第一章

1. 软件的定义：程序+数据+文档
2. 测试驱动的开发;要求在编写某个功能代码之前先编写测试代码
3. 面向对象三大特性：封装、继承、多态
4. 面向对象与面向过程的区别：

面向过程：以分步骤的方式解决问题，用方法来组织代码。

面向对象：以分类的方式解决问题，用类组织代码

二者关系：相辅相成，并不是对立的

1. 面向对象思想的优势：
   1. 解耦、封装：隐藏对象的属性和具体细节，仅对外公开访问方法，增强了安全性和简化编程。
   2. 复用：根据新成员所属的类，复用之前的代码
   3. 多态：支持不同的方式实现同一个接口，这样可以将实现细节与上层对象分开。

第二章

1. 标识符：即变量名、类名、方法名等。
2. 标识符不能以数字开头
3. 类名首字母大写；常量名全部大写；变量名、对象名、方法名、包名使用驼峰原则(首字母小写、所有单词仅首字母大写)
4. 基本数据类型及其大小：byte（1）、short（2）、int（4）、long（8）、char（2）、float（4）、double（8）、boolaen（4）。
5. 引用数据类型：都是占4个字节，包括类、接口、数组。
6. java整形默认int，声明long类型需要在后面+l或者+L。
7. 浮点型默认double，舍命float类型需要在后面加F或者f。
   1. float f=3.15是错的，
   2. float f=3.15f是对的
8. 类型转换规则：低级别值赋给高级别变量时，完成自动转换

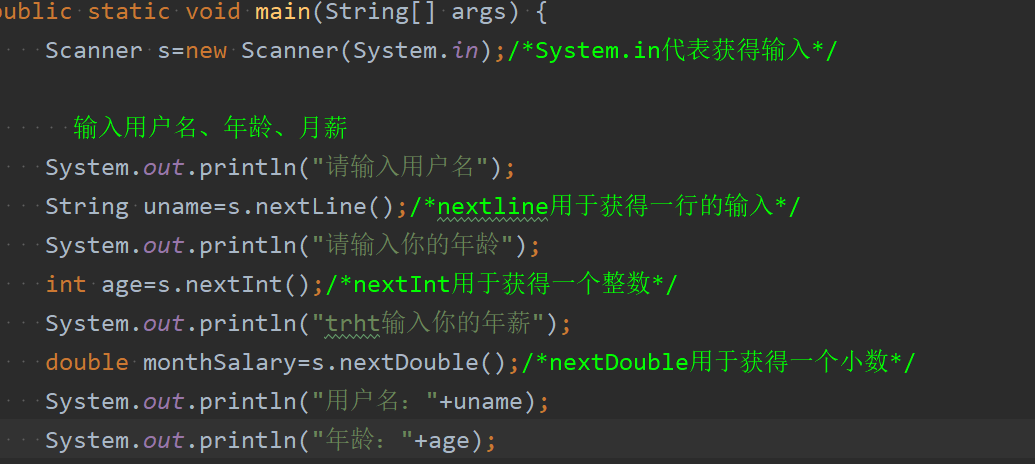
高级别值赋值给低级别值时，必须进行强制类型转换。

eg：i=26L;

1. 常量用Final修饰，放置在public后面

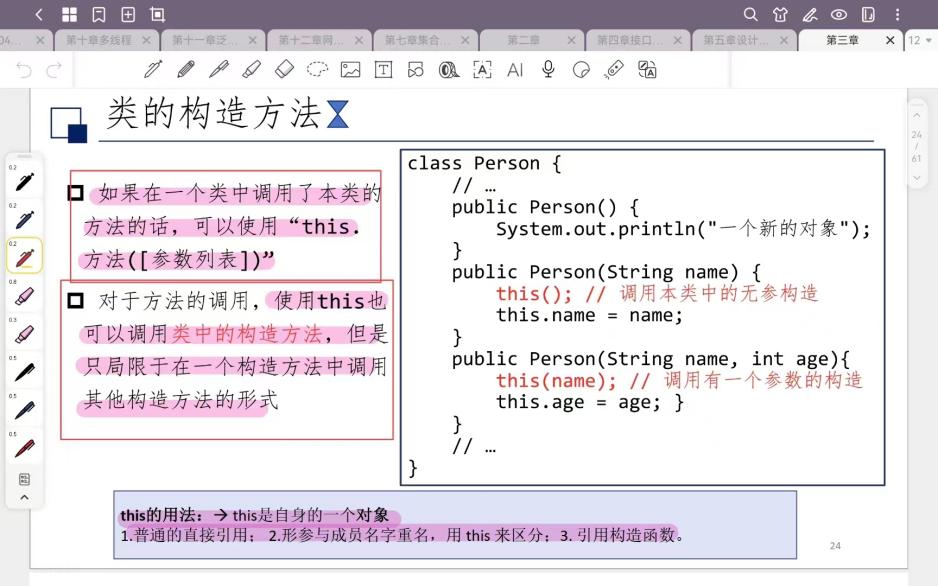
final关键字的作用：  
\* 1.修饰变量：被修饰的变量不可被重新赋值  
\* 2.修饰方法：不可被子类重写，但是可以重载  
\* 3.修饰类：修饰的类不能被继承，比如Math等\*/

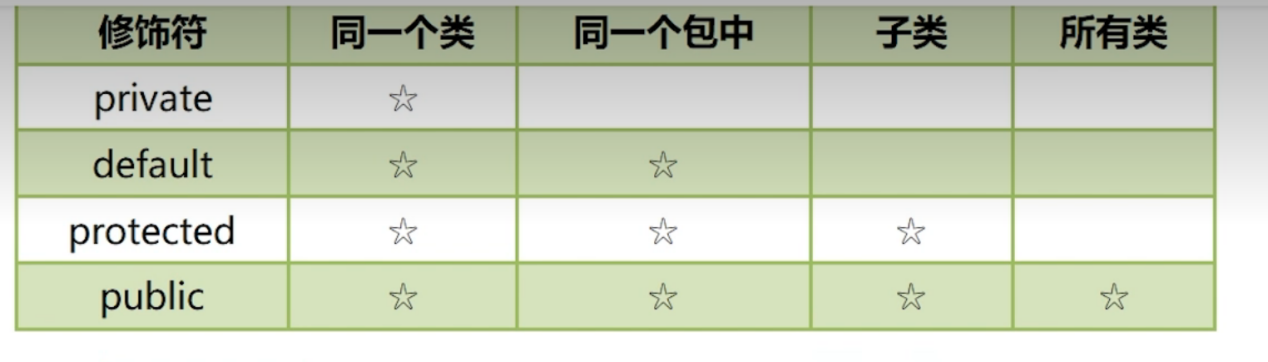
1. switch语句中表达式的值必须是整形、字符型、或者字符串，常量值同上。
2. java输入：实例化Scanner类进行使用，（）里写System.in。



