# 反射

1.java语言中万事万物皆对象，只有静态成员和普通数据类型不是对象。

类是对象，类是java.lang.Class类的实例对象。

类的默认无参构造函数仅在未定义任何构造函数时存在。

2.Class.forName(“类的全称”)

1）不仅表示了类的类型，还代表了动态加载类

2）区分编译和运行，编译时刻加载类是静态加载类、运行时刻加载类是动态加载类。

new 创建对象是静态加载类，在编译时刻就需要加载所有可能使用到的类。通过动态加载类可以解决该问题。

基本的数据类型、void关键字都存在类类型。

3.为什么要用invoke方法，为什么要用制定方法名称调用方法，举个实际应用的案例：通过标准javaBean的属性名获取其属性值BeanUtil类

4.通过Class、Method来认识泛型的本质

# 设计模式

设计模式是一套被反复使用、多人知晓的、经过分类编目的、代码设计经验的总结，目的是为了可重用代码、让代码更容易被他人理解、保证代码的可靠性。

设计模式基本有23种：单例模式、抽象工厂模式、建造者模式、工厂模式、原型模式……

#### 单例模式（懒汉模式、饿汉模式）

应用场合：有些对象只需要一个就足够了；作用：保证整个应用程序中某个实例有且只有一个。

所谓“懒汉式”与“饿汉式”的区别，是**建立单例对象的时间不同**。

**1.“饿汉式”**是在不管你用的用不上，一开始就建立这个单例对象；

1）将构造方法私有化，不允许外部直接创建对象

2）自己创建类的唯一实例

3）提供一个用于获取实例的方法

**2.“懒汉式”**是在你真正用到的时候才去建这个单例对象；

与饿汉式区别在于第2）步懒汉式只是声明类的实例，真正创建是在要用的时候。

饿汉在加载类时比较慢，但运行时获得类的对象速度比较快，线程安全

懒汉式在加载类时比较快，但运行时获得类的对象速度比较慢，线程不安全

#### 工厂模式

概念：实例化对象，用工厂方法代替new操作。工厂模式包括工厂方法模式和抽象工厂模式。抽象工厂模式是工厂方法模式的拓展。

意图：定义一个接口来创建对象，但是让子类来决定那些类需要被实例化。工厂方法把实例化的方法推迟到子类中去实现。

什么情况下适合？

1）有一组类似的对象需要创建

2）在编码时不能预见需要创建哪种类的实例

3）系统需要考虑拓展性，不应依赖于产品类实例如何被创建、组合和表达细节。

简单工厂：简单工厂模式的工厂类一般是使用静态方法，通过接收的参数的不同来返回不同的对象实例。不修改代码的话，是无法扩展的。

抽象工厂：抽象工厂是应对产品族概念的。比如说，每个汽车公司可能要同时生产轿车，货车，客车，那么每一个工厂都要有创建轿车，货车和客车的方法。应对产品族概念而生，增加新的产品线很容易，但是无法增加新的产品。

#### 模板方法模式

概念：将定义的功能，推迟到子类中去实现。（重载？）

优点：封装性好，复用性好，屏蔽细节，便于维护

缺点：继承（java的单继承规则，限制了子类对所需其它功能的弹性拓展）

在模板类中定义一个**模板方法**，封装了所有子类共同遵循的算法框架，然后在该模板类中实现共同的操作，将所有有区别的操作都置为抽象方法。在各个子类中去分别重写自己特定的该抽象行为方法。

为了在子类中更加灵活的应用父类提供的方法，可以在父类中定义一个Hook方法，钩子函数，提供一个默认或空的实现，在**模板方法**中通过if判断是否实现相应操作。具体的子类可以通过重载自行决定是否挂钩以及如何挂钩。需要注意的是，JAVA不可以将所覆盖的方法的访问权限变得比父类的小。所以若父类中定 义方法默认为protected，子类中至少不能低于默认protected。

# IO

1. gbk编码中文占用2个字节，英文占用1个字节；utf-8编码中文占用3个字节，英文占用1个字节；java是双字节编码 utf-16be；当你的字节序列是某种编码时，这个时候想把字节序列变成字符串，也需要使用这种编码方式，否则会出现乱码。

文本文件 就是字节序列，可以是任意编码的字节序列，如果我们在中文机器上创建文本文件，那么该文件只认识ansi编码。

2.java.io.File类用于表示文件（目录）File类只用于表示文件（目录）的信息（名称大小等），不能用于文件内容的访问。File提供了直接返回File对象的方法,利用此方法结合递归可遍历出目标目录下的所有文件结构。

