抽象方法需要子类重写，而静态的方法是无法被重写的

1. ****阐述静态变量和实例变量的区别。****  
   答：静态变量是被static修饰符修饰的变量，也称为类变量，它属于类，不属于类的任何一个对象，一个类不管创建多少个对象，静态变量在内存中有且仅有一个拷贝；实例变量必须依存于某一实例，需要先创建对象然后通过对象才能访问到它。静态变量可以实现让多个对象共享内存。
2. 是否可以从一个静态（static）方法内部发出对非静态（non-static）方法的调用？  
   答：不可以，静态方法只能访问静态成员，因为非静态方法的调用要先创建对象，在调用静态方法时可能对象并没有被初始化
3. 内部类必须在外部类实例化后才能实例化，一个内部类对象可以访问创建它的外部类对象的成员，包括私有成员
4. Java 中的final关键字有哪些用法？  
   答：(1)修饰类：表示该类不能被继承；(2)修饰方法：表示方法不能被重写；(3)修饰变量：表示变量只能一次赋值以后值不能被修改（常量）。
5. 创建对象时构造器的调用顺序是：先初始化静态成员，然后调用父类构造器，再初始化非静态成员，最后调用自身构造器。相关代码见mavenTest工程com.yx.mavenTest包下A,B,C类；

32、怎样将GB2312编码的字符串转换为ISO-8859-1编码的字符串？  
答：代码如下所示：

|  |  |
| --- | --- |
| 1  2 | String s1 = "你好";  String s2 = new String(s1.getBytes("GB2312"), "ISO-8859-1"); |
|  |  |