1. java输出：用System.out如上图所示
   1. 注意：print输出后不换行
   2. println输出后换行
   3. wirte方法用于输出字节数组，在输出时不换行
2. java虚拟机和垃圾回收以及异常机制将在阐述第四章进行

第三章

1. 构造方法是一种特殊的方法，与类名同名，并且不写返回数据类型，用于对象中普通属性的初始化而不是建造对象。
2. this关键字是指向对象自身的关键字，例如this.name是指类中private String name语句定义的私有变量name。
3. 一个类可以有一个或多个构造方法。
4. 如果没有定义构造方法构造方法，系统会自动产生一个构造方法，称为默认构造方法
5. this也可以调用构造方法：但是只能用在构造方法当中，且必须放在构造方法的首行。
6. private：方法或者属性被其修饰时，被修饰的细节只能在同一个类之中使用  
    default：当什么都不写时，缺省值就为default。  
    在同一个类、同一个包里面时可以使用，即使是子类，但不在同一 个包里面也不能使用  
   \* protected：同一个类、同一个包中、不同包的子类可以使用  
   \* protected使用细节：1.若父类和子类在同一个包里，子类可以通过super 访问父类的protected成员，  
   \* 也可以访问父类对象的protected成员  
   \* 2.若父类和子类不在同一个包里，子类可以通过 super访问父类的protected成员，  
   \* 不能访问父类对象的protected成员  
   \* public：同一个类、同一个包中、不同包的子类、所有类都可以使用

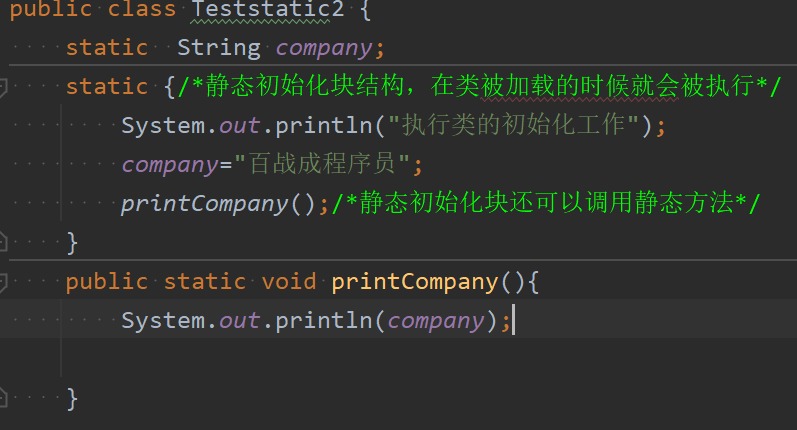


1. static属性和方法都是属于类的，普通的属性和方法和属性都被使用在对象（存储在堆中）里，static的方法和属性在方法区里，方法区存储类的一些信息和一些常量。
2. 注意：在static方法中不可直接访问非static的成员，而普通方法的方法里面可以访问static的属性和方法。

原因：普通方法使用都是属于实例化后的对象的，而静态方法属于类，在类载入时就被初始化

1. static的成员一般用“类名.类变量/方法”来调用
2. 静态初始化块用于初始化静态属性。

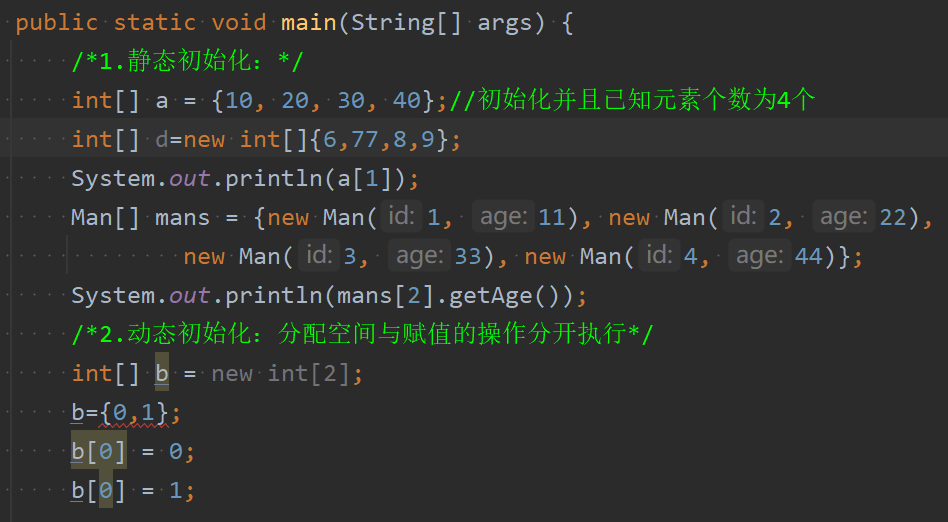
静态初始化块中不能直接访问非static成员



1. 一个类中可以含有多个静态初始块，
2. 不管构建多少个类的对象实例，static{}即静态块只会被执行一次，就算main方法里什么语句都不执行，也会打印：“执行类的初始化工作”。因为静态初始化块在类加载的时候就会被执行，在加载完之后才会开始执行main方法。

所以上图中就算创建了多少个Teststatic2的类，“执行类的初始化工作”这句话都会被执行且只会被执行一次

1. 数组一旦初始化，其长度不可变。
2. 每个数组都有一个属性length指明他的长度
3. java中的二维数组不必都是规则矩阵形式，每一行的元素个数可以不同
4. 注意：数组的动态初始化是将声明和赋值分开行，赋值时只能一个一个赋值，不能用大括号一起赋值，像下图一样。



1. 增强for循环和for循环的适用范围或者不同：
   1. 增强for循环：专门用于读取数组或集合中所有元素  
       /\*注意点：1.for-each在遍历数组过程中只能读取不能修改某元素的值  
       \* 2.for-each仅适用于遍历，不适合有关于下标索引的操作\*/
   2. 普通for循环适合在数组遍历时用到增删查改的情况。

第四章

1.子类继承父类，可以得到父亲的全部属性和方法（除了父亲的构造方法）  
\* 但不见得可以直接访问(比如：父亲的私有属性和方法)  
\* 如果定义一个类时没有调用extends，则他的父类是java.lang.Object