3.RandomAccessFile java提供的对文件内容的访问，既可以读文件，也可以写文件，且该类支持随机访问文件（可以访问文件的任意位置）。

RandomAccessFile有种模式，“rw”(读写) “r”(只读)

RandomAccessFile raf = new RandomAccessFile(file,”rw”) 文件指针，打开文件时指针在开头pointor = 0

写方法：raf.write(int) ---->只写一个字节（后8位），同时指针指向下一个位置，准备再次写入

读方法：int b = raf.read() ---->读一个字节

官方说明文件读写完成以后一定要关闭，不然会发生意想不到的错误

4.InputStream抽象了应用程序读取数据的方式,OutputStream抽象了应用程序写出数据的方式；EOF = End 读到-1就读到结尾；int b = in.read();读取一个字节无符号填充到int低8位，-1是EOF。

DataOutputStream/DataInputStream是对流的功能的扩展，可以更加方便的读取int，long，字符等类型的数据；DataOutputStream writeInt()/writeDouble()/writeUTF()

5.BufferedInputStream&BufferedOutputStream字节缓冲流，这两个流类为IO提供了带缓冲区的操作，一般打开文件进行写入或读取操作时，都会加上缓冲，这种流模式提高了IO的性能。

FileOutputStream相当于一滴一滴把水转移过去

DataOutputStream相当于一瓢一瓢把水转移过去

BufferedOutputStream相当于一瓢一瓢把水转移到桶中，再从桶转移到缸中

6.字符流

1）编码问题

2）认识文本和文本文件

Java的文本（char）是16为无符号整数，是字符unicode编码（双字节编码）

文件是byte byte byte……的数据序列

文本文件是文本（char）序列按照某种编码方案（utf-8/utf-16be/gbk）序列化为byte的存储形式

3）字符流（Reader/Writer）----- > 操作的是文本文件

字符的处理，一次处理一个字符

字符的底层仍然是基本的字节序列

InputStreamReader 完成byte流解析为char流，按照编码解析，参数为FileInputStream对象

OutputStreamWriter 提供char流到byte流，按照编码处理，参数为FileOutputStream对象

FileRader/FileWriter 分别为继承了InputStreamReader/OutputStreamWriter的两个类，创建对象时直接调用父类的构造方法，不能传入指定编码参数。

**InputStream实际上是FileInputStream的超类或者父类，这具体要看情况，File比较狭隘，如果是网络中通信的话就是用InputStream**

字符流的过滤器：BufferedReader ——>readLine 一次读一行

BufferedWriter ——>一次写一行

用法：BufferedReader br = new BufferedReader(new InputStreamReader(一个实例化的字符流对象InputStream/FileInputStream))

7.对象序列化与反序列化

序列化接口（Serializable）对象必须实现序列化接口才能进行序列化，否则将出现异常，这个接口没有任何方法，只是一个标准。

ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(file))；

Oos.writeObject(stu);向file中写入序列化后的stu对象。

Ois.readObject();读出stu对象。

# Socket

1.网络的基础知识

IP地址：主机唯一标志（类似于手机号码）（32位二进制组成）

协议：主机之间进行信息传输的格式标准

端口：不同应用程序之间通信的标志（常用端口号：http/80、ftp/21、telnet/23、）

TCP/IP协议：是以TCP和IP为基础的不同层次上多个协议的集合（物理层/数据链路层/网络层/传输层（TCP/IP）/应用层（Http/FTP/SMTP/Telnet））

IP和端口号组成了所谓的Socket，Socket是网络上运行的程序之间双向通信链路的终结点，是TCP和UDP的基础。

针对网络通信的不同层次，Java提供的网络功能有四大类：

InetAddress：用于标识网络上的硬件资源

URL：统一资源定位符 通过URL可以直接读取或者写入网络上的数据

Sockets：使用TCP协议实现网络通信的Socket相关的类

1）创建ServerSocket和Socket

2）打开连接到Socket的输入/输出流

3）按照协议对Socket进行读写操作

4）关闭输入输出流、关闭Socket

Datagram：使用UDP协议，将数据保存在数据报中，通过网络进行通信

UDP是面向Packet；TCP是面向Stream

2.TCP编程

**Server端：**

//1.创建一个服务器端的Socket,即ServerSocket，指定绑定的端口，并监听此端口  
ServerSocket serverSocket = new ServerSocket(8888);  
//2.调用accept()方法等待客户端连接  
System.out.println("\*\*\*\*\*服务即将启动，等待客户端的连接\*\*\*\*\*");  
Socket socket = serverSocket.accept();  
//3.获取输入流，读取客户端提交的登陆信息  
InputStream is = socket.getInputStream();

