**1.package** org.homework.book.commons;

ImageUtil 中的UUID.randomUuid()唯一通用识别码

2.lastIndexOf（）:一个指定的字符串的最后出现时的位置

3.Lambda表达式：（1）作为方法参数列表（在具体的调用时，而不是在类构造方法之时设置，在类中方法构造之时仍然是（接口 变量名）），然后调用该方法时进行Lambda表达式的设置）

（2）只有一个抽象方法的接口

（3）只需要写出接口抽象方法参数再加上{}的方法实现

（4）无须写出方法名和返回值类型

4.事件监听器：（1）实现监听器接口（2）创建控件对象（3）调用addXxxListener方法来进行实现。

如果是有一个抽象参数的接口，可以使用Lambda表达式。

5.条件适配器：条件适配器是监听器接口的空实现，然后再创建类继承条件适配器，然后重写父类方法即可。

6.接口：（1）成员变量是静态常量（public static final），不能有普通成员变量；

（2）方法只能是抽象方法（public abstract），类方法（用static 而不用default）和默认方法（用default 而不用static）。

（3）不能有初始化和构造器。

抽象类：有抽象方法的类只能定义为抽象类，抽象类里可以没有抽象方法。

（1）抽象类可以包括普通方法；

（2）既可以定义普通变量，也可以定义静态常量。

（3）可以定义初始化和构造器，用来让子类继承创造对象；

7.异常从发生异常的方法逐渐向外传播，首先传给该方法的调用者，直到最后传给main方法，如果 main方法没有处理该异常，JVM会终止该程序，并打印跟踪栈信息。 （一行的信息包括被调用方法，并标明类名，方法名，及故障点对应的行）。

8.RunTime异常可用try…catch来进行处理，亦可以不用；

Checked异常要么用try…catch 来进行处理，要么throws 声明抛出异常；

两者的共同点：要么用try…catch就地解决；如果不用，则由方法的调用者处理（但checked必须在方法的调用者里用throws 或者try…catch；而RunTime则不用）。

注：无论是对于系统自己抛出的，还是对于通过throw抛出的。

9.在构造器重载中，如果B构造器的内容完全包含A构造器，则可以使用this关键字进行重载，同时必须作为构造器执行体的第一条语句。

10.软件开发里有一个规则：不要把相同的代码段书写两次以上。

11.当调用子类构造器初始化对象时，父类构造器总会在子类构造器之前执行；

Super 可以调用父类构造器，并且卸载第一行。

12.数组初始化分为静态初始化，和动态初始化

静态初始化：arrayName=new type[]{放一组数据}；

动态初始化：arrayName=new type[n]；

13.使用构造器先完成整个Java对象的状态初始化，然后将java对象返回给程序。

14.java类里可有4种成员：成员变量、方法、构造器、初始化块。

15.当创建java对象时，系统总是调用该类里定义的初始化块。

16. 类名：首字母大写，其他单词中首字母大写，其他小写  
方法名：首字母小写，其他单词中首字母大写，其他小写  
变量：与方法名规则同  
包名：全部小写

17.配置文件中不用；和“”的，而在属性提取中，使用双引号获得相关的driver,url,user,password。

18.对于数据库中的数据，是int型的用getInt，是String型的用getString,

是double型的用getDouble。

19.对于String类型的属性，在sql语句中需要加’’这样的字段。

20.能尽量使用try{}catch(){},就不要使用throws Exception ，因为一旦

用了第二者，随着不断调用，就要不断设置它，会很繁琐。

21.对于图书管理系统的建立，对于ui层调用service层，要在构造器里设置关于service的赋值初始值，在service 层调用dao层，需要设置构造器的dao的赋值。因此LoginFrame中：

BookDao *bookDao*=new BookImpl();  
ConcernDao *concernDao*=new ConcernImpl();  
TypeDao *typeDao*=new TypeImpl();  
  
  
 UserService *userService*;  
 BookService *bookService*=new BookServiceImpl(*concernDao*,*typeDao*,*bookDao*);  
 ConcernService *concernService*=new ConcernServiceImpl(*concernDao*);  
 TypeService *typeService*=new TypeServiceImpl(*typeDao*);