2.instanceof二元运算符：左边是对象，右边是类，当对象是右边类或者右边类的子类所创建的对象时  
\* 返回true；否则返回false

3.继承的明显特征：继承适用于is-a-king-of的关系，例如person和student

4.继承的优点：

（1）提高了代码的可复用性

（2）提高了程序的扩展性

（3）是雷雨类之间产生了联系，构成了多态的基础

4.子类不能直接继承父类的构造方法，需要用super关键字。

5.在一个类中，只能在第一行调用super()或者this();。

（1）若是构造方法的第一行没有调用super()或者this();  
\* 那么java会默认都会调用super()，含义是调用父类的无参数构造方 法

6.super可以看做对父类对象的引用，可通过super来访问父类中被子类覆盖的方法或属性

7.构造方法调用顺序：构造方法第一句总是：super(...)来调用父类对应的构造方法，  
所以流程是：先向上追溯到Object,然后再依次向下执行类的初始化块和创造方法，直到当前子类位置。 静态初始化快调用顺序，与构造方法调用顺序一样

8.抽象类和接口不能实例化

9.访问修饰符：只能用public或默认(default)

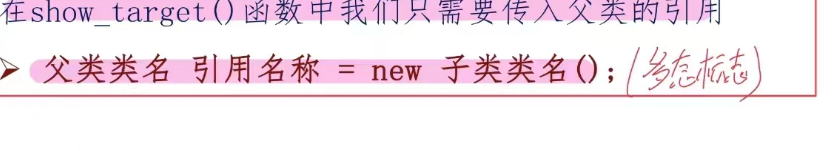
10.extends:接口可以多继承

11.接口中的属性只能是常量，  
\* 总是：public static final修饰。不写也可以，

12.接口中的方法只能是：public abstract,不写也可以

13.注意：接口不是特殊的抽象类，接口是接口，类是类，接口不是类

14.使用多态的三个必要条件：继承、重写、父类引用指向子类。

15.多态标志：

用该父类引用调用子类重写的方法，此时多态出现

16.方法的重写：方法名和形参列表相同，方法体内容进行重写  
\* 2.返回值类型和声明异常类型，子类小于等于父类  
\* 3.访问权限，子类大于等于父类



1. 方法的重载：一个类中方法的名称相同，但参数列表不同  
   \* 主要表现为：形参类型、形参个数、形参顺序不同。  
   \* 注意：只有返回值不同不构成方法的重载
   1. 构造器的重载：与普通方法同样，名字相同，参数不同  
      //主要表现为：形参类型、形参个数、形参顺序不同
2. 内部类：在类内部还可以定义另一个类，并且内部类拥有外部类的所有元素的访问权限
3. .java不允许多继承，实现多继承的两种方式：1.内部类2.接口
4. Object类的equas方法用于判断连个对象是否引用的是同一个对象，即较两个对象的地址，对为true，否则为false。大多数情况下会重写equals方法。
5. “==”对于基本数据类型表示值相等，对于引用数据类型则表示地址相等即是同一个对象。

