1.下面程序的运行结果是? 为什么？

```
String str1 = “hello”;
String str2 = “he”+new String(“llo”);
String str3 = “he”+”llo”;
System.err.println(str1== str2);
System.err.println(str1 == str3);
答案： false true
```

2.HashSet 里的元素是不能重复的, 那用什么方法来区分重复与否呢?

答案：当往集合在添加元素时，调用 add（Object）方法时候，首先会调用 Object 的 hashCode()方法判hashCode 是否已经存在，如不存在则直接插入元素；

如果已存在则调用 Object 对象的 equals()方法判断是否返回 true，如果为 true 则说明元素已经 存在，如为 false 则插入元素。

3.List ,Set, Map 是否继承来自 Collection 接口? 存取元素时, 有何差异?

答案： List,Set 是继承 Collection 接口； Map 不是。
List：元素有放入顺序，元素可重复 ，通过下标来存取 和值来存取
Map：元素按键值对存取，无放入顺序
Set：元素无存取顺序，元素不可重复（注意：元素虽然无放入顺序，但是元素在 set 中的位置是有该元素的 HashCode 决定的，其位置其实是固定的）

4.简述 Java 中的值传递和引用传递?

按值传递是指的是在方法调用时，传递的参数是按值的拷贝传递。
按值传递重要特点：传递的是值的拷贝，也就是说传递后就互不相关了

```
public class TempTest {
    private void test1(int a){
        a = 5;
        System.out.println("test1 方法中的 a="+a);
    }
    public static void main(String[] args) {
        TempTest t = new TempTest();
        int a = 3;
        t.test1(a);//传递后， test1 方法对变量值的改变不影响这里的 a
        System.out.println("main 方法中的 a="+a);
    }
}
```

运行结果是：
test1 方法中的 a=5
main 方法中的 a=3

按引用传递是指的是在方法调用时，传递的参数是按引用进行传递，其实传递的引用的地址，也就是变量所 对应的内存空间的地址。传递的是值的引用，也就是说传递前和传递后都指向同一个引用（也就是同一个内存空间）

示例如下：

```
public class TempTest {
    private void test1(A a){
        a.age = 20;
        System.out.println("test1 方法中的 age="+a.age);
    }
    public static void main(String[] args) {
        TempTest t = new TempTest();
        A a = new A();
        a.age = 10;
        t.test1(a);
        System.out.println(" main 方法中的 age=" +a.age);
    }
}
class A{
    public int age = 0;
}
```

运行结果如下：
[java] view plain copy
test1 方法中的 age=20
main 方法中的 age=20

5.switch 是否作用在 byte 上, 是否能作用在 long 上, 是否能作用在 String 上？

答案： switch 可作用于 char byte short int；
switch 可作用于 char byte short int 对应的包装类；
switch 不可作用于 long double float boolean，包括他们的包装类；

6.Java 语言如何进行异常处理? 请写出几个常见的运行时异常的编译时异常

答案： java 语言进行异常处理的方式有：

throws: throws 是方法可能抛出异常的声明。 (用在声明方法时，表示该方法可能要抛出异常)
throw : throw 是语句抛出一个异常。
常见的运行时异常的编译时异常:
NullPointerException - 空指针引用异常
ClassCastException - 类型强制转换异常。
IllegalArgumentException - 传递非法参数异常。
ClassNotFoundException - 类找不到异常
ArrayStoreException - 向数组中存放与声明类型不兼容对象异常
IndexOutOfBoundsException - 下标越界异常
NegativeArraySizeException - 创建一个大小为负数的数组错误异常
NumberFormatException - 数字格式异常
SecurityException - 安全异常
UnsupportedOperationException - 不支持的操作异常

7.简述数据库事务和实际工作中的作用

所谓事务是用户定义的一个数据库操作序列,这些操作要么全做要么全不做,是一个不可分割的工作单位。例如,在关系数据库中,一个事务可以是一条 SQL 语句、一组 SQL 语句或整个程序。

简单举个例子就是你要同时修改数据库中两个不同表的时候，如果它们不是一个事务的话，当第一个表修改完，可是第二表改修出现了异常而没能修改的情况下，就只有第二个表回到未修改之前的状态，而第一个表已经被修改完毕。而当你把它们设定为一个事务的时候，当第一个表修改完，可是第二表改修出现了异常而没能修改的情况下，第一个表和第二个表都要回到未修改的状态！这就是所谓的事务回滚。

例如，在将资金从一个帐户转移到另一个帐户的银行应用中，一个帐户将一定的金额贷记到一个数据库表中，同时另一个帐户将相同的金额借记到另一个数据库表中。由于计算机可能会因停电、网络中断等而出现故障，因此有可能更新了一个表中的行，但没有更新另一个表中的行。如果数据库支持事务，则可以将数据库操作组成一个事务，以防止因这些事件而使数据库出现不一致。如果事务中的某个点发生故障，则所有更新都可以回滚到事务开始之前的状态。如果没有发生故障，则通过以完成状态提交事务来完成更新。