1. 一般来说，断言用于保证程序最基本、关键的正确性。断言检查通常在开发和测试时开启。为了保证程序的执行效率，在软件发布后断言检查通常是关闭的。断言不应该以任何方式改变程序的状态。简单的说，如果希望在不满足某些条件时阻止代码的执行，就可以考虑用断言来阻止它。
2. Error表示系统级的错误和程序不必处理的异常，是恢复不是不可能但很困难的情况下的一种严重问题；比如内存溢出，不可能指望程序能处理这样的情况；Exception表示需要捕捉或者需要程序进行处理的异常，是一种设计或实现问题；也就是说，它表示如果程序运行正常，从不会发生的情况。
3. 用递归编写程序时一定要牢记两点：1. 递归公式；2. 收敛条件（什么时候就不再继续递归）。
4. try{}里有一个return语句，那么紧跟在这个try后的finally{}里的代码会不会被执行，什么时候被执行，在return前还是后?  
   答：会执行，在方法返回调用者前执行。在finally中改变返回值的做法是不好的，因为如果存在finally代码块，try中的return语句不会立马返回调用者，而是记录下返回值待finally代码块执行完毕之后再向调用者返回其值，然后如果在finally中修改了返回值，就会返回修改后的值。显然，在finally中返回或者修改返回值会对程序造成很大的困扰*，*
5. finally为确保一段代码不管发生什么异常状况都要被执行；try语句可以嵌套，每当遇到一个try语句，异常的结构就会被放入异常栈中，直到所有的try语句都完成。如果下一级的try语句没有对某种异常进行处理，异常栈就会执行出栈操作，直到遇到有处理这种异常的try语句或者最终将异常抛给JVM。
6. 列出一些你常见的运行时异常？  
   答：  
   - ArithmeticException（算术异常）  
   - ClassCastException （类转换异常）  
   - IllegalArgumentException （非法参数异常）  
   - IndexOutOfBoundsException （下标越界异常）  
   - NullPointerException （空指针异常）  
   - SecurityException （安全异常）
7. ArrayList 和Vector都是使用数组方式存储数据，它们都允许直接按序号索引元素，但是插入元素要涉及数组元素移动等内存操作，所以索引数据快而插入数据慢，Vector中的方法由于添加了synchronized修饰，因此Vector是线程安全的容器，但性能上较ArrayList差，因此已经是Java中的遗留容器；LinkedList使用双向链表实现存储，形成一个可以按序号索引的线性结构，与数组的连续存储方式相比，内存的利用率更高，按序号索引数据需要进行前向或后向遍历，但是插入数据时只需要记录本项的前后项即可，所以插入速度较快，由于ArrayList和LinkedListed都是非线程安全的，如果遇到多个线程操作同一个容器的场景，则可以通过工具类Collections中的synchronizedList方法将其转换成线程安全的容器后再使用。
8. Collection和Collections的区别？  
   答：Collection是一个接口，它是Set、List等容器的父接口；Collections是个一个工具类，提供了一系列的静态方法来辅助容器操作，这些方法包括对容器的搜索、排序、线程安全化等等。
9. TreeMap和TreeSet在排序时如何比较元素？Collections工具类中的sort()方法如何比较元素？  
   答：TreeSet要求存放的对象所属的类必须实现Comparable接口，该接口提供了比较元素的compareTo()方法，当插入元素时会回调该方法比较元素的大小。TreeMap要求存放的键值对映射的键必须实现Comparable接口从而根据键对元素进行排序。Collections工具类的sort方法有两种重载的形式，第一种要求传入的待排序容器中存放的对象比较实现Comparable接口以实现元素的比较；第二种不强制性的要求容器中的元素必须可比较，但是要求传入第二个参数，参数是Comparator接口的子类型（需要重写compare方法实现元素的比较），相当于一个临时定义的排序规则，其实就是通过接口注入比较元素大小的算法，也是对回调模式的应用。前者在定义对象属性时就重写compareTo方法，short方法则需要传入要比较的元素容器和要比较的对象Comparator<>,之后重写compare方法。
10. sleep()方法（休眠）是线程类（Thread）的静态方法，调用此方法会让当前线程暂停执行指定的时间，将执行机会（CPU）让给其他线程，但是对象的锁依然保持，因此休眠时间结束后会自动恢复，线程回到就绪状态；wait()是Object类的方法，调用对象的wait()方法导致当前线程放弃对象的锁（线程暂停执行），进入对象的等待池（wait pool），只有调用对象的notify()方法（或notifyAll()方法）时才能唤醒等待池中的线程进入等锁池（lock pool），如果线程重新获得对象的锁就可以进入就绪状态。线程是进程的一个实体，是CPU调度和分派的基本单位，
11. 线程的sleep()方法和yield()方法有什么区别？  
    答：① sleep()方法给其他线程运行机会时不考虑线程的优先级，因此会给低优先级的线程以运行的机会；yield()方法只会给相同优先级或更高优先级的线程以运行的机会；  
    ② 线程执行sleep()方法后转入阻塞（blocked）状态，而执行yield()方法后转入就绪（ready）状态；  
    ③ sleep()方法声明抛出InterruptedException，而yield()方法没有声明任何异常；  
    ④ sleep()方法比yield()方法（跟操作系统CPU调度相关）具有更好的可移植性。
12. 当一个线程进入一个对象的synchronized方法A之后，其它线程是否可进入此对象的synchronized方法B？synchronized关键字可以将对象或者方法标记为同步，  
    答：不能。其它线程只能访问该对象的非同步方法，同步方法则不能进入。因为非静态方法上的synchronized修饰符要求执行方法时要获得对象的锁，如果已经进入A方法说明对象锁已经被取走，那么试图进入B方法的线程就只能在等锁池（注意不是等待池哦）中等待对象的锁。
13. 请说出与线程同步以及线程调度相关的方法。  
    答：- wait()：使一个线程处于等待（阻塞）状态，并且释放所持有的对象的锁；  
    - sleep()：使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要处理InterruptedException异常；  
    - notify()：唤醒一个处于等待状态的线程，当然在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，而且与优先级无关；  
    - notityAll()：唤醒所有处于等待状态的线程，该方法并不是将对象的锁给所有线程，而是让它们竞争，只有获得锁的线程才能进入就绪状态；
14. 编写多线程程序有几种实现方式？一种是继承Thread类；另一种是实现Runnable接口。两种方式都要通过重写run()方法来定义线程的行为，推荐使用后者，因为Java中的继承是单继承，实现Callable接口，该接口中的call方法可以在线程执行结束时产生一个返回值。
15. 所谓的同步就是指阻塞式操作，而异步就是非阻塞式操作.
16. 启动一个线程是调用run()还是start()方法？  
    答：启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM 调度并执行，这并不意味着线程就会立即运行。run()方法是线程启动后要进行回调（callback）的方法.
17. newSingleThreadExecutor：创建一个单线程的线程池,如果这个唯一的线程因为异常结束，那么会有一个新的线程来替代它。newFixedThreadPool：创建固定大小的线程池。每次提交一个任务就创建一个线程，直到线程达到线程池的最大大小。newCachedThreadPool：创建一个可缓存的线程池。如果线程池的大小超过了处理任务所需要的线程，那么就会回收部分空闲（60秒不执行任务）的线程，当任务数增加时，此线程池又可以智能的添加新线程来处理任务。此线程池不会对线程池大小做限制，线程池大小完全依赖于操作系统（或者说JVM）能够创建的最大线程大小。 newScheduledThreadPool：创建一个大小无限的线程池。此线程池支持定时以及周期性执行任务的需求。newSingleThreadExecutor：创建一个单线程的线程池。此线程池支持定时以及周期性执行任务的需求。
18. 简述synchronized 和java.util.concurrent.locks.Lock的异同？Lock 能完成synchronized所实现的所有功能；主要不同点：Lock有比synchronized更精确的线程语义和更好的性能，而且不强制性的要求一定要获得锁。synchronized会自动释放锁，而Lock一定要求程序员手工释放，并且最好在finally 块中释放（这是释放外部资源的最好的地方）。
19. Blocked表示阻塞状态，阻塞状态又有多种情况，可能是因为调用wait()方法进入等待池，也可能是执行同步方法或同步代码块进入等锁池，或者是调用了sleep()方法或join()方法等待休眠或其他线程结束，或是因为发生了I/O中断。
20. Java中如何实现序列化，有什么意义？  
    答：序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决对象流读写操作时可能引发的问题（如果不进行序列化可能会存在数据乱序的问题）。要实现序列化，需要让一个类实现Serializable接口，该接口是一个标识性接口，标注该类对象是可被序列化的，然后使用一个输出流来构造一个对象输出流并通过writeObject(Object)方法就可以将实现对象写出（即保存其状态）；如果需要反序列化则可以用一个输入流建立对象输入流，然后通过readObject方法从流中读取对象。序列化除了能够实现对象的持久化之外，还能够用于对象的深度克隆.