23:（1）try用于监听、catch用于捕获异常，当try块后的代码遇到异常后，异常代码之后的语句不再执行。

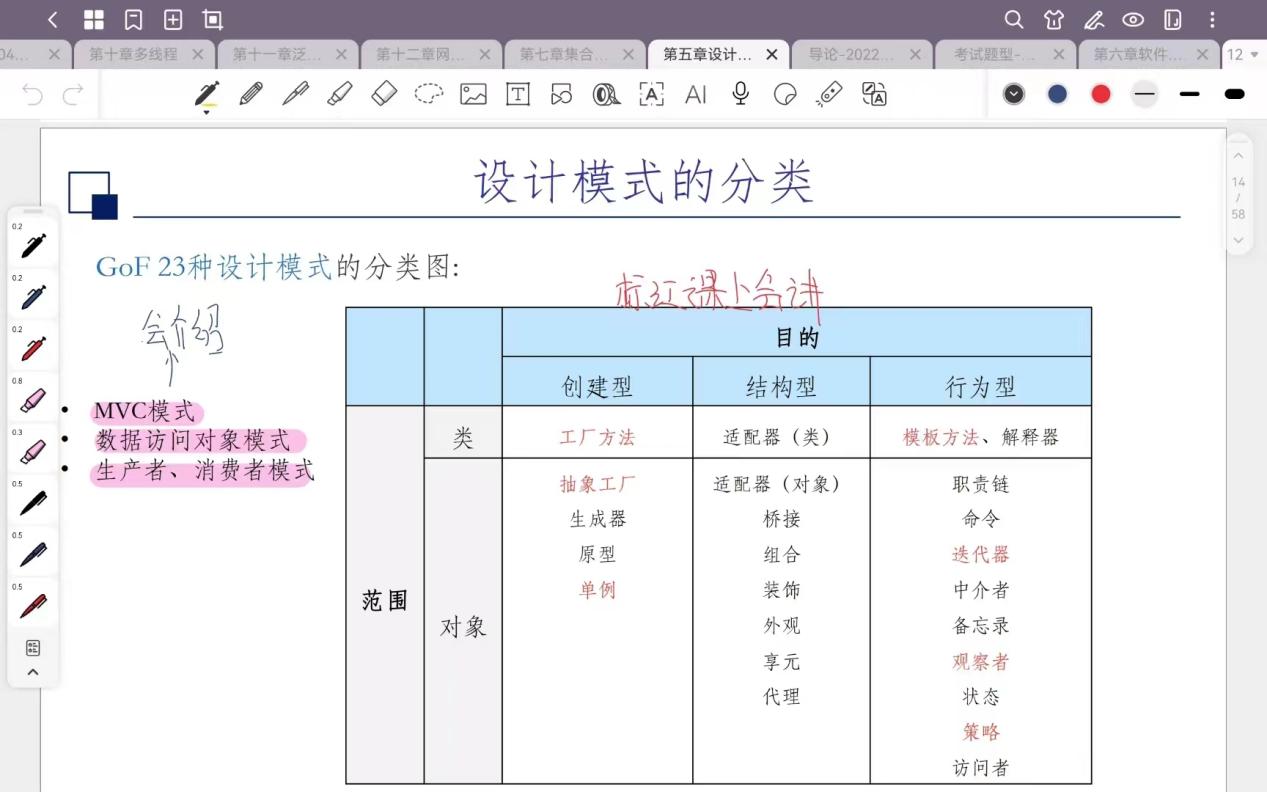
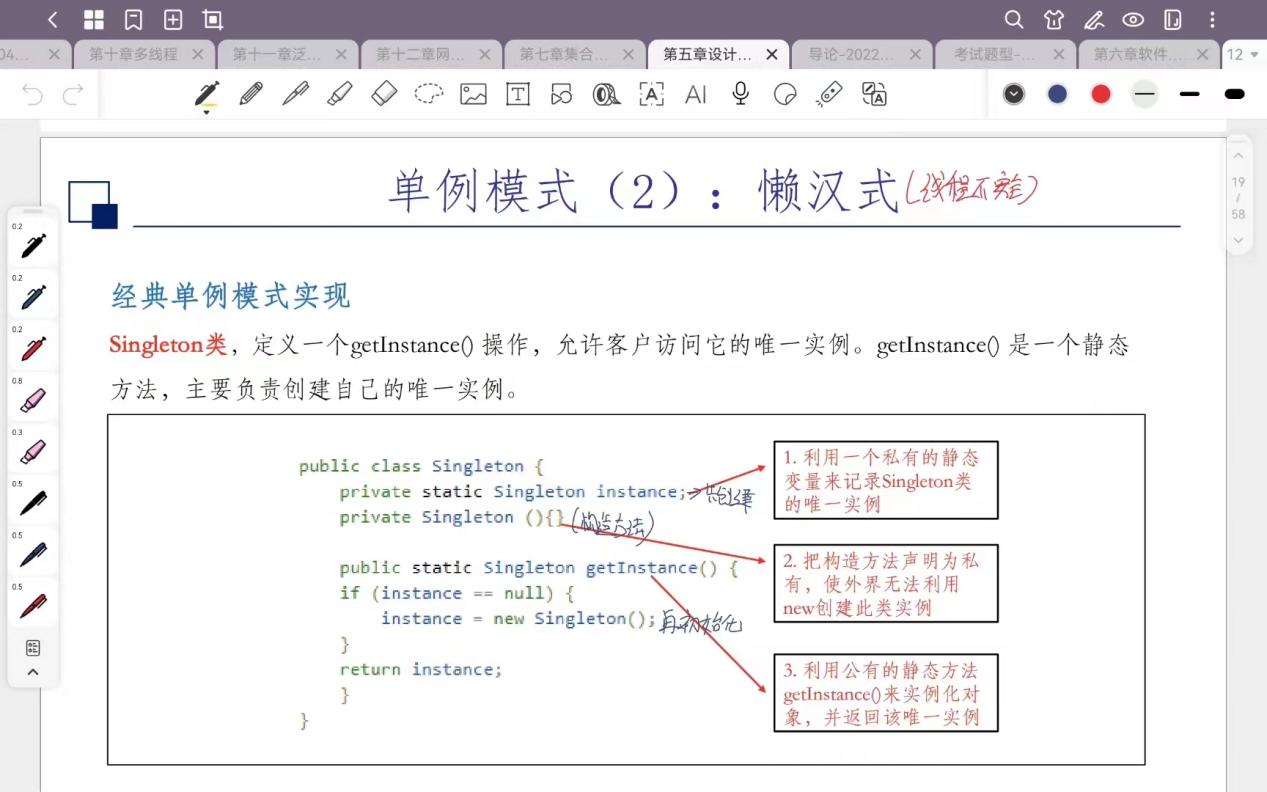
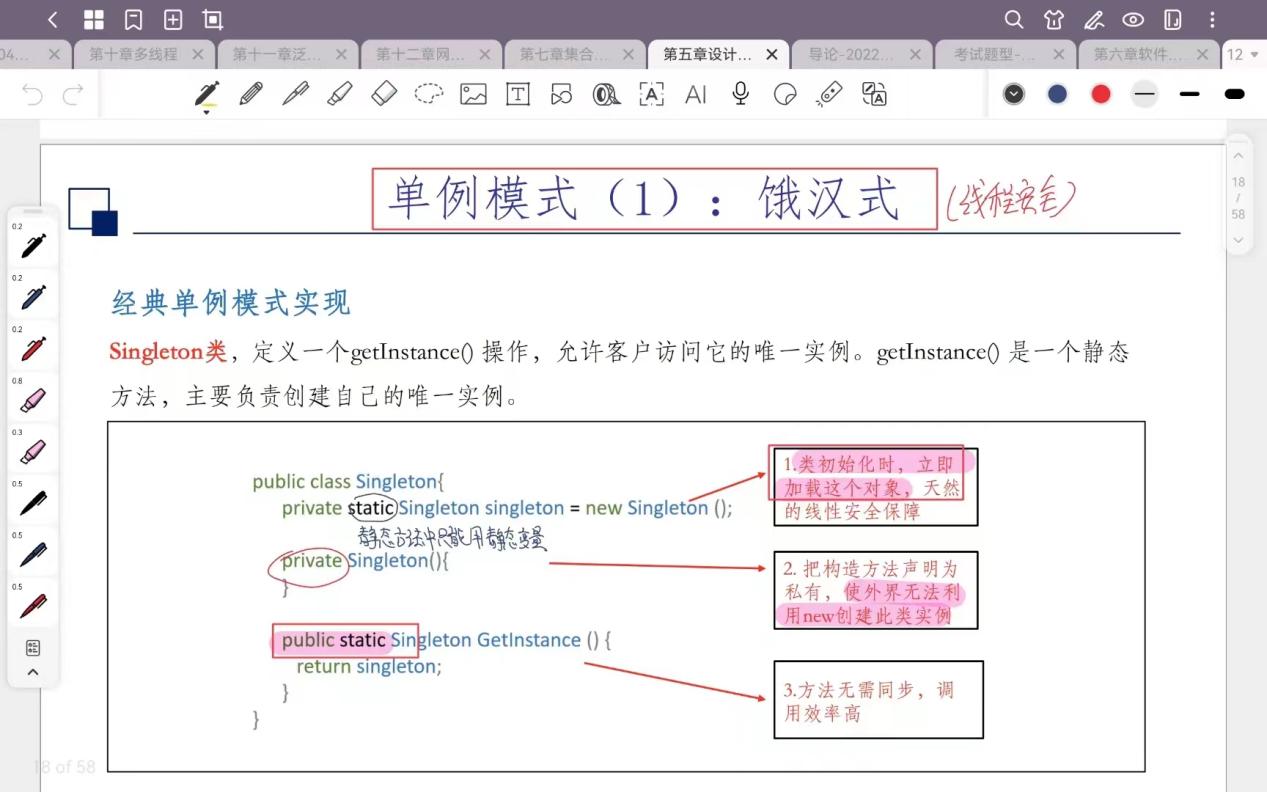
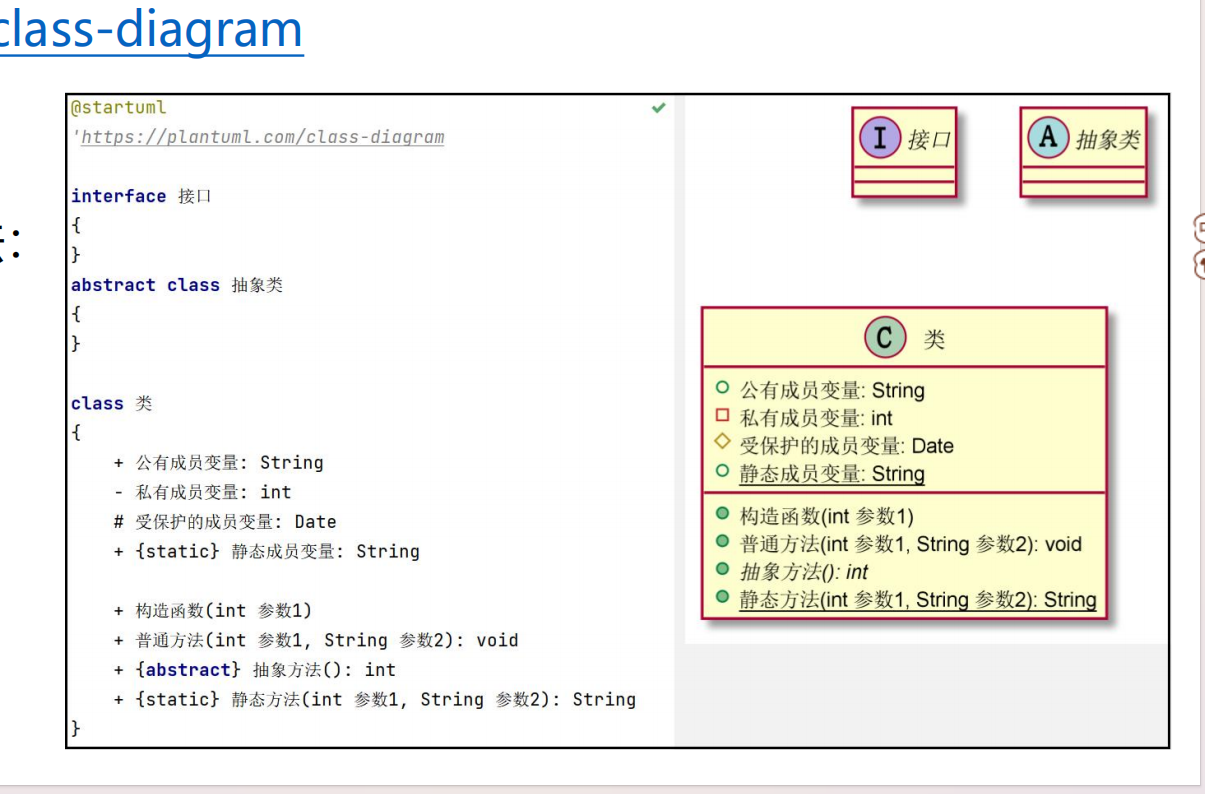
