# 数据结构需要回顾的java知识及额外的积累

1. 异常：异常的捕获和抛出是独立的，且在一般在不同的方法上进行的；

抛出：一个方法如果可能抛出异常，则必须在方法头使用throws关键字声明，多个异常以逗号隔开。抛出异常的关键字是throw；

1. 如果抛出的异常是不是自己创建的异常（java包装好的异常），则只要符合某种条件，一般程序会自动抛出，往往不需要程序程序人员自己抛出这种异常。

2. 如果是自定义的异常，往往需要在某种情况下自己显示抛出异常。异常的抛出不一定要要捕获，即使捕获了异常也不一定要处理。

3. 一个方法只要抛出异常，就不会继续执行此方法的后续代码！

捕获：java异常的捕获机制使用try{ } catch ( ){ }结构实现。

1. 在一个方法内，捕获的是try{ } 语句体内所抛出的异常, 捕获是否成功则看catch( ){ }语句. 捕获异常的方法不用对异常进行声明。声明只是在抛出异常而又不进行捕获的情况下才需要的！

2. try结构的最后可以有finally{ } 块，只要程序不提前结束，则此语句块一定会被执行

3. 如果捕获多个异常，则要注意捕获异常的顺序。

创建自己的异常：创建自己的异常的类原则上应该只描述一种具体的异常，且异常类的名字应该符合异常规范，由类名提示异常的信息。其次，自定义的异常必须继承Exception 类。

2. 泛型数据

3. 迭代器

4. intellij IDEA 快捷方式

5. UML类图的绘制

(一)软件工程

1. 高质量代码

1）正确性：用户层面完全符合用户的需求，开发者层面，始终考虑另外几个提高代码质量的关键点，同时使用最为恰当的设计模式和准确算法。

2）可靠性：考虑故障的发生时机和合理的故障决解方案，最大化保证用户在使用软件的过程中由于故障造成不严重的后果。

特别是不受人控制的软件故障或者硬件故障，影响范围因尽量小。故障造成的后果可大可小，绝不可忽视。

3）健壮性：平稳可靠的决解所有具体问题，在任何情况下的可能都具有合理的处理方式。特别是对用户数据的互交，“决不要信任用户会正确输入”。作为开发人员，对于健壮性造成的其他结果不因考虑在内，而因把健壮性的设计作为一种开发习惯，自然而然！

4）可用性：一个软件产品，对于用户来说非但要可用，还必须用起来简单，即设计思路上应该符合大众化的使用要求。

5）有效性：有效的使用硬件所能分配给软件的资源，其中两种关键资源是cpu和主存，有效性主要跟软件设计原理和实现算法有关。

6）可维护性：维护主要围绕修复bug和增加新功能。其必须对软件进行合理修改。重要的软件系统应具有良好的结构和文档。

7）可重用性：组件或子系统的设计应实用于特定系统的同时又能独立于本系统，即，代表特定的功能，而不与整个系统过多耦合。

8）可移植性：其设计要求应轻松的在不同计算环境完美运行。即让软件与特定的操作系统和cpu无关。

----------------------------------------------------------------------------------------------------------------------------------------------------

（二）集合

集合：集合是一种收集和组织其他对象的对象。 组织方式：

1. 按照添加到集合中的次序

2. 某种固有的关系

抽象：这里的抽象是根据接口来讨论的，广义的接口概念指类，狭义的接口概念指public方法.

抽线在某些时候隐藏或忽略了某些细节。与一次处理过多细节相比，一次只处理一个抽象要更为容易些。抽象的最大特点便是站在用户的角度提供软件产品！

数据结构是实现集合的基本编程结构！

数据结构：

1. 集合的实现方式及定义集合类型

2. 元素的数据类型

3. 对元素的特定操作