然后再调用new Panel(bookService,concernService,typeService).init();即可。

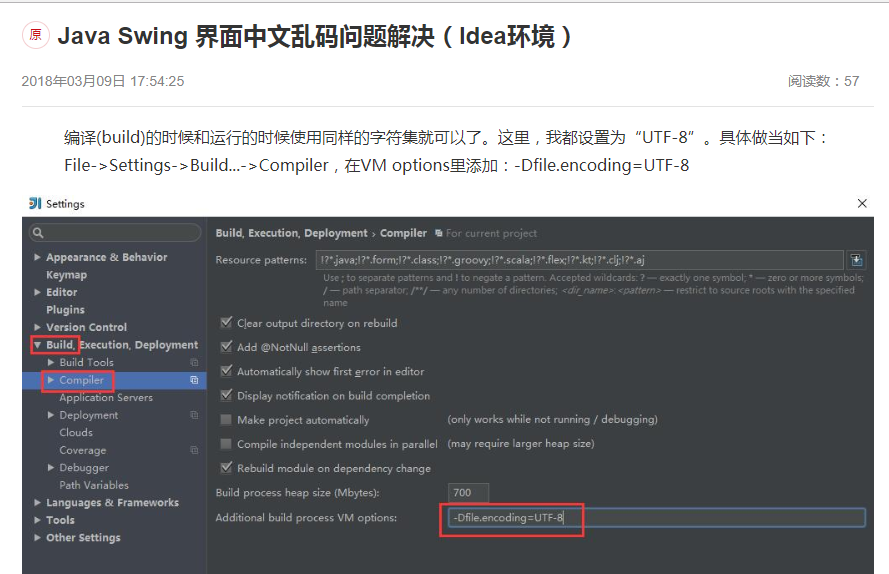
22.对于一个项目要进行日志的设立，对于方法的互相调用时，每个方法执行的时候可以设立日志，日志的设立最好用英文明显表示相关方法，可以便利的查询出是哪个地方出了问题，可以考虑用if-else 语句，对于出错密集型的地方可以多多设置。

22.对于数组的的初始化，如果是int,short,long,byte的初始化默认值，则为0，如果为float,double的初始化默认值0.0。引用型数组的初始化默认值为null。类也是一种引用的数据类型（引用变量在栈内存中，而实际变量在堆内存中）。

23.构造器B的代码完全包含构造器A的代码，则可以使用this(\*,\*)代码放到构造B中。构造器调用父类构造器时，可以使用super（\*，\*）。

24.编译时类型由声明该变量时使用的类型决定，运行时类型由实际赋给该变量的对象决定。编写Java程序时，引用变量只能调用它编译时类型的方法，而不能调用它运行时类型的方法，但是执行的仍然是其子类的方法。对于编译期间没有的方法，而运行时有的方法，这样调用会出错的。 但是对于引用变量来访问其包含的实例变量时，系统总是试图访问它编译时类型所定义的成员变量，而不是它运行时类型所定义的成员变量。接口调用方法时也是这样。

25.界面中文乱码问题



26.与构造器类似，创建一个Java对象时，先执行父类的初始化块，再执行父类的构造器，然后再执行子类的初始化块和子类的构造器。

27.在编程过程中，最好是使用面向接口编程，而不是面向实现类编程。可以降低耦合率。

28.系统永远先执行静态初始化块（从父类到子类），执行完后再从父到子，执行普通初始化块，构造器—>普通初始化块，构造器，这样的顺序。注：一旦类初始化成功之后，一直将会在虚拟机里存在，因此当第二次创建对象时，无须再对Leaf类进行初始化，也不会显示出来。

Super在构造器中位于第一句话；如果子类构造器没写super或者是super(),则

默认调用无参构造器，如果有super(\*)则是显示调用父类有参构造器；（注：因

此对于每个类最好加一个无参构造器。）this在本类里面的构造器就仅仅是调用。

静态成员无法调用非静态成员。因此super，this均不可出现在static中，因

为两个词都是指向对象（this本对象，super父对象）。

This和super不能同时出现在一个构造器里。因为this可以调用其他构造函数，而其他构造函数也必然会有super语句，如果一个构造函数里面存在着一个this和super语句，相当于有两个super语句。

This:普通的直接引用；

形参和成员名重复，则用this区分；

构造函数见上面第二段。

Super：普通的父类的直接引用；

子类中的成员变量或方法与父类中的成员变量或方法同名；

构造函数见第二段。

29.接口间继承也是用extends，而不是用implements 。

30.Java集合大致可分为Set、List、Queue和Map四种体系，其中Set代表无序，不可重复的集合；List代表有序、重复的集合；而Map则代表具有映射关系的集合，queue代表一种队列集合的实现。运用集合的优势在于避免了数组长度的不可变化的缺点，也不能保存具有映射关系的集合。但是集合可以随意改变长度。

31.无论是数组名，还是创建对象时的变量，都是引用变量，其实就是指针，相当于地址传递，是可以改变相关变量的值的。

**泛型**

1.泛型有接口、类、方法，这将在声明变量、创建对象、调用方法时动态的指定。

2.并不存在泛型类，所以也就没有泛型父子类。

3.如果没有泛型，所有对象进入集合都会忘记其类型，取出后都会变成Object类型。其中类型通配符是所有泛型的父类，类型是Object。类型通配符的集合不能添加元素，因为程序无法确定集合中元素的类型。

4.Collection<T extends E> 是错的；Collection<？ extends E> 是对的；

Method (Collection<T > name )在这种情况下是可以使用的。

对于泛型方法开头不可以使用<T ,T extends E>这种情况，但是可以使用<T extends E> 这种方式。

5.当创建带泛型声明的自定义类，为该类定义构造器时，构造器名还是原来的类名，不要增加泛型。但是如果不带泛型类则可以构建带泛型的构造器。