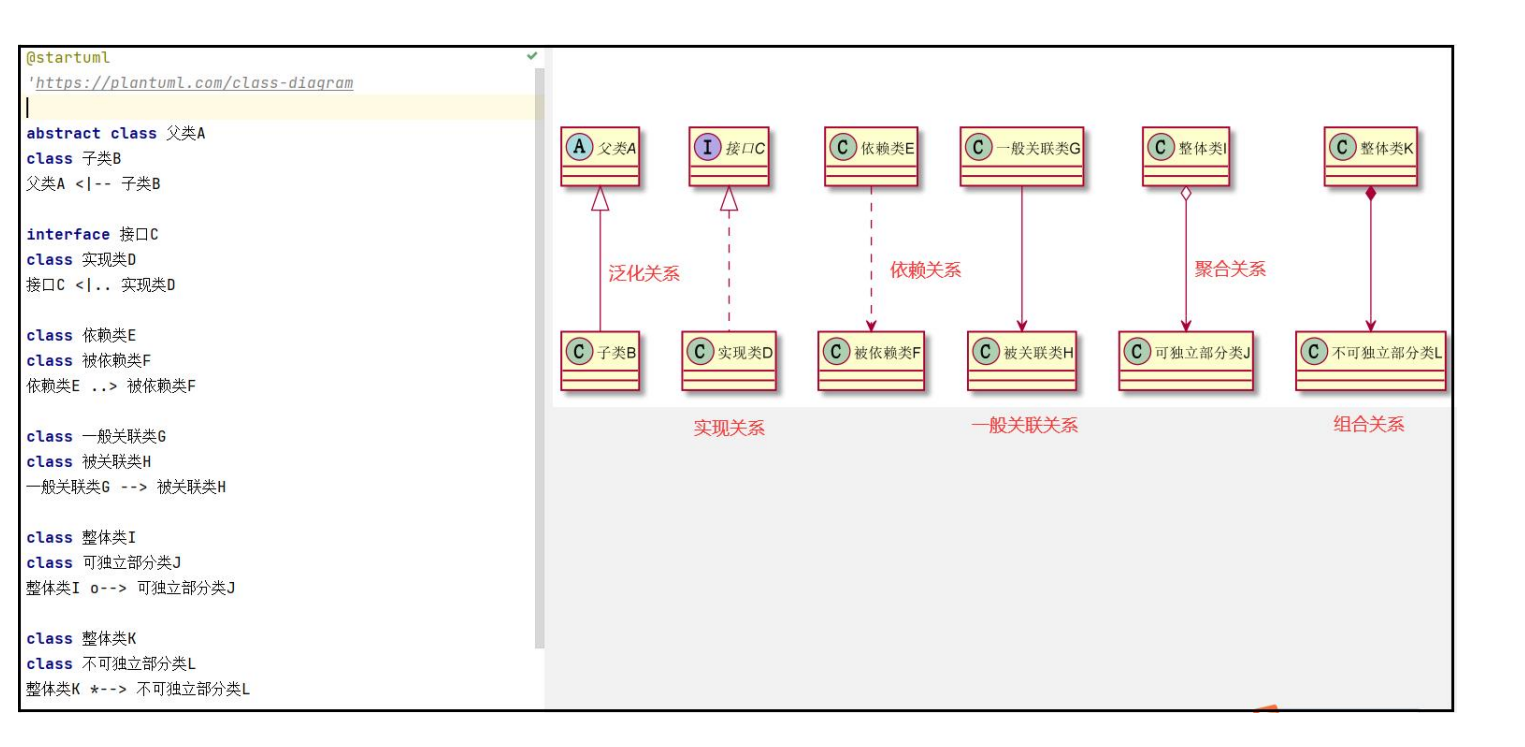
1. finally语句块总是被执行（有没有异常都会执行），用于回收try块中打开的物力资源。
2. throws用于方法头，throw用于方法内部
3. throws一次性可抛出多个异常，而throw只能一个
4. throws和throw都是消极处理异常的方式，只负责抛出异常
5. Throwable类有两个子类异常和错误。
   1. 异常分为运行时异常和检查性异常
   2. 运行时异常在编译时可以被忽略，检查性异常在编译时无法被忽略
6. 在编译时就报错的是检查性异常，处理方式：try-catch来捕获异常，或者用throws声明异常
7. java虚拟机：首先j程序被编译成字节码.class文件，然后通过虚拟机进行运行和处理
8. java虚拟机内存模型：
   1. jvm虚拟机的内存可以分为三个区域：栈、堆、方法区