53、编程实现文件拷贝**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**import** java.io.OutputStream;

**import** java.nio.ByteBuffer;

**import** java.nio.channels.FileChannel;

**public** **final** **class** MyUtil {

**private** MyUtil() {

**throw** **new** AssertionError();

    }

**public** **static** **void** fileCopy(String source, String target) **throws** IOException {

**try** (InputStream in = **new** FileInputStream(source)) {

**try** (OutputStream out = **new** FileOutputStream(target)) {

**byte**[] buffer = **new** **byte**[4096];

**int** bytesToRead;

**while**((bytesToRead = in.read(buffer)) != -1) {

                    out.write(buffer, 0, bytesToRead);

                }

            }

        }

    }

**public** **static** **void** fileCopyNIO(String source, String target) **throws** IOException {

**try** (FileInputStream in = **new** FileInputStream(source)) {

**try** (FileOutputStream out = **new** FileOutputStream(target)) {

                FileChannel inChannel = in.getChannel();

                FileChannel outChannel = out.getChannel();

                ByteBuffer buffer = ByteBuffer.allocate(4096);

**while**(inChannel.read(buffer) != -1) {

                    buffer.flip();

                    outChannel.write(buffer);

                    buffer.clear();

                }

            }

        }

    }

}

54、****写一个方法，输入一个文件名和一个字符串，统计这个字符串在这个文件中出现的次数。**import** java.io.BufferedReader;

**import** java.io.FileReader;

**public** **final** **class** MyUtil {

    // 工具类中的方法都是静态方式访问的因此将构造器私有不允许创建对象(绝对好习惯)

**private** MyUtil() {

**throw** **new** AssertionError();

    }

  /\*\*

     \* 统计给定文件中给定字符串的出现次数

     \*

     \* @param filename  文件名

     \* @param word 字符串

     \* @return 字符串在文件中出现的次数

     \*/

**public** **static** **int** countWordInFile(String filename, String word) {

**int** counter = 0;

**try** (FileReader fr = **new** FileReader(filename)) {

**try** (BufferedReader br = **new** BufferedReader(fr)) {

                String line = **null**;

**while** ((line = br.readLine()) != **null**) {

**int** index = -1;

**while** (line.length() >= word.length() && (index = line.indexOf(word)) >= 0) {

                        counter++;

                        line = line.substring(index + word.length());//截取剩下的内容

                    }

                }

            }

        } **catch** (Exception ex) {

            ex.printStackTrace();

        }

**return** counter;

    }

 }

使用JDBC操作数据库时，如何提升读取数据的性能？如何提升更新数据的性能？  
答：要提升读取数据的性能，可以指定通过结果集（ResultSet）对象的setFetchSize()方法指定每次抓取的记录数（典型的空间换时间策略）；要提升更新数据的性能可以使用PreparedStatement语句构建批处理，将若干SQL语句置于一个批处理中执行。PreparedStatement的一个缺点是，我们不能直接用它来执行in条件语句，解决方案：别进行单条查询——这样做性能很差，不推荐。使用存储过程——这取决于数据库的实现，不是所有数据库都支持。动态生成PreparedStatement——这是个好办法，但是不能享受PreparedStatement的缓存带来的好处了。

1. 什么是DAO模式？DAO（Data Access Object）顾名思义是一个为数据库或其他持久化机制提供了抽象接口的对象，在不暴露底层持久化方案实现细节的前提下提供了各种数据访问操作。在实际的开发中，应该将所有对数据源的访问操作进行抽象化后封装在一个公共API中。用程序设计语言来说，就是建立一个接口，接口中定义了此应用程序中将会用到的所有事务方法。在这个应用程序中，当需要和数据源进行交互的时候则使用这个接口，并且编写一个单独的类来实现这个接口，在逻辑上该类对应一个特定的数据存储。
2. JDBC能否处理Blob和Clob？  
   答： Blob是指二进制大对象（Binary Large Object），而Clob是指大字符对象（Character Large Objec），因此其中Blob是为存储大的二进制数据而设计的，而Clob是为存储大的文本数据而设计的。JDBC的PreparedStatement和ResultSet都提供了相应的方法来支持Blob和Clob操作。
3. 获得一个类的类对象有哪些方式？方法1：类型.class，例如：String.class  
   - 方法2：对象.getClass()，例如：”hello”.getClass()  
   - 方法3：Class.forName()，例如：Class.forName(“java.lang.String”)
4. 如何通过反射创建对象？  
   答：- 方法1：通过类对象调用newInstance()方法，例如：String.class.newInstance()  
   - 方法2：通过类对象的getConstructor()或getDeclaredConstructor()方法获得构造器（Constructor）对象并调用其newInstance()方法创建对象，例如：String.class.getConstructor(String.class).newInstance(“Hello”);
5. JDBC中如何进行事务处理？  
   答：Connection提供了事务处理的方法，通过调用setAutoCommit(false)可以设置手动提交事务；当事务完成后用commit()显式提交事务；如果在事务处理过程中发生异常则通过rollback()进行事务回滚。除此之外，从JDBC 3.0中还引入了Savepoint（保存点）的概念，允许通过代码设置保存点并让事务回滚到指定的保存点。
6. 如何通过反射获取和设置对象私有字段的值？答：可以通过类对象的getDeclaredField()方法字段（Field）对象，然后再通过字段对象的setAccessible(true)将其设置为可以访问，接下来就可以通过get/set方法来获取/设置字段的值了。

