1：Java对象存储位置：1：寄存器最快的存储区；2：栈，速度仅次于寄存器，Java系统知道没个对象的确切生命周期；3：堆，存放所有的Java对象；4：常量存储：常量值直接存放在代码内部，永远不可变。

2：一个字节byte8为，一个short2字节，int4字节，char2字节，long4字节，float4字节，double8字节，boolean占以为

3：static方法或变量前不需要创建任何对象，直接调用；但对于非静态的则不行

4：对象赋值如a=b；那么他们指向同一个引用，修改a也会修改b；a.length = b.length不是同一个引用，修改a.length不会改变b.length

5:指数被作为双进度数处理，必须尾随d或是f

6：无符号移位操作 >>> ;如果对byte或short值进行移位运算需先将其转换成int类型

7：printBinaryInt（）和printBinaryLong分别是接受int和long的参数将其用二进制格式输出

8：小数转int总是截尾，如果想四舍五入就用Math.round（数）；

9：range（）方法：range（10）表示从0开始到9生成int数；range（5，10）表示5到9；range（5，20，3）表示5到20，步长为3；

10：return和break都是跳出循环，continue使代码返回循环的最开头（使i值递增）；

return跳出外层循环；break跳出内层循环，但如果带标签则跳出标签所值得循环；continue退回最内层循环开头，有标签则退回到标签紧接着的循环；

11：static方法内部不能调用非静态方法，反之可以

12：可变参数：想定义一个方法，但是不知道以后要用的时候想传几个参数进去，比如求和，你想定义一个传几个int数据都能给你算出和的方法，可以在方法的参数列表中这样写

public static int Sum(int... arr){

     int sum=0;

           for(int i=0;i<arr.length-1;i++){

               sum+=arr[i]

           }

      return sum;

}

使用方法：int sum = Sum（1，2，3，4）；

13：重构：如果你将一段代码放在某个位置，等过一短时间回头再去看你就会发现有更好的方式去实现相同的功能。

14：Java访问权限：public：；protected：；private：；如果没有任何权限修饰词则为包访问权限。继承的类可以访问protechted，不可以访问private。

15：super与this：this代表自身；super理解为是指向自己超（父）类对象的一个指针，而这个超类指的是离自己最近的一个父类。super.value可以指调用父类的方法或是变量。

16：基类在导出类可以访问他之前就已经初始化，即使没有为基类创建构造器，基类会自动合成一个默认的构造器。

想调用一个带参数的基类构造器就必须用关键字super显示地编写基类构造器的语句，并配适当的参数列表，调用基类构造器必须在你导出类构造器中要做到第一件事；

17：当需要亲自清理时最好编写自己的清理方法，但不要使用finalize（）；一旦涉及垃圾回收，垃圾回收器就不可靠了，不能依赖垃圾回收器去做任何事情。

18：向上转型：将子类应用转换为父类应用的动作；向上转型可能会丢失具体的类型信息。

19：带有恒定值的final static基本类型全用大写字母命名。且字与字之间用下划线隔开；

20：空白final是指被声明为final但又未给定初值的域，但必须在域的定义处或者每个构造器中用表达式对final赋值。

21：final参数，在参数列表中以声明的方式将参数指明为final，则只能在方法中读参而不能修改参数得值，主要用来在向匿名内部类传递数据。

22：final方法：把方法锁定，确保在继承中是方法的行为保持不变，且不会被覆盖。

23：final类是被禁止继承的，意味着所有方法也是fianl的。

24：public static void echo(List<Object> list) {

      System.out.println("List" + list);

  }

  public static void echo(Object o) {

      System.out.println("object" + o);

  }

  public static void main(String[] args) {

     List<String> l = new LinkedList<String>();

     for (int i = 1; i <= 5; i++) {

      l.add(i + "");

   }

    echo(l);

  } 结果为object[1, 2, 3, 4, 5]；echo（Object o）中object改为其他类型则会报错

25：接口是可以继承的；

26：.this：在外部类的类名加上.this表示调用的是外部类的方法；A.this.run();//调用外部类的run方法。

27：.new：

public class DotNew {

public class Inner{}

public static void main(String[] args){

DotNew dn =new DotNew();

DotNew.Inner dn2 =dn.new Inner();使用外部类的对象来创建该内部类对象

28：匿名内部类：某个类只用一次将其编写为单独的一个类比较麻烦，故引入匿名内部类：

abstract class Person {

    public abstract void eat();

}

 public class Demo {

    public static void main(String[] args) {

        Person p = new Person() {

            public void eat() {

                System.out.println("eat something");

            }

        };

        p.eat();

    }

}   只要一个类是抽象的或是一个接口，那么其子类中的方法都可以使用匿名内部类来实现。常用在多线程，因为要实现多线程必须继承Thread类或是继承Runnable接口。

29：静态嵌套类：

public class StaticTest {

   private static String name = "javaJohn";

　　private String id = "X001";

　　static class Person{

　　　　private String address = "swjtu,chenDu,China";

　　　　public String mail = "[josserchai@yahoo.com](mailto:josserchai@yahoo.com)";//内部类公有成员

　　　　public void display(){

　　　　　　//System.out.println(id);//不能直接访问外部类的非静态成员

　　　　　　System.out.println(name);//只能直接访问外部类的静态成员

　　　　　　System.out.println("Inner "+address);//访问本内部类成员。

　　　　}

　　}

    public void printInfo(){

　　　　Person person = new Person();

　　　　person.display();

　　　　//System.out.println(mail);//不可访问

　　　　//System.out.println(address);//不可访问

　　　　System.out.println(person.address);//可以访问内部类的私有成员

　　　　System.out.println(person.mail);//可以访问内部类的公有成员

   }

　　public static void main(String[] args) {

　　StaticTest staticTest = new StaticTest();

　　staticTest.printInfo();

}

}  在静态嵌套类内部，不能访问外部类的非静态成员

30：局部内部类：方法内部类；

31：Arrays.asList()：数组转容器，其输出被传递到list中，打乱引用不会修改·数组，但直接用其输出结果会修改数组的顺序。故如果你不想原来的数组被修改，那么你就需要在另一个容器中创建一个副本.