栈是方法执行的内存模型，每个方法被调用都会创建一个栈帧来存放  
 方法里的局部变量、操作数、方法出口

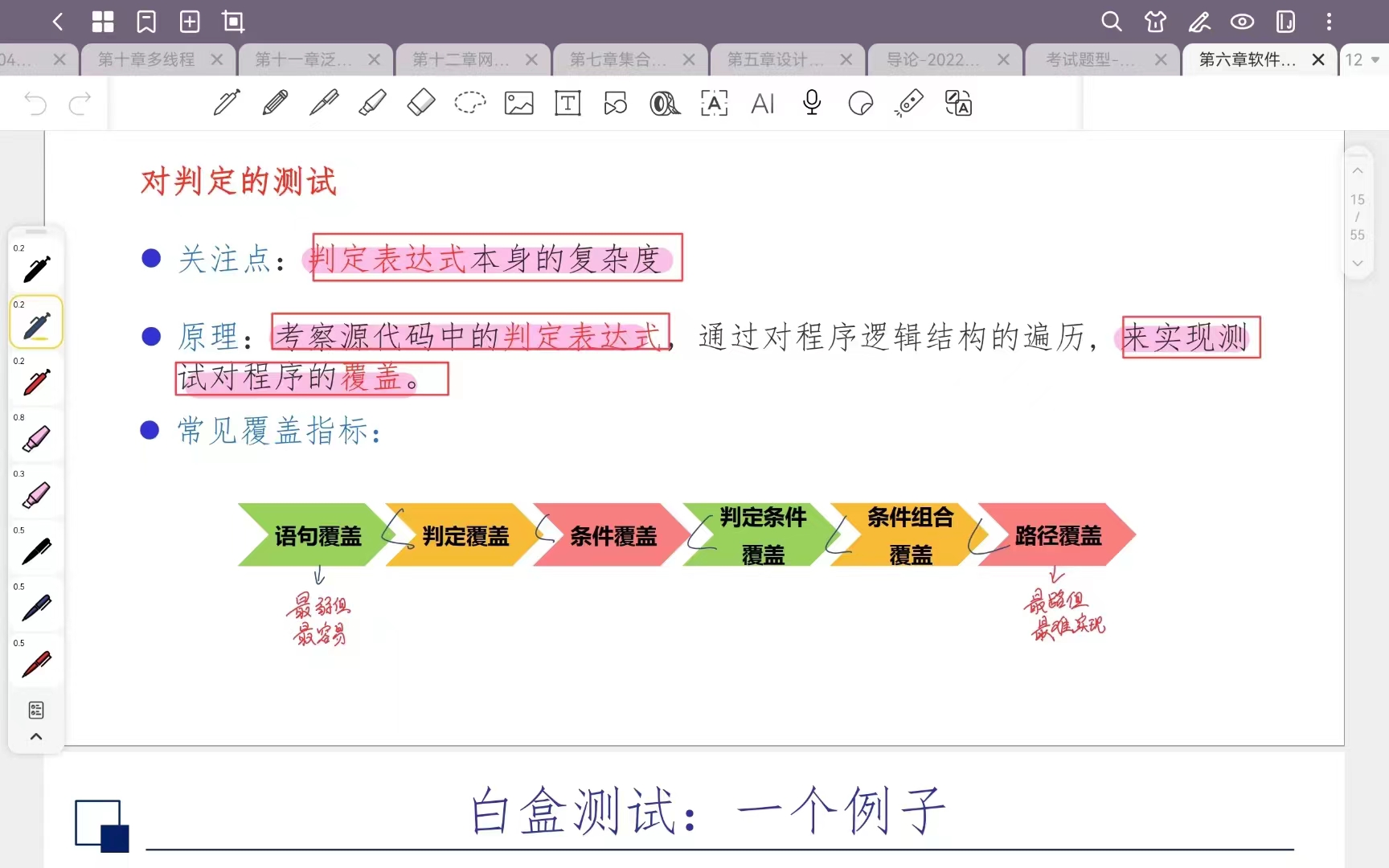
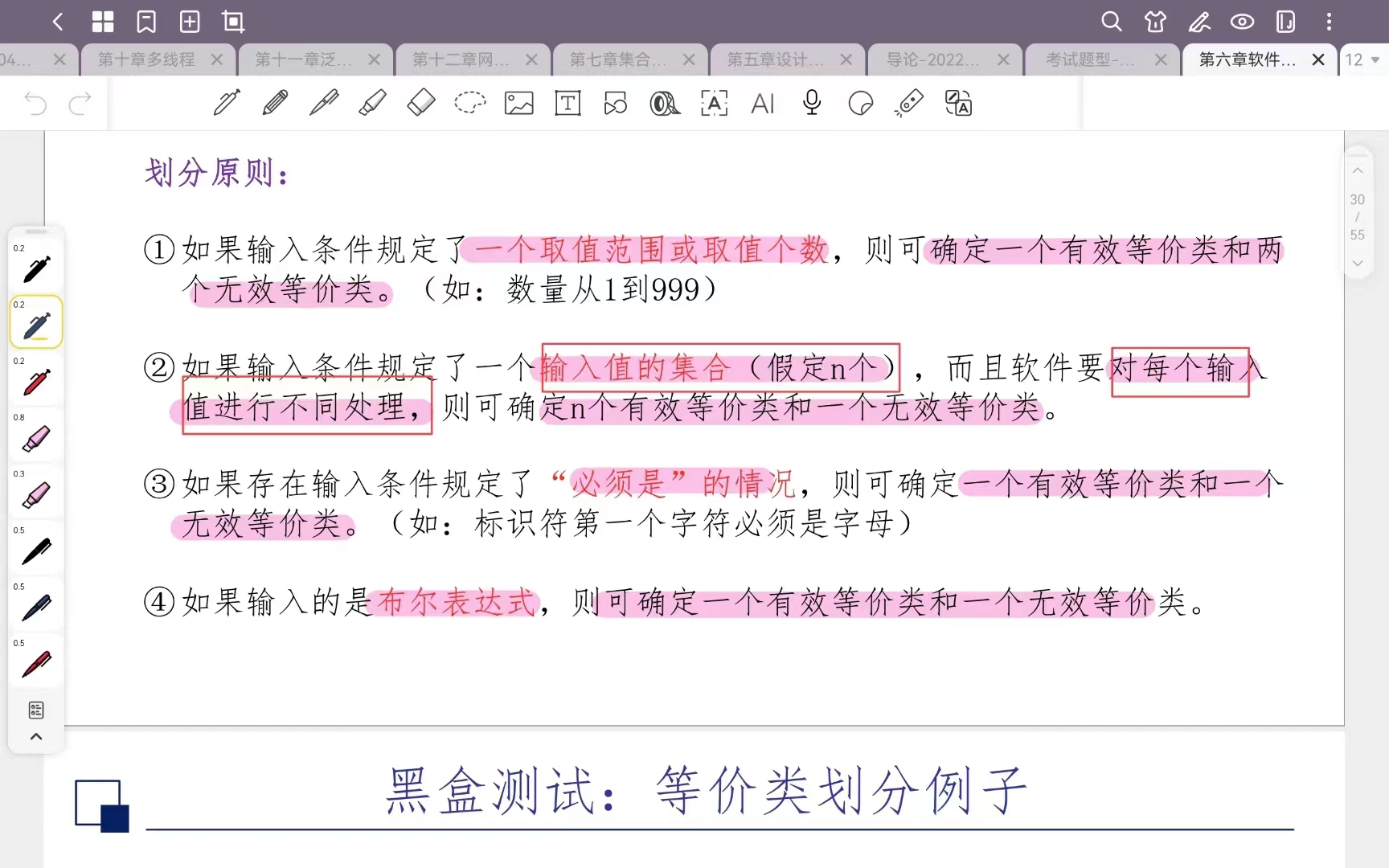
* 1. jvm为每个线程创建一个栈，用于存放该线程执行方法的信息。
  2. 堆用于存放创建好的对象和数组（数组也是对象）
  3. 方法区本质上也是堆，用于存放一些不变或唯一的内容  
      比如：类信息、字符串常量、静态变量(static)