简述一下你了解的设计模式：单例模式（定义私有构造方法，定义一个私有静态类型的变量，定义一个公开静态的方法调用），工厂模式，代理模式，适配器模式：模板方法模式。

1. 用Java写一个冒泡排序。**import** java.util.Comparator;

**public** **interface** Sorter {

**public** <T **extends** Comparable<T>> **void** sort(T[] list);

**public** <T> **void** sort(T[] list, Comparator<T> comp);

}

**import** java.util.Comparator;

**public** **class** BubbleSorter **implements** Sorter {

  @Override

**public** <T **extends** Comparable<T>> **void** sort(T[] list) {

**boolean** swapped = **true**;

**for** (**int** i = 1, len = list.length; i < len && swapped; ++i) {

            swapped = **false**;

**for** (**int** j = 0; j < len - i; ++j) {

**if** (list[j].compareTo(list[j + 1]) > 0) {

                    T temp = list[j];

                    list[j] = list[j + 1];

                    list[j + 1] = temp;

                    swapped = **true**;

                }

            }

        }

    }

    @Override

**public** <T> **void** sort(T[] list, Comparator<T> comp) {

**boolean** swapped = **true**;

**for** (**int** i = 1, len = list.length; i < len && swapped; ++i) {

            swapped = **false**;

**for** (**int** j = 0; j < len - i; ++j) {

**if** (comp.compare(list[j], list[j + 1]) > 0) {

                    T temp = list[j];

                    list[j] = list[j + 1];

                    list[j + 1] = temp;

                    swapped = **true**;

                }

            }

        }

    }

}

64、用Java写一个折半查找。**import** java.util.Comparator;

**public** **class** MyUtil {

**public** **static** <T **extends** Comparable<T>> **int** binarySearch(T[] x, T key) {

**return** binarySearch(x, 0, x.length- 1, key);

   }

  // 使用循环实现的二分查找

**public** **static** <T> **int** binarySearch(T[] x, T key, Comparator<T> comp) {

**int** low = 0;

**int** high = x.length - 1;

**while** (low <= high) {

**int** mid = (low + high) >>> 1;

**int** cmp = comp.compare(x[mid], key);

**if** (cmp < 0) {

            low= mid + 1;

          }

**else** **if** (cmp > 0) {

            high= mid - 1;

          }

**else** {

**return** mid;

          }

      }

**return** -1;

   }

  // 使用递归实现的二分查找

**private** **static**<T **extends** Comparable<T>> **int** binarySearch(T[] x, **int** low, **int** high, T key) {

**if**(low <= high) {

**int** mid = low + ((high -low) >> 1);

**if**(key.compareTo(x[mid])== 0) {

**return** mid;

        }

**else** **if**(key.compareTo(x[mid])< 0) {

**return** binarySearch(x,low, mid - 1, key);

        }

**else** {

**return** binarySearch(x,mid + 1, high, key);

        }

      }

**return** -1;

   }

}计算中间位置时不应该使用(high+ low) / 2的方式，因为加法运算可能导致整数越界，这里应该使用以下三种方式之一：low + (high – low) / 2或low + (high – low) >> 1或(low + high) >>> 1（>>>是逻辑右移，是不带符号位的右移）

1. 什么是ORM？答：对象关系映射（Object-Relational Mapping，简称ORM）是一种为了解决程序的面向对象模型与数据库的关系模型互不匹配问题的技术；简单的说，ORM是通过使用描述对象和数据库之间映射的元数据（在Java中可以用XML或者是注解），将程序中的对象自动持久化到关系数据库中或者将关系数据库表中的行转换成Java对象，其本质上就是将数据从一种形式转换到另外一种形式。
2. UML是统一建模语言（Unified Modeling Language）的缩写,是一个支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持。使用UML可以帮助沟通与交流，辅助应用设计和文档的生成，还能够阐释系统的结构和行为。用例图（用来捕获需求，描述系统的功能，通过该图可以迅速的了解系统的功能模块及其关系）、类图（描述类以及类与类之间的关系，通过该图可以快速了解系统）、时序图（描述执行特定任务时对象之间的交互关系以及执行顺序，通过该图可以了解对象能接收的消息也就是说对象能够向外界提供的服务）。
3. 持久层设计要考虑的问题有哪些？你用过的持久层框架有哪些？
4. 持久,简单的说，就是将内存中的数据保存到关系型数据库、文件系统、消息队列等提供持久化支持的设备中。持久层就是系统中专注于实现数据持久化的相对独立的层面。

