**1.反射的作用：**

普通的java对象是通过new关键字把对应类的字节码文件加载到内存，然后创建该对象的。

反射是通过一个名为Class的特殊类，用Class.forName("className");得到类的字节码对象，然后用newInstance()方法在虚拟机内部构造这个对象（针对无参构造函数）。

也就是说反射机制让我们可以先拿到java类对应的字节码对象，然后动态的进行任何可能的操作，包括：

在运行时判断任意一个对象所属的类

在运行时构造任意一个类的对象

在运行时判断任意一个类所具有的成员变量和方法

在运行时调用任意一个对象的方法

这些都是反射的功能。使用反射的主要作用是方便程序的扩展。

**反射的缺点：**

1、性能相比非反射会差

2、程序的安全性会降低

3、破坏了类的封装性，可以通过反射获取这个类的私有方法和属性

**2.抽象方法与接口的区别**

抽象类可以有构造方法，接口中不能有构造方法。

抽象类中可以有普通成员变量，接口中没有普通成员变量

抽象类中可以包含非抽象的普通方法，接口中的所有方法必须都是抽象的，不能有非抽象的普通方法。

抽象类中的抽象方法的访问类型可以是public，protected和（默认类型,虽然  eclipse下不报错，但应该也不行），但接口中的抽象方法只能是public类型的，并且默认即为public abstract类型。

抽象类中可以包含静态方法，接口中不能包含静态方法。

抽象类和接口中都可以包含静态成员变量，抽象类中的静态成员变量的访问类型可以任意，但接口中定义的变量只能是。

public static final类型，并且默认即为public static final类型。

一个类可以实现多个接口，但只能继承一个抽象类。

下面接着再说说两者在应用上的区别：

接口更多的是在系统架构设计方法发挥作用，主要用于定义模块之间的通信契约。而抽象类在代码实现方面发挥作用，可以实现代码的重用。

**3.super与this的区别与用法**

调用super()必须写在子类构造方法的第一行，否则编译不通过。每个子类构造方法的第一条语句，都是隐含地调用super()，如果父类没有这种形式的构造函数，那么在编译的时候就会报错。

super()和this()类似,区别是，super从子类中调用父类的构造方法，this()在同一类内调用其它方法。

super()和this()均需放在构造方法内第一行。

尽管可以用this调用一个构造器，但却不能调用两个。

this和super不能同时出现在一个构造函数里面，因为this必然会调用其它的构造函数，其它的构造函数必然也会有super语句的存在，所以在同一个构造函数里面有相同的语句，就失去了语句的意义，编译器也不会通过。

this()和super()都指的是对象，所以，均不可以在static环境中使用。包括：static变量,static方法，static语句块。

从本质上讲，this是一个指向本对象的指针, 然而super是一个Java关键字。

**4．算法结构**

算法包括0个或多个输入，1个或多个输出，中间有穷个处理过程。

存储结构不属于算法结构

**5．线程**

在每个线程中都是顺序执行的，所以sl.printAll();必须在前三句执行之后执行，也就是输出的内容必有ABC。

而线程之间是穿插执行的