JavaSE -05 测试题

注：请在文本文件中写出试题的题目及答案

一、问答题

1. 写出 5 个 File 类的常用方法，并说明作用

   答：查看 api。

2. 谈谈 in、out

   答：在 java 中 in、out 是站在内存的角度来说的，从硬盘读到内存中是 in，从内存写到硬盘是 out。

3. 实现多线程的方式有哪些？分别如何启动一个子线程？

   答：在 java 中实现线程有两种方式。第一种继承 java.lang.Thread 类，并重写其中的 run 方法，调用 start()方法启动线程；第二种是实现 java.lang.Runnable 接口，并实现接口中定义的 run 方法，这种方式定义的子线程需要借助 Thread 类的 start()方法来启动。

4. 如何引起线程安全问题，如何解决？

   答：多线程的情况下，多个线程访问同一个对象的实例变量会引起线程安全问题。第一种在方法的声明处添加 synchronized 关键字；第二种使用 synchronized 同步代码块；第三种使用 ReentrantLock 同步锁。

5. 写出线程的生命周期

   答：新建—>就绪—>运行—>死亡

   新建—>就绪—>运行—>（阻塞—>就绪—>运行）—>死亡

6. Collection 和 Collections 的区别

   答：Collection 是 List、Set 的父接口。Collections 是 java 提供的工具类，包含

了操作集合的一系列工具方法，例如提供了实现线程安全的集合的方法和二分法搜索集合的方法等。

7. 谈谈 java 序列化

答 :将 java 对象写入流的过程称为序列化，将对象从流中读取出来称为反序列化。参加序列化的类必须要实现 java.io.Serializeble 接口，并生成序列化 ID。静态属性和使用 transient 关键字修饰的属性不参与序列化。如果父类是可序列化的，那么子类必然是可序列化的。类序列化时，属性依赖的对象必须是可序列化的对象。

8. 什么是守护线程，怎么实现

答：守护线程是指随着主线程死亡而死亡的子线程。在子线程启动之前，调用 setDeamon(true)方法。

9. 什么是 join 线程，怎么实现

答 : join 线程可以让其他线程进入阻塞状态，直到自己执行完。在线程启动后调用 join()方法即可。

10. 创建线程的安全的 ArrayList 集合

答 :List<String> list = Collections.synchronizedList(new ArrayList<String>());

11. 请说明三大范式

答：1） 确保每列的原子性

2） 在第一范式的基础上，确保每列都和主键相关

3） 在第二范式的基础上，确保每列都和主键直接相关，而不是间接相关

12. 拆表带来的优势和劣势

答：优势：降低数据冗余度，方便修改

劣势：增加查询难度

13. 解释 Select * from t_user 中*的作用和缺点

答：如果查询时要显示所有列，可以使用通配符*，但是要尽量避免使用，因为执行的时候要先解析*获得该表的所有列在执行查询，会降低查询的性能，因此采用写出所有列的形式代替通配符*。

14. Having 和 where 有什么区别

答：where 用于分组前过滤，having 用于分组后的过滤。

15. count(*)和 count（name）的区别

答：count(*)统计表的行数，count（列）会忽略该列中所有的 null 值和空值再进行统计。

二、编程题

1. 请使用流的形式将 D 盘根目录中名字为 1.jpg 的图片拷贝到 E 盘根目录中，并命名为 2.jpg。

```
try {
    BufferedInputStream bufferInput = new BufferedInputStream(new
        FileInputStream("d:/1.jpg"));
    BufferedOutputStream bufferOutput = new BufferedOutputStream(
        new FileOutputStream("e:/2.jpg"));

    int len = -1;
    while((len = bufferInput.read()) != -1) {
        bufferOutput.write(len);
    }

    bufferOutput.flush();
    bufferOutput.close();
    bufferInput.close();
} catch (FileNotFoundException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
```

2. 第一个人的年龄为 10，第二个人的年龄比第一个人大两岁，以此类推，求第 8 个人的年龄。（递归算法）

```
public int age(int n) {
        if(n == 1) {
            return 10;
        }
        return age(n-1) + 2;
}
```

调用 age(8);

3. 有表如下图：

t_score:

id	stuid	subject	score
1	1001	数学	80
2	1001	语文	53
3	1001	英语	59
4	1002	数学	55
5	1002	语文	56
6	1002	英语	50
7	1003	数学	100
8	1003	语文	90
9	1003	英语	88

t_stu：

id	name
1001	张三
1002	李四
1003	王五

- 不及格科目大于或等于 2 科的学生的姓名

```
SELECT count(*),stu.id ,

(select name from t_stu where id = stu.id)
```

```
FROM  t_score` as s, t_stu as stu

where  s.score <= 60  and s.stuid = stu.id

GROUP BY stu.id    having count(*) >=2;
```

- 找出总成绩最高的学生姓名

```
SELECT sum(score) as total, stu.name

FROM `t_score` as s, t_stu stu

where s.stuid = stu.id

GROUP BY stu.name order by total desc limit 0,1;
```