持久层设计的目标包括： 数据存储逻辑的分离，提供抽象化的数据访问接口。数据访问底层实现的分离，可以在不修改代码的情况下切换底层实现。资源管理和调度的分离，在数据访问层实现统一的资源调度（如缓存机制）。数据抽象，提供更面向对象的数据操作。持久层框架有：[Hibernate](http://hibernate.org/" \t "http://www.importnew.com/_blank),[MyBatis](http://blog.mybatis.org/" \t "http://www.importnew.com/_blank), [TopLink](http://www.oracle.com/technetwork/cn/middleware/toplink/overview/index.html" \t "http://www.importnew.com/_blank),[Guzz](https://code.google.com/p/guzz/" \t "http://www.importnew.com/_blank),[jOOQ](http://www.jooq.org/" \t "http://www.importnew.com/_blank),[Spring Data](http://projects.spring.io/spring-data/" \t "http://www.importnew.com/_blank),[ActiveJDBC](https://code.google.com/p/activejdbc/" \t "http://www.importnew.com/_blank)

1. 举一个多对多关联的例子，并说明如何实现多对多关联映射。  
   答：例如：商品和订单、学生和课程都是典型的多对多关系。可以在实体类上通过@ManyToMany注解配置多对多关联或者通过映射文件中的和标签配置多对多关联，但是实际项目开发中，很多时候都是将多对多关联映射转换成两个多对一关联映射来实现的。
2. 谈一下你对继承映射的理解。  
   答：继承关系的映射策略有三种：① 每个继承结构一张表（table per class hierarchy），不管多少个子类都用一张表。② 每个子类一张表（table per subclass），公共信息放一张表，特有信息放单独的表。③ 每个具体类一张表（table per concrete class），有多少个子类就有多少张表。第一种方式属于单表策略，其优点在于查询子类对象的时候无需表连接，查询速度快，适合多态查询；缺点是可能导致表很大。后两种方式属于多表策略，其优点在于数据存储紧凑，其缺点是需要进行连接查询，不适合多态查询。
3. @OneToMany注解的mappedBy属性有什么作用？@OneToMany用来配置一对多关联映射，但通常情况下，一对多关联映射都由多的一方来维护关联关系，例如学生和班级，应该在学生类中添加班级属性来维持学生和班级的关联关系（在数据库中是由学生表中的外键班级编号来维护学生表和班级表的多对一关系），如果要使用双向关联，在班级类中添加一个容器属性来存放学生，并使用@OneToMany注解进行映射，此时mappedBy属性就非常重要。
4. MyBatis中使用#和$书写占位符有什么区别？  
   答：#将传入的数据都当成一个字符串，会对传入的数据自动加上引号；$将传入的数据直接显示生成在SQL中。注意：使用$占位符可能会导致SQL注入，能用#的地方就不要使用$，写order by子句的时候应该用$而不是#。$符是直接拼成sql的 ，#符则会以字符串的形式 与sql进行拼接。#是预编译的
5. 解释一下MyBatis中命名空间（namespace）的作用。答：在项目中，可能存在大量的SQL语句，这时候为每个SQL语句起一个唯一的标识（ID）就变得并不容易了。为了解决这个问题，在MyBatis中，可以为每个映射文件起一个唯一的命名空间，这样定义在这个映射文件中的每个SQL语句就成了定义在这个命名空间中的一个ID。只要我们能够保证每个命名空间中这个ID是唯一的，即使在不同映射文件中的语句ID相同，也不会再产生冲突了。
6. ThreadLocal为解决多线程程序的并发问题提供了一种新的思路。ThreadLocal，顾名思义是线程的一个本地化对象，当工作于多线程中的对象使用ThreadLocal维护变量时，ThreadLocal为每个使用该变量的线程分配一个独立的变量副本，所以每一个线程都可以独立的改变自己的副本，而不影响其他线程所对应的副本。从线程的角度看，这个变量就像是线程的本地变量。ThreadLocal类非常简单好用，只有四个方法，能用上的也就是下面三个方法：- void set(T value)：设置当前线程的线程局部变量的值。- T get()：获得当前线程所对应的线程局部变量的值。- void remove()：删除当前线程中线程局部变量的值。
7. 选择使用Spring框架的原因: 非侵入式：不强制性的要求实现Spring框架中的接口或继承Spring框架中的类。 IoC容器:程序员再也不需要自己编写工厂、单例，

AOP（面向切面编程）:将所有的横切关注功能封装到切面（aspect）中，通过配置的方式将横切关注功能动态添加到目标代码上，进一步实现了业务逻辑和系统服务之间的分离。另一方面，有了AOP程序员可以省去很多自己写代理类的工作。

