化

程序计数器

内存空间小，线程私有。字节码解释器工作是就是通过改变这个计数器的值来选取下一条需要执行指令的字节码指令，分支、循环、跳转、异常处理、线程恢复等基础功能都需要依赖计数器完成

如果线程正在执行一个 Java 方法，这个计数器记录的是正在执行的虚拟机字节码指令的地址；如果正在执行的是 Native 方法，这个计数器的值则为 (Undefined)。此内存区域是唯一一个在 Java 虚拟机规范中没有规定任何 OutOfMemoryError 情况的区域。

Java 虚拟机栈

线程私有，生命周期和线程一致。描述的是 Java 方法执行的内存模型：每个方法在执行时都会床创建一个栈帧(Stack Frame)用于存储局部变量表、操作数栈、动态链接、方法出口等信息。每一个方法从调用直至执行结束，就对应着一个栈帧从虚拟机栈中入栈到出栈的过程。

局部变量表：存放了编译期可知的各种基本类型(boolean、byte、char、short、int、float、long、double)、对象引用(reference 类型)和 returnAddress 类型(指向了一条字节码指令的地址)

StackOverflowError：线程请求的栈深度大于虚拟机所允许的深度。

OutOfMemoryError：如果虚拟机栈可以动态扩展，而扩展时无法申请到足够的内存。

本地方法栈

区别于 Java 虚拟机栈的是，Java 虚拟机栈为虚拟机执行 Java 方法(也就是字节码)服务，而本地方法栈则为虚拟机使用到的 Native 方法服务。也会有 StackOverflowError 和 OutOfMemoryError 异常。

Java 堆

对于绝大多数应用来说，这块区域是 JVM 所管理的内存中最大的一块。线程共享，主要是存放对象实例和数组。内部会划分出多个线程私有的分配缓冲区(Thread Local Allocation Buffer, TLAB)。可以位于物理上不连续的空间，但是逻辑上要连续。

OutOfMemoryError：如果堆中没有内存完成实例分配，并且堆也无法再扩展时，抛出该异常。

方法区

属于共享内存区域，存储已被虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等数据。

-Xms设置堆的最小空间大小。

-Xmx设置堆的最大空间大小。

-Xmn堆中新生代初始及最大大小（NewSize和MaxNewSize为其细化）。

-XX:NewSize设置新生代最小空间大小。

-XX:MaxNewSize设置新生代最大空间大小。

-XX:PermSize设置永久代最小空间大小。

-XX:MaxPermSize设置永久代最大空间大小。

-Xss设置每个线程的堆栈大小。

基本优化：

适当扩大eden区即-Xmn 避免full gc

最好只进行minor gc 效率高

eden:s0:s1

8:1:1

数据结构和算法：

数据结构：

线性结构：链表 Hash 数组 栈 堆

树性结构：树 二叉树

图：图的搜索遍历 （广度优先，深度优先）

算法：

搜索算法：

排序算法：

设计模式：

原则：

会实现：单例 工厂 代理

能看懂：适配器 观察者 模板

服务器与容器：

Tomcat:

Apache:

Nginx:

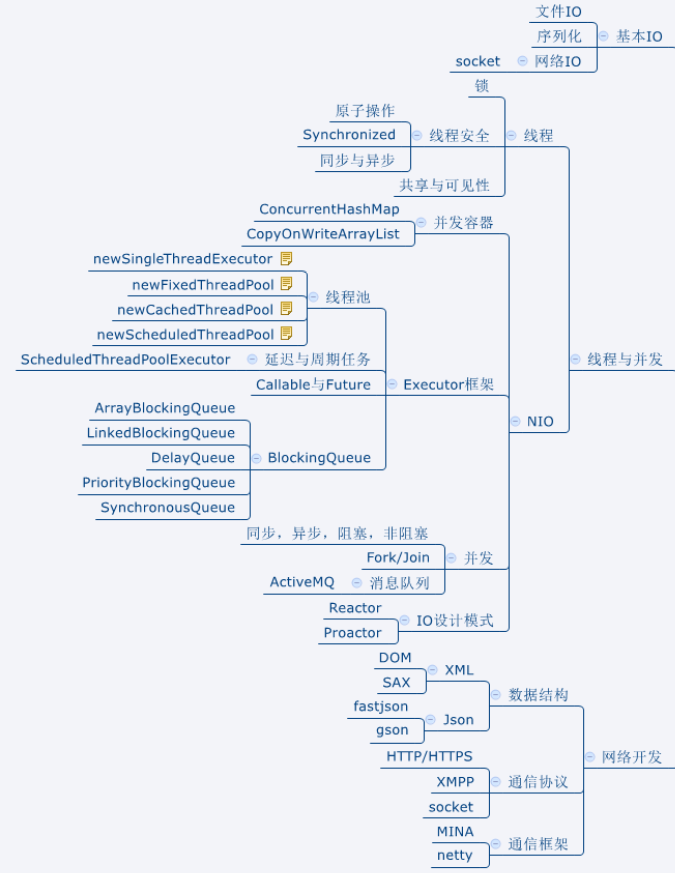
测试与日志：

单元测试：Junit

日志类：log4j slf4j logback MDC应用

编码与加解密：

I/O与基本网络编程:



计算机网络：

分层结构：

TCP/UDP：区别以及TCP的三次握手和四次握手

HTTP/HTTPS：区别 无状态 长连接 短连接

状态码：

Cookie和Session：

URL和URI：