一、接口描述1.1Coollect:一组Object的集合就是Collection，Java不能直接继承Collection只能继承它的子接口（list和set）1.2List接口是一个有效的Collection，使用此接口可以控制每个元素的插入位置，能通过索引来访问List中的元素，而且允许有相同的元素1.3Set与Collection有相同的接口，只是Set不保存重复的元素1.4Map将唯一的键映射到值。1.5Iteraror接口next()：返回迭代器刚越过的元素的引用，返回值是Object，需要强制转换成自己需要的类型hasNext()：判断容器内是否还有可供访问的元素

二、遍历Map2.1 for (String key : map.keySet())2.2for (String v : map.values())2.3for (Map.Entry<String, String> entry : map.entrySet())2.4Iterator<Map.Entry<String, String>> it = map.entrySet().iterator(); while (it.hasNext()) {Map.Entry<String, String> entry = it.next();}

三、泛型3.1泛型的本质是参数化类型，也就是说所操作的数据类型被指定为一个参数。3.2所有泛型方法声明都有一个类型参数声明部分（由尖括号分隔），该类型参数声明部分在方法返回类型之前。每一个类型参数声明部分包含一个或多个类型参数，参数间用逗号隔开。一个泛型参数，也被称为一个类型变量，是用于指定一个泛型类型名称的标识符。类型参数能被用来声明返回值类型，并且能作为泛型方法得到的实际参数类型的占位符。泛型方法体的声明和其他方法一样。注意类型参数只能代表引用型类型，不能是原始类型（像int,double,char的等）。3.3<? extends T>表示该通配符所代表的类型是T类型的子类。<? super T>表示该通配符所代表的类型是T类型的父类。3.4Integer a = 1;自动装箱。int b = a;自动拆箱。

四、异常4.1检查性异常：最具代表的检查性异常是用户错误或问题引起的异常，这是程序员无法预见的。运行时异常： 运行时异常是可能被程序员避免的异常。错误： 错误不是异常，而是脱离程序员控制的问题。错误在代码中通常被忽略。4.2捕获异常：try{//程序代码}catch（Exceptionname e1）{}。4.3throws： 在方法的声明上，声明这个方法会抛出的异常类型throw： 在方法的内部，抛出一个异常对象时使用finally： 无论是否发生异常，finally 代码块中的代码总会被执行。

五、递归5.1递归就是自己调用自己5.2 重复的操作，并且不知道什么时候结束，有很多层，并你不确定有几层。

六、可变参数：public void show(String... s){}//s被当做一个数组使用。

七、xml解析：7.1SAXReader saxReader = new SAXReader();使用xml的工具7.2Document document = saxReader.read(new File(path)); 读取文档7.3Element root = document.getRootElement(); /获取文档的根元素7.4root.attribute(name);List<Attribute> attrs = root.attributes();获得root的属性7.5Element child = root.element(eleName); List<Element> hildren = root.elements();获得root的子元素7.6Element child = root.element(eleName);List<Element> children = root.elements();获得root的子元素7.7OutputFormat outputFormat =OutputFormat.createPrettyPrint(); outputFormat.setEncoding("UTF-8"); 改xml格式7.8xmlWriter.write(document); 写入document7.9xmlWriter.close();将这个工具关闭，并且把工具中的内容，写进目标文件中7.10先读取一个xml文件，再在这个文件的基础上添加新的标签和属性，就不会出现覆盖问题。

八、线程8.1线程是程序的执行单元，正常情况下每个线程都有自己的执行流程互不干扰8.2thread = new HelloThread(new HelloRunnable());创建一个Thread的子类8.3thread = new Thread(new HelloRunnable()); 创建一个Runnable接口的实现类8.4thread = new Thread(new Runnable() {} 匿名内部类没有名字有写在另一个类内部。8.5start方法，才是真正开启一个线程thread.run()，只是调用了某个对象的方法8.6GC是java中的垃圾回收机制GC就是帮助我们把内存中不用的对象，销毁掉，释放内存空间，当一个main方法执行时实际是启动了两个线程一个是我们能看到的主线程一个是在后台运行，我们看不见的GCGC什么时候销毁对象，由它自己决定，我们无法决定8.7线程的抢占式执行多个线程共同去争抢CPU的执行权谁抢到了谁就执行8.8Thread.sleep(1000); 线程休眠1s.8.9int a = 2; 是原子性操作int b = a + 1; 不是原子性操作。8.10同步：如果你先开始了，那等你干完了我再开始 异步：你干你的，我干我的8.11线程是不安全的。

九、反射9.1通过反射可以直接拿到某一个类所有的信息包含成员变量，方法等9.2用来在运行时检查或者修改一个对象信息。