1. 事务管理：Spring以宽广的胸怀接纳多种持久层技术，并且为其提供了声明式的事务管理，在不需要任何一行代码的情况下就能够完成事务管理。
2. 快速失败(fail-fast)和安全失败(fail-safe)的区别是什么？Iterator的安全失败是基于对底层集合做拷贝，因此，它不受源集合上修改的影响。java.util包下面的所有的集合类都是快速失败的，而java.util.concurrent包下面的所有的类都是安全失败的。快速失败的迭代器会抛出ConcurrentModificationException异常，而安全失败的迭代器永远不会抛出这样的异常。
3. Comparable和Comparator接口是干什么的？列出它们的区别，只包含一个compareTo()方法的Comparable接口，包含compare()和equals()两个方法的Comparator接口。equals()方法需要一个对象作为参数，它用来决定输入参数是否和comparator相等。只有当输入参数也是一个comparator并且输入参数和当前comparator的排序结果是相同的时候，这个方法才返回true。
4. Java优先级队列(Priority Queue)，是一个基于优先级堆的无界队列，它的元素是按照自然顺序(natural order)排序的，不允许null值，不是线程安全的，入队和出队的时间复杂度是O(log(n))。
5. 大O符号（Big O notation）是用于描述函数渐进行为的数学符号，

|  |  |
| --- | --- |
| O(1) | 常数（阶，下同） |
| O(log\*n) | [迭代](http://baike.baidu.com/item/%E8%BF%AD%E4%BB%A3/8415523" \t "http://baike.baidu.com/_blank)[对数](http://baike.baidu.com/item/%E5%AF%B9%E6%95%B0/91326" \t "http://baike.baidu.com/_blank) |
| O(log n) | 对数 |
| O[(log n)^c] | 多对数 |
| O(n) | 线性，次线性 |
| O(n log n) | [线性对数](http://baike.baidu.com/item/%E7%BA%BF%E6%80%A7%E5%AF%B9%E6%95%B0/8010082" \t "http://baike.baidu.com/_blank)，或[对数](http://baike.baidu.com/item/%E5%AF%B9%E6%95%B0/91326" \t "http://baike.baidu.com/_blank)线性、拟线性、超线性 |
| O(n^2) | 平方 |
| O(n^c),Integer(c>1) | 多项式，有时叫作“代数”（阶） |
| O(c^n) | 指数，有时叫作“几何”（阶） |
| O(n!) | [阶乘](http://baike.baidu.com/item/%E9%98%B6%E4%B9%98/4437932" \t "http://baike.baidu.com/_blank)，有时叫做“组合”（阶） |

1. .Enumeration接口和Iterator接口的区别有哪些？Enumeration速度是Iterator的2倍，同时占用更少的内存。但是，Iterator远远比Enumeration安全，因为其他线程不能够修改正在被iterator遍历的集合里面的对象。同时，Iterator允许调用者删除底层集合里面的元素，这对Enumeration来说是不可能的。
2. HashSet是由一个hash表来实现的，因此，它的元素是无序的，TreeSet是由一个树形的结构来实现的，它里面的元素是有序的。
3. finalize()方法什么时候被调用，在释放对象占用的内存之前，垃圾收集器会调用对象的finalize()方法。一般建议在该方法中释放对象持有的资源。
4. 如果对象的引用被置为null，垃圾收集器是否会立即释放对象占用的内存？不会，在下一个垃圾回收周期中，这个对象将是可被回收的。
5. 串行(serial)收集器和吞吐量(throughput)收集器的区别是什么？吞吐量收集器使用并行版本的新生代垃圾收集器，它用于中等规模和大规模数据的应用程序。而串行收集器对大多数的小应用。
6. .在Java中，对象什么时候可以被垃圾回收？当对象对当前使用这个对象的应用程序变得不可触及的时候，这个对象就可以被回收了。
7. .JVM的永久代中会发生垃圾回收么？垃圾回收不会发生在永久代，如果永久代满了或者是超过了临界值，会触发完全垃圾回收(Full GC)。如果你仔细查看垃圾收集器的输出信息，就会发现永久代也是被回收的。这就是为什么正确的永久代大小对避免Full GC是非常重要的原因。
8. 什么时候使用CallableStatement？用来准备CallableStatement的方法是什么？

CallableStatement用来执行存储过程。存储过程是由数据库存储和提供的。提高sql语句执行效率，将业务逻辑加入存储过程，减少sql语句网络传输的时间，提供整体执行性能。