1. java虚拟机的特点 ：
   1. 一次编译，到处运行
   2. 自动内存管理
   3. 自动垃圾回收功能
2. java采用隐式分配器进行垃圾回收。

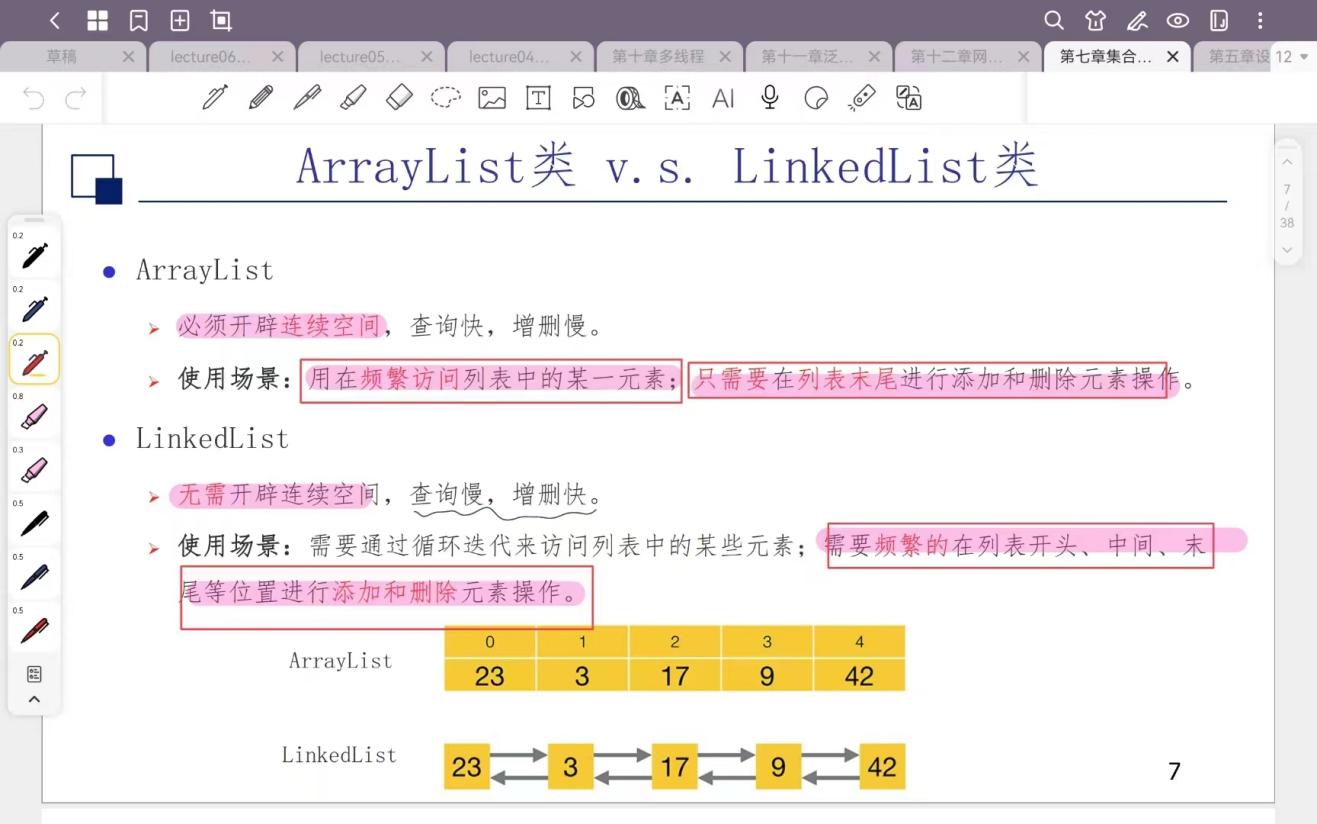
第五章

1. 单一职责原则：就一个类而言，应该仅有一个引起它变化的原因
2. OCP开闭原则：对扩展开放1，对修改关闭
3. 依赖倒转原则：高层模块不应该应该底层模块，都应该依赖抽象。针对接口编程而不是针对实现编程
4. 设计模式的本质是面向对象设计原则的实际运用
5. 重用设计：自动带来代码的重用
6. 注意：
   1. 设计模式根据目的来分（完成什么工作）
      1. 创建型模式：单例模式、工厂方法模式、抽象工厂模式
      2. 结构性模式
      3. 行为型模式：模板方法模式、观察者模式、策略模式
   2. 根据范围(应用于类或者是对象)可以分为
      1. 类模式
      2. 对象模式：
   3. 具体实例见图：
   4. 
   5. 注意：mvc模式、dao模式、生产者消费者模式不属于23个设计模式里
7. 单例模式：保证一个类仅有一个实例，并且提供一个访问它的全局访问点。
8. 单例模式要点：
   1. getInstance是静态方法，他是全局访问点，所以实例对象也得是静态变量才能放到static方法当中
   2. 构造器和变量全部私有，只有getInstance方法公有
   3. 饿汉式（类初始化时就加载这个对象）线程安全，懒汉式（只声明变量，并不用new关键字来创建对象）
   4. 
   5. 
   6. 多线程访问懒汉式单例模式，调用getInstance可能会创建多个实例，只需要加同步锁synchronized在方法名上（static和返回值中间），不允许两个线程同时进入
9. uml类图画法
   1. 
   2. +代表public，-代表private、#代表protected，{static}代表静态变量，{abstract}代表抽象方法
   3. 属性：后面跟类型，方法：后面跟返回值类型，并且括号里面需要写形参类型和名字
   4. 类与类之间的关系
   5. 
   6. 继承是实线空心箭头、实现是虚线空心箭头
10. 简单工厂模式不满足开闭原则
11. 组合：在新类中创建已有类的对象，通过该对象来调用已有类中的属性和方法。
12. 抽象工厂拓展一个产品等级（产品类型）很困难，不符合OCp该原则，拓展一个产品组族（工厂）很容易，符合开闭原则，和我你方便

第六章

1. 
2. 黑盒测试包括：
   1. 等价类划分
   2. 边界值分析
   3. 场景法
3. 等价类测试的定义：将无穷多的测试数据缩减到有限个等价区域内，通过测试等价类区域完成穷尽测试

第七章

1. list有序由下标元素可重复，set无序无下表元素不可重2.
2. hashset线程不安全，多线程访问需要加同步锁
3. map中key与value的类型可以相同也可以不同
4. 所有集合都适用于iterator
5. set可以使用for循环访问，但必须是增强for循环

第八章

1. 流具有方向性：以程序为参考
   1. 从程序流入设备——输出流
   2. 从设备流入程序——输入流
2. 文件是以字节的方式进行存储的
3. 处理流用于包装节电流
4. 字节：包括文本图片声音等
5. 字符：为纯文本
6. 在java中以Stream结尾为字节流、以er结尾为字符流
7. 流家族的根节点：Reader、Writer、InputStream、OutputStream
8. 流都需要close方法，结束后关闭
9. 序列化：将对象转换成字节序列的过程
10. 反序列化：将字节序列恢复成对象的过程
11. 对象序列化需要实现：实现Serializable接口，用wirteobject（）谢对象,反序列化用readobject（）读取对象
12. 被static修饰的字段不能被序列化

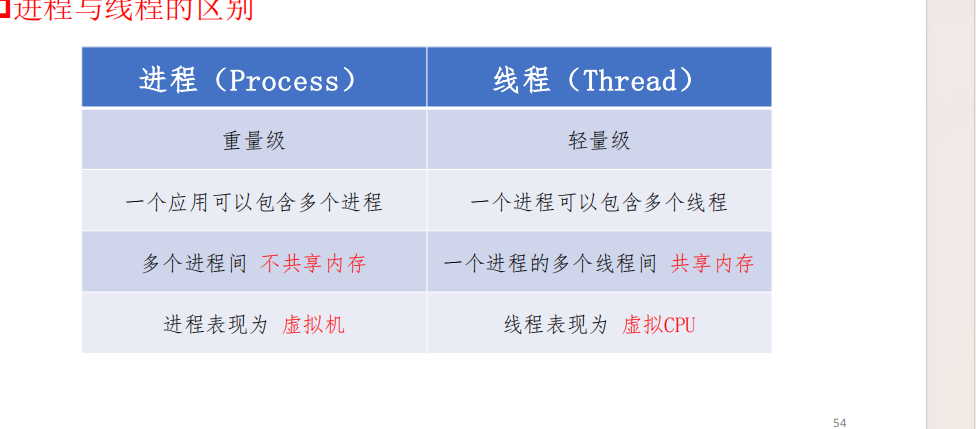
第九章

1. Swing的两种元素：容器和组件
2. Swing的组件继承与Jcomponet类
3. 顶层容器不能被别的容器所包含，
4. 轻量级容器既是容器，又是组件，必须包含在其他容器里
5. 事件处理机制三类对象：事件、事件源、事件监听器
6. MVC设计模式：m——model模型：对应操作内容：存储完整的内容

v——view视图：显示内容：一个模型可以有多个视图

c——controller控制器： 处理用户输入：使视图和模型分离开

第十章

1. 进程与线程的区别：
2. Runnable更常用，其优势在于：
   1. 任务与运行机制解耦，降低开销。
   2. 更容易实现多线程的资源共享
   3. 避免由于java'的单继承局限所带来的影响。

第十一章

1. 泛型提供了编译时类型安全检测极值，在编译时就可以识别出这种错误
2. 泛型只能使用引用数据类型
3. 泛型类中使用了泛型成员的方法不是泛型方法，只有声明泛型类型的才是静态方法
4. 如果泛型类和泛型方法的额·1名称一致，那么泛型方法会覆盖类的泛型
5. 静态方法无法访问类上定义的泛型
6. 上限通配符：extends，下线通配符：super
7. 获取class对象的三种方法：
   1. 对象.getclass（）方法
   2. Class.fforName()方法
   3. 类名.class方法
8. 反射构造类的实例方法：
   1. Class.newInstance（只会调用无参构造器，如果无无参构造器就异常）
   2. 首先调用getConstructor来或者构造器对象,使用Constructor的newInstance(可以携带参数)，
9. 