**Client端：**

//1.创建一个Socket，指定服务器地址和端口  
Socket socket = new Socket("127.0.0.1",8888);  
//2.获取输出流，用于向服务器端发送登录信息  
OutputStream os = socket.getOutputStream();//字节输出流

原则：server端必须优于client端先启动，阻塞8888端口，等待客户端的请求

3.UDP编程

**Server端：**

//1.创建服务器端DatagramSocket，指定端口号  
DatagramSocket socket = new DatagramSocket(8800);  
System.out.println("\*\*\*\*\*\*服务端已经启动，等待客户端请求\*\*\*\*\*");  
byte[] buf = new byte[1024];//创建字节数组，指定接收的数据包大小  
DatagramPacket packet = new DatagramPacket(buf,buf.length);  
//3.调用receive()方法开始监听客户端发送来的请求  
socket.receive(packet);//此方法在接收到数据报之前会一直阻塞(将接收到的信息存入数据报)

//4.读取数据  
String info = new String(buf,0,packet.getLength());  
System.out.println("我是服务器；客户端说："+info);

**Client端：**

//1.定义服务器的地址、端口号、数据  
try {  
 InetAddress address = InetAddress.*getByName*("127.0.0.1");  
 int port = 8800;  
 byte[] req ="用户名：admin;密码：123".getBytes();  
 //2.创建数据报，包含发送的数据信息  
 DatagramPacket packet = new DatagramPacket(req,req.length,address,port);  
 //3.创建DatagramSocket对象，发送数据报  
 socket = new DatagramSocket();  
 socket.send(packet);

4.总结

1）使用多线程时应考虑设置线程优先级，否则可能导致运行速度过慢

2）对于同一个socket如果关闭了输出流，则与该输出流关联的socket也会被关闭，所以一般不用关闭输出流，直接关闭socket即可

3）TCP数据传输过程中常需要传递的是对象，这时可以使用ObjectOutputStream 来进行对象序列化的传输。同样传递文件，使用文件输入输出流读出后写入到输出流。

# 多线程

1.进程：程序的执行过程（动态的概念），持有资源（共享内存，共享文件）和线程（如QQ，运行时可以开启多窗口聊天、收发文件、等）

线程：是系统中最少的执行单元，进程拥有多个线程，线程共享进程的资源。

2.java.lang包中的class Thread 或interface Runnable都有一个共同的方法run()方法

线程的创建：Thread()/Thread(String name)/Thread(Runnable target)/Thread(Runnable target , String name)

线程的方法：void start()/static void sleep(long milis)/static void sleep(long millis , int nanos)/void join()/void join(long millis)/void join(long millis , int nanos)/static void yield()

获取线程引用：static Thread currentThread()

3.stop方法，会让目标线程突然终止，发生意想不到的错误，现已被废弃，不建议使用。

4.正确的停止线程的方法：使用volatile设置停止的标志，在外部线程控制该线程结束。

5.interrupt的初衷并不是停止线程，他跟volatile作用类似，也是对线程设置一个退出标志，在通过对目标线程调用isInterrupted()方法作出判断，来判断该线程状态是否应该被终止，而在对该线程进行sleep休眠操作的时候使用Interrupted方法表示清除打断标志，使终止线程的目的失效，综合所述，使用volatile设置停止标志是比较好的方法。

6.线程的互斥：synchronized

线程的同步：wait/notifyAll

7.扩展：Java Memory Mode 、Locks&Condition 、 线程安全性/原子性与可见性 、并发编程工具：线程池ExcutorService/Callable&Future/BlockingQueue

8.几种Java线程池的分析和使用：

优势：a)降低资源消耗，通过重复利用已创建的线程降低线程创建和销毁造成的消耗

b)提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。

c)提高线程的可管理性，线程是稀缺资源，如果无限制的创建，不仅会消耗系统资源，还会降低系统的稳定性，使用线程池可以进行统一的分配，调优和监控。

Java的四种线程池：

1. newCachedThreadPool:创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。线程池为无限大，当执行第二个任务时，会复用执行第一个任务的线程，而不用每次新建线程。
2. newFixedThreadPool:创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。
3. newScheduledThreadPool:创建一个定长线程池，支持定时及周期性任务执行。
4. newSingleThreadPool:创建一个单线程化的线程池，它会用唯一的工作线程来执行任务，保证所有任务按照指定顺序执行。