1. JSP有什么优点？下面列出了使用JSP的优点：JSP页面是被动态编译成Servlet的，因此，开发者可以很容易的更新展现代码。JSP页面可以被预编译。JSP页面可以很容易的和静态模板结合，包括：HTML或者XML，也可以很容易的和产生动态内容的代码结合起来。开发者可以提供让页面设计者以类XML格式来访问的自定义的JSP标签库。开发者可以在组件层做逻辑上的改变，而不需要编辑单独使用了应用层逻辑的页面。
2. JSP请求是如何被处理的？浏览器首先要请求一个以.jsp扩展名结尾的页面，发起JSP请求，然后，Web服务器读取这个请求，使用JSP编译器把JSP页面转化成一个Servlet类。需要注意的是，只有当第一次请求页面或者是JSP文件发生改变的时候JSP文件才会被编译，然后服务器调用servlet类，处理浏览器的请求。一旦请求执行结束，servlet会把响应发送给客户端。
3. InnoDB是事务型数据库的首选引擎，支持事务安全表（ACID），支持行锁定和外键；MyISAM存储引擎，是在Web、数据仓储和其他应用环境下最常使用的存储引擎之一。MyISAM拥有较高的插入、查询速度，但不支持事物。数据表主要用来插入和查询记录，则MyISAM引擎能提供较高的处理效率。

如果只是临时存放数据，数据量不大，并且不需要较高的数据安全性，可以选择将数据保存在内存中的Memory引擎，MySQL中使用该引擎作为临时表，存放查询的中间结果。

如果只有INSERT和SELECT操作，可以选择Archive，Archive支持高并发的插入操作，但是本身不是事务安全的。Archive非常适合存储归档数据，如记录日志信息可以使用Archive

1. mysql中myisam与innodb的区别，至少5点。

1>.InnoDB支持事物，而MyISAM不支持事物

2>.InnoDB支持行级锁，而MyISAM支持表级锁

3>.InnoDB支持MVCC, 而MyISAM不支持

4>.InnoDB支持外键，而MyISAM不支持

5>.InnoDB不支持全文索引，而MyISAM支持。

93、varchar与char的区别：char是一种固定长度的类型，varchar则是一种可变长度的类型。

94、问了innodb的事务与日志的实现方式。 （1）有多少种日志

错误日志：记录出错信息，也记录一些警告信息或者正确的信息

慢查询日志：设置一个阈值，将运行时间超过该值的所有SQL语句都记录到慢查询的日志文件中。 二进制日志：记录对数据库执行更改的所有操作

查询日志：记录所有对数据库请求的信息，不论这些请求是否得到了正确的执行。

1. MYSQL数据表在什么情况下容易损坏？ 服务器突然断电导致数据文件损坏。   
   强制关机，没有先关闭mysql 服务等
2. mysql里记录货币用什么字段类型好？NUMERIC和DECIMAL类型被MySQL实现为同样的类型，他们被用于保存值，该值的准确精度是极其重要的值。如salary DECIMAL(9,2) 9为总位数，2是小数点后两位。
3. Mysql中当我们使用‘='号时用‘AND'连接，用‘!='时用‘OR'连接。
4. 使用IFNULL()方法能使MySQL中的查询更加精确。IFNULL()方法将会测试它的第一个参数，若不为NULL则返回该参数的值，否则返回第二个参数的值。SELECT name, IFNULL(id,'Unknown') AS 'id' FROM taxpayer;如果id为空则返回Unknown。
5. JDBC通过Statement和PreparedStatement中的addBatch和executeBatch方法来支持批处理。
6. JDBC的RowSet是什么，有哪些不同的RowSet？RowSet用于存储查询的数据结果，和ResultSet相比，它更具灵活性。RowSet继承自ResultSet，因此ResultSet能干的，它们也能，而ResultSet做不到的，它们还是可以。RowSet接口定义在javax.sql包里。

RowSet对象默认是可滚动，可更新的，因此如果数据库系统不支持ResultSet实现类似的功能，可以使用RowSet来实现。提供了Java Bean的功能，可以通过settter和getter方法来设置和获取属性。

1. 常见的JDBC异常有哪些？SQLException——这是JDBC异常的基类。BatchUpdateException——当批处理操作执行失败的时候可能会抛出这个异常。DataTruncation——字段值由于某些非正常原因被截断了。
2. 乐观锁——只有当更新数据的时候才会锁定记录。 最常见的乐观锁是通过数据版本标识来实现的，读取数据时获得数据的版本号，更新数据时将此版本号加1，然后和数据库表对应记录的当前版本号进行比较，如果提交的数据版本号大于数据库中此记录的当前版本号则更新数据，否则认为是过期数据无法更新。悲观锁——从查询到更新和提交整个过程都会对数据记录进行加锁。
3. DDL语句用来定义数据库模式Create，Alter, Drop, Truncate, Rename都属于DDL语句，一般来说，它们是不返回结果的。DML语句用来操作数据库中的数据。select, insert, update, delete, call等，都属于DML语句。
4. java.util.Date包含日期和时间，而java.sql.Date只包含日期信息，而没有具体的时间信息。如果你想把时间信息存储在数据库里，可以考虑使用Timestamp或者DateTime字段。
5. 如何把图片或者原始数据插入到数据库中？

