<https://github.com/Snailclimb/JavaGuide#%E5%9F%BA%E7%A1%80>

Java 基础知识

1.2.7. Java 泛型了解么？什么是类型擦除？介绍一下常用的通配符？

类型擦除：**Java 的泛型是伪泛型，这是因为 Java 在编译期间，所有的泛型信息都会被擦掉，这也就是通常所说类型擦除**

List<Integer> a = new ArrayList<>();

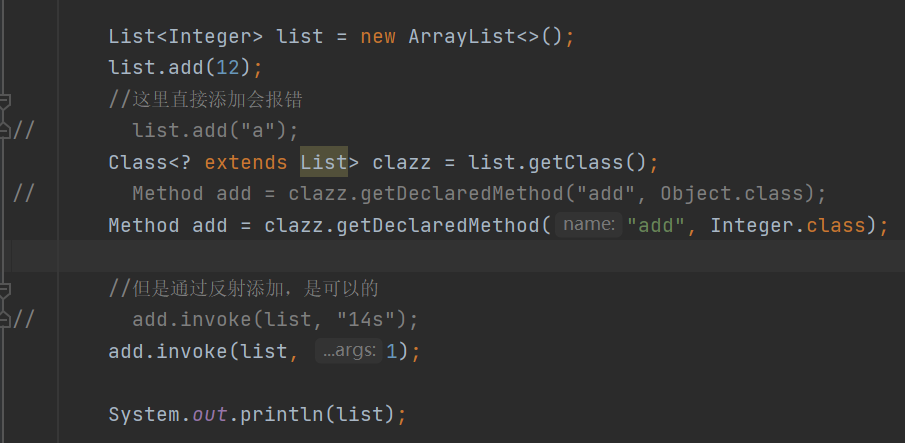
List<Character> b = new ArrayList<>();

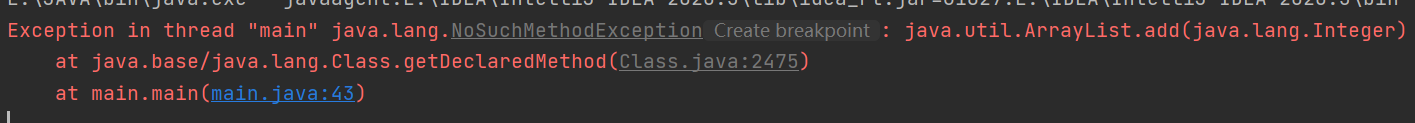
A与b其实都是一个类，泛型只是在编译的过程中限定了范围，可以通过反射的方式往a里添加一个不是integer类型的元素





但是不能用往list中直接添加一个integer类型的对象，因为list根本没有参数类型为integer类型的add方法





常用的通配符为： T，E，K，V，？

？ 表示不确定的 java 类型

T (type) 表示具体的一个 java 类型

K V (key value) 分别代表 java 键值中的 Key Value

E (element) 代表 Element

2.1.4. 成员变量与局部变量的区别有哪些？

**2.1.4. 成员变量与局部变量的区别有哪些？**

1. 从语法形式上看:成员变量是属于类的，而局部变量是在代码块或方法中定义的变量或是方法的参数；成员变量可以被 public,private,static 等修饰符所修饰，而局部变量不能被访问控制修饰符及 static 所修饰；但是，成员变量和局部变量都能被 final 所修饰。
2. 从变量在内存中的存储方式来看:如果成员变量是使用static修饰的，那么这个成员变量是属于类的，如果没有使用static修饰，这个成员变量是属于实例的。而对象存在于堆内存，局部变量则存在于栈内存。
3. 从变量在内存中的生存时间上看:成员变量是对象的一部分，它随着对象的创建而存在，而局部变量随着方法的调用而自动消失。
4. 成员变量如果没有被赋初值:则会自动以类型的默认值而赋值（一种情况例外:被 final 修饰的成员变量也必须显式地赋值），而局部变量则不会自动赋值。

2.2.3. 多态

当子类实例指向父类引用（A a2 = new B()）时，是子类决定了父类引用调用谁的成员方法。

1. 如果子类重写了父类的方法，父类引用会优先调用子类重写的方法

2. 如果子类没有重写父类的方法，父类引用会调用自己的方法

当子类实例指向子类引用(B b = new B())时，子类引用优先调用子类的方法，如果子类没有此方法，则会调用父类的方法

当超类对象引用变量引用子类对象时，被引用对象的类型而不是引用变量的类型决定了调用谁的成员方法，但是这个被调用的方法必须是在超类中定义过的，也就是说被子类覆盖的方法。 （但是如果强制把超类转换成子类的话，就可以调用子类中新添加而超类没有的方法了。）

this. method (obj) >super.method(obj)>this.method((super) obj)>super.method((super) obj)

2.4.2. Object 类的常见方法总结

**2.4.2. Object 类的常见方法总结**

Object 类是一个特殊的类，是所有类的父类。它主要提供了以下 11 个方法：

public final native Class<?> getClass()//native方法，用于返回当前运行时对象的Class对象，使用了final关键字修饰，故不允许子类重写。

public native int hashCode() //native方法，用于返回对象的哈希码，主要使用在哈希表中，比如JDK中的HashMap。

public boolean equals(Object obj)//用于比较2个对象的内存地址是否相等，String类对该方法进行了重写用户比较字符串的值是否相等。

protected native Object clone() throws CloneNotSupportedException//naitive方法，用于创建并返回当前对象的一份拷贝。一般情况下，对于任何对象 x，表达式 x.clone() != x 为true，x.clone().getClass() == x.getClass() 为true。Object本身没有实现Cloneable接口，所以不重写clone方法并且进行调用的话会发生CloneNotSupportedException异常。

public String toString()//返回类的名字@实例的哈希码的16进制的字符串。建议Object所有的子类都重写这个方法。

public final native void notify()//native方法，并且不能重写。唤醒一个在此对象监视器上等待的线程(监视器相当于就是锁的概念)。如果有多个线程在等待只会任意唤醒一个。

public final native void notifyAll()//native方法，并且不能重写。跟notify一样，唯一的区别就是会唤醒在此对象监视器上等待的所有线程，而不是一个线程。

public final native void wait(long timeout) throws InterruptedException//native方法，并且不能重写。暂停线程的执行。注意：sleep方法没有释放锁，而wait方法释放了锁 。timeout是等待时间。

public final void wait(long timeout, int nanos) throws InterruptedException//多了nanos参数，这个参数表示额外时间（以毫微秒为单位，范围是 0-999999）。 所以超时的时间还需要加上nanos毫秒。

public final void wait() throws InterruptedException//跟之前的2个wait方法一样，只不过该方法一直等待，没有超时时间这个概念

protected void finalize() throws Throwable { }//实例被垃圾回收器回收的时候触发的操作

3.2.4. 使用 try-with-resources 来代替try-catch-finally

3.4.1.2. BIO,NIO,AIO 有什么区别?

反射