# JDBC：

1. 在使用jdbc查询的时候，构建sql语句时，注意在拼接sql语句的时候两端留有空格，否则字符之间相连会无法识别。
2. 数据库字段中定义了sql的Date类型时，在模型中使用util包的Date型，插入时转成sql类型。
3. 数据库初始化和连接工作可以放在static{}静态块里面进行，这样每次需要使用，直接通过该工具类调用在静态块里生成的Connection实例就可以了。

# 概念：

##### 长连接与短连接

长连接是指客户端和服务端之间始终建立着一个通信连接，在连接没有中断之前，可随时通信，如tcp/udp协议。

短连接指双方有数据交互时，就建立一个连接，数据发送完成就断开连接，如http协议。

长连接的工作原理称作推送、短连接工作原理称作轮询。

socket是一个针对[TCP](https://www.baidu.com/s?wd=TCP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvPjfYnhwhmH6kuH-9PWnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHbYPHRvn1R3n10zPjcvnW6vr0" \t "_blank)和[UDP](https://www.baidu.com/s?wd=UDP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvPjfYnhwhmH6kuH-9PWnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHbYPHRvn1R3n10zPjcvnW6vr0" \t "_blank)编程的借口，你可以借助它建立[TCP](https://www.baidu.com/s?wd=TCP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvPjfYnhwhmH6kuH-9PWnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHbYPHRvn1R3n10zPjcvnW6vr0" \t "_blank)连接等等。而TCP和[UDP](https://www.baidu.com/s?wd=UDP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvPjfYnhwhmH6kuH-9PWnv0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHbYPHRvn1R3n10zPjcvnW6vr0" \t "_blank)协议属于传输层 而http是个应用层的协议，它实际上也建立在TCP协议之上。

##### 堆、栈与常量池

堆：存放所有new出来的对象；

栈：存放基本类型的变量数据和对象的引用，对象（new出来的对象）本身并不存在栈中，而是存放在堆中或者常量池中（字符串常量对象存放在常量池中）；

常量池：存放基本类型常量和字符串常量。

对于栈和常量池中的对象可以共享，对于堆中的对象不可以共享。栈中的数据大小和生命周期是可以确定的，当没有引用指向数据时，这个数据就会自动消失。堆中的对象的由垃圾回收器负责回收，因此大小和生命周期不需要确定，具有很大的灵活性。

而对于字符串来说，其对象的引用都是存储在栈中的，如果是编译期已经创建好（即指用双引号定义的）的就存储在常量池中，如果是运行期（new出来的对象）则存储在堆中。对于equals相等的字符串，在常量池中是只有一份的，在堆中则有多份。

**程序**运行时**，除非手动向常量池中添加常量(比如调用intern方法)，否则jvm不会自动添加常量到常量池。**

# 集合

##### List

1.List的contains方法，返回一个boolean值，判断List集合中是否存在目标值。

2.list.remove(Object object)可以直接删除list中的object对象。

3.list.add(index,object)可以直接向list中指定位置添加object对象。

# Tips

1. <<n 左移位运算符等于\*2^n >>n右移位运算符等于/ 2^n , 但前者效率均高于后者。

2. jav中使用子类对象去实例化父类接口，原因是面向接口编程，提高代码维护灵活性及移植性！

3.不必要的时候尽量避免使用Threa.join方法（例如在点击事件中调用）。因为可能会阻塞主线程的操作，造成错误

4.java中Integer默认值为0，没有初始化之前不能进行运算操作。

5.嵌套List类在实例化时，需要在上级构方法中实例化其中的List属性，否则使用该List属性会报错。

6.else if与if的区别在于，使用else if成立后会隔断下一个else if，但if会全部顺序执行。

7.接口，定义一个未实现的方法，针对不同的实现只需要去实现该接口，然后通过调用接口，间接调用自定义的方法实现。

# Mysql 2003无法连接本地服务器终极解决办法

打开命令提示符（管理员权限），执行

net stop mysql

如果已经启动了MySQL，把它关闭，然后切换到MySQL的bin目录下，执行

sc delete mysql

删除已有的服务，执行

mysqld --install

手工安装服务，然后执行

net start mysql

启动MySQL服务，如果服务启动成功，执行

mysql -u root -p

会提示你输入密码，输入你的密码，然后就能连接进数据库了，执行

UPDATE mysql.user SET password=PASSWORD('新密码') WHERE user='root';

FLUSH PRIVILEGES;

修改root密码。

手动安装后，默认编码为客户端输入编码、数据库编码均为为gbk，系统编码为utf8、修改my.ini文件：保证编码不会出现错乱

* [mysql]
* default-character-set=utf8
* [mysqld]
* character-set-server=utf8