可以使用BLOB类型将图片或者原始的二进制数据存储到数据库里。

1. 联合索引特点：最左原则。
2. http是80端口，https是443端口。
3. Mybatis中Xml映射文件中，除了常见的select|insert|updae|delete标签之外，还有哪些标签？<resultMap>、<parameterMap>、<sql>、<include>、<selectKey>，加上动态sql的9个标签，trim|where|set|foreach|if|choose|when|otherwise|bind等，其中<sql>为sql片段标签，通过<include>标签引入sql片段，<selectKey>为不支持自增的主键生成策略标签。
4. Mybatis中通常一个Xml映射文件，都会写一个Dao接口与之对应，Dao接口里的方法，是不能重载的，因为是全限名（namespace）+方法名的保存和寻找策略。接口方法内的参数，就是传递给sql的参数，Dao接口的工作原理是JDK动态代理，Mybatis运行时会使用JDK动态代理为Dao接口生成代理proxy对象，代理对象proxy会拦截接口方法，转而执行MappedStatement所代表的sql，然后将sql执行结果返回。
5. Mybatis执行批量插入，能返回数据库主键列表吗？JDBC都能，Mybatis当然也能
6. Mybatis中<resultMap>标签，逐一定义列名和对象属性名之间的映射关系
7. Mybatis中如何执行批处理？使用BatchExecutor完成批处理
8. Mybatis有三种基本的Executor执行器，SimpleExecutor、ReuseExecutor、BatchExecutor。SimpleExecutor：每执行一次update或select，就开启一个Statement对象，用完立刻关闭Statement对象。ReuseExecutor：执行update或select，以sql作为key查找Statement对象，存在就使用，不存在就创建，用完后，不关闭Statement对象，而是放置于Map<String, Statement>内，供下一次使用。简言之，就是重复使用Statement对象。BatchExecutor：执行update（没有select，JDBC批处理不支持select），将所有sql都添加到批处理中（addBatch()），等待统一执行（executeBatch()），它缓存了多个Statement对象，每个Statement对象都是addBatch()完毕后，等待逐一执行executeBatch()批处理。与JDBC批处理相同。

作用范围：Executor的这些特点，都严格限制在SqlSession生命周期范围内。

114、Mybatis可以映射枚举类，不单可以映射枚举类，Mybatis可以映射任何对象到表的一列上。

115、Mybatis映射文件中，如果A标签通过include引用了B标签的内容，请问，B标签能否定义在A标签的后面，还是说必须定义在A标签的前面？虽然Mybatis解析Xml映射文件是按照顺序解析的，但是，被引用的B标签依然可以定义在任何地方，Mybatis都可以正确识别。

116、为什么说Mybatis是半自动ORM映射工具？它与全自动的区别在哪里？

Hibernate属于全自动ORM映射工具，使用Hibernate查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。而Mybatis在查询关联对象或关联集合对象时，需要手动编写sql来完成，所以，称之为半自动ORM映射工具。

117、SpringMVC的工作流程

1、用户发送请求至前端控制器DispatcherServlet

2、DispatcherServlet收到请求调用HandlerMapping处理器映射器。

3、处理器映射器找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。

4、DispatcherServlet调用HandlerAdapter处理器适配器

5、HandlerAdapter经过适配调用具体的处理器(Controller，也叫后端控制器)。

6、Controller执行完成返回ModelAndView

7、HandlerAdapter将controller执行结果ModelAndView返回给DispatcherServlet

8、DispatcherServlet将ModelAndView传给ViewReslover视图解析器

9、ViewReslover解析后返回具体View

10、DispatcherServlet根据View进行渲染视图（即将模型数据填充至视图中）。

11、DispatcherServlet响应用户

118、JSP page 指令： import 属性指定 JSP 页面转换成的 servlet 应该输入的包，contentType 属性设置 Content-Type 响应报头，标明即将发送到客户程序的文档的 MIME 类型，如果只想更改字符集，使用 pageEncoding 属性更为简单，session 属性控制页面是否参与 HTTP 会话，isELIgnored 属性控制的是：忽略（true）JSP 2.0 表达式语言（EL），还是进行正常的求值（false），buffer 属性指定 out 变量(类型为 JspWriter )使用的缓冲区的大小，autoFlush 属性控制当缓冲区充满之后，是应该自动清空输出缓冲区（默认），还是在缓冲区溢出后抛出一个异常（autoFlush="false"）。info 属性定义一个可以在 servlet 中通过 getServletInfo 方法获取的字符串。errorPage 属性用来指定一个 JSP 页面，由该页面来处理当前页面中抛出但未被捕获的任何异常，isErrorPage 属性表示当前页是否可以作为其他 JSP 页面的错误页面。extends 属性指定 JSP 页面所生成的 servlet 的超类（superclass ），language 属性的作用是指定页面使用的脚本语言。

119、与Servlet主动关联的是ServletConfig、ServletRequeset和ServletResponse，它们通过容器传递给Servlet。其中，Servlet初始化时使用ServletConfig，另外两个是在请求调用时传给Servlet，ServletConfig是一个Servlet的配置信息对象，在Servlet初始化时，Servlet容器用它将sevlet配置信息传递给Servlet.调用init()方法后，servlet可以从每个servlet获得一个ServletConfig对象，从web应用程序获得一个ServletContext对象。

