**1： 从控制台输入先加入import java.util.Scanner;**

**整型：**

**Scanner con = new Scanner(Scanner.in);**

**int num = con.nextInt();**

**字符串：**

**String line = con.nextLine();**

**输出：  
System.out.print(); //不换行**

**System.out.println(); //换行**

**2： 字符串的定义：string s = “音节”;**

**3： 数组定义：int []a; //可以不定义长度**

**4: 数组的输出：import java.util.Arrays;**

**Int []a={1,2,3,4,5};**

**System.out.println(Arrays.toString(a));**

**结果为：[1,2,3,4,5]**

**5: 判断字符串与固定字符串是否相等**

**String line = con.nextLine();**

**If(line.equals(“1234”))**

**6： 数值的排序：import java.util.Arrays;**

**Int []a={8,6,7,1,2,3,4,5};**

**System.out.println(Arrays.toString(a));**

**结果为：[1,2,3,4,5,6,7,8]**

**7： 生成随机数：import java.util.Random;**

**Random a = new Random();**

**Int a = ran.nextInt(1000); //生成0到1000的一个随机数**

**8： 函数调用问题：**

**同一函数中：静态调用非静态函数需要通过对象，其他情况可直接调用；**

**不用类中；一个类函数调用另一个类中函数，若调用静态函数，直接用类名.函数名；**

**调用非静态函数，需通过另一个类建造对象才能调用**

**9： 包的封装：package.包名；**

**包的导入：import.包名.类名；**

**10：接口问题：在Java语言规范中，一个方法的特征仅包括方法的名字，参数的数目和种类，而不包括方法的返回类型，参数的名字以及所抛出来的异常。**

**接口的定义及实现：**

**定义eg:**

**public interface Predator {**

**boolean chasePrey(Prey p);**

**void eatPrey(Prey p);**

**}**

**实现eg：**

**public class Lion implements Predator {**

**public boolean chasePrey(Prey p) {**

**// programming to chase prey p (specifically for a lion)**

**}**

**public void eatPrey (Prey p) {**

**// programming to eat prey p (specifically for a lion)**

**}**

**}**

**注意：接口内的成员皆为静态（static）、final及公开（public）**

**11： 继承问题：需要用到关键字extends：**

**public class people{}**

**public class student extends people{}**

**12： 向上向下转型：**

**向上转型：父类的引用指向子类的对象：**

**people p = student ();**

**向下转型：指向子类的父类引用赋给子类：**

**people p = student();**

**student p2 = (student) p; //类型强制转换**

**13： 类的定义：  
 public class student{**

**public int number;**

**public string str;**

**public student(){} //无参构造器，初始化类**

**public student(int numValue, string strValue){ //含参..，可以直接在初始化时对对象进行赋值**

**number = numValue;**

**str = strValue;**

**}**

**}**

**14：构造器还有另一种方法赋值：**

**public student(int numValue,string strValue){**

**this number = numValue;**

**this str = strValue;**

**}**

**相对于上一种方法，这样不用担心参数名与变量名重复**

**15：**

**16：**