杂

**Q1. C#和JAVA的异同？**

A：

相同：

1. 他们都是面向对象的语言,都具有封装、继承、多态的特性。
2. 语言风格类似
3. 都有接口的概念，且两者都只能继承一个类，但是可以继承多个接口。

差异：

1. Java和C#都有自动垃圾回收机制，运行时环境可以自动清理不再使用的内存，C#中分为托管资源和非托管资源，托管资源由垃圾回收器主动释放，非托管资源（文件、窗口等）由程序员主动释放。而JAVA中都是托管资源，由垃圾回收器主动回收。
2. JAVA里是package，C#里是namespace
3. 在跨平台上，C#使用的是.NET Framework，而JAVA使用的是JVM，
4. 应用上，java的应用比C#广泛，C#用于开发一些桌面和网络，游戏开发等，JAVA可以应用于企业级应用、网站建设、移动应用、大数据等。

**Q2. C++和JAVA的对比？**

**A:**

1. Java 是纯粹的面向对象语言，所有的对象都继承自 java.lang.Object，C++ 为了兼容 C 即支持面向对象也支持面向过程。
2. Java 通过虚拟机从而实现跨平台特性，但是 C++ 依赖于特定的平台。
3. Java 没有指针，它的引用可以理解为安全指针，而 C++ 具有和 C 一样的指针。
4. Java 支持自动垃圾回收，而 C++ 需要手动回收。
5. Java 不支持多重继承，只能通过实现多个接口来达到相同目的，而 C++ 支持多重继承。Java使用接口进行多重继承，类似于C++种的虚函数
6. Java 不支持操作符重载，虽然可以对两个 String 对象执行加法运算，但是这是语言内置支持的操作，不属于操作符重载，而 C++ 可以。
7. Java 的 goto 是保留字，但是不可用，C++ 可以使用 goto。

**Q3. 简单介绍下地面站软件项目**

A:

**Q4. 地面站软件开发中遇到的问题？解决方法？**

A: 总的来说，因为我们是基于一个开源MP地面站软件来开发的，开源MP地面站功能非常多且复杂，但因为我们的地面站需求是简化和优化，所以对MP地面站某项功能实现的结构和原理进行梳理就非常的重要，需要从源码一点一点调试，来梳理清楚其功能实现的过程，需要的控件、程序集等。然后根据我们自己的功能需求进行改写完善，实现自己的功能。

一个小点：起降点航线功能实现中，遇到的一个问题是通过一个其实点，根据一个角度和距离，来确定另一个点的位置，在分析时利用笛卡尔坐标系可以很容易的计算出点的位置，但是具体实现时却发现位置总是有偏差。然后发现因为在最终生成坐标点时其实确定经纬度坐标出现了问题，在不同的经度下，其刻度在不同纬度时所对应的距离是变化的，所以并不是知道其度数和距离的比例就可以直接算。因此需要根据坐标的改变，而实时改变其刻度所对应距离的比例。

**Q5. 定点实时测距？起降点航线添加？**

A:测距：测距功能在MP地面站中是通过添加两个点，然后菜单选择距离功能得到的，操作较为麻烦，我们的需求是，添加起始点后，随着鼠标指针的移动，实时显示起始点到鼠标指针位置处的距离。设置一个CheckBox控件，勾选后触发checkBox的CheckedChanged事件，修改测距的bool标志为true，进入测距状态，随后在地图上点击一个起始点，地图会添加起始点标记，之后鼠标移动时触发地图控件的Paint重绘事件，判断确定为测距标志位true后，清除原有的连线，重新绘制当前位置下的连线，并显示距离。

起降点航线：顺应固定翼飞机飞行轨迹，寻找合适的起飞点，使得固定翼能直接进入盘旋升高点的盘旋半径。起飞时的直线会和盘旋的的圆，通过这个相切关系，利用角度和距离比来确定两个起飞点的具体位置，

**Q6. 自我介绍**

A:

JAVA

**Q1. 自动装箱和自动拆箱**

A: 装箱就是自动将基本数据类型转换为包装器类型；拆箱就是自动将包装器类型转换为基本数据类型。包装类型是基本数据类型的封装类，在泛型中，会定义一个或多个类型参数，是一个object，因此需要提高基本类型的包装类，来与泛型间接的工作。装箱调用Integer.valueOf(2)，拆箱调用了X.intValue()。包装类不能被继承，string类也是。

**Q2. 各基本数据类型大小？**

A: byte/8（一个字节）; char/16; short/16; int/32; float/32; long/64; double/64; boolean/~ bool只有两个值，可以用1bit来存储。JVM 支持 boolean 数组，但是是通过读写 byte 数组来实现的。

**Q3. String不可变的好处？**

A:1可以缓存hash值，String 的 hash 值经常被使用，例如 String 用做 HashMap 的 key。不可变的特性可以使得 **hash 值也不可变**，因此只需要进行一次计算。

2.string pool. 如果一个 String 对象已经被创建过了，那么就会**从 String Pool 中取得引用**。只有 String 是不可变的，才可能使用 String Pool。

3.安全性，**String 经常作为参数，String 不可变性可以保证参数不可变**。例如在作为网络连接参数的情况下如果 String 是可变的，那么在网络连接过程中，String被改变，改变String 的那一方以为现在连接的是其它主机，而实际情况却不一定是。

4. 线程安全. String 不可变性天生具备线程安全，**可以在多个线程中安全地使用**。

引：在 Java 9 之后，String 类的实现改用 byte 数组存储字符串{ private final byte[] value;} value 数组被声明为 final，这意味着 value 数组初始化之后就不能再引用其它数组。并且 String 内部没有改变 value 数组的方法，因此可以保证 String 不可变。

**Q4. 什么是深拷贝和浅拷贝？clone方法执行的是什么拷贝？如何解决？**

A:浅拷贝是指拷贝对象和原对象的引用类型引用同一个对象，深拷贝是指拷贝对象和原对象的引用类型引用不同的对象。Clone方法**返回一个新的对象**，执行的是浅拷贝，如果要实现深拷贝必须实现cloneable接口，实现clone方法，在clone方法内部，把该对象引用的其他对象也clone一份。但是不建议使用clone方法，使用 clone() 方法来拷贝一个对象即复杂又有风险，它会抛出异常，并且还需要类型转换。Effective Java 书上讲到，最好不要去使用 clone()，可以使用**拷贝构造函数**或者拷贝工厂来拷贝一个对象。

**Q5. 抽象类和接口有什么区别？**

1. 1.从设计层面上看，抽象类提供了一种 IS-A 关系，需要满足里式替换原则，即子类对象必须能够替换掉所有父类对象。而接口更像是一种 LIKE-A 关系，它只是提供一种方法实现契约，并不要求接口和实现接口的类具有 IS-A 关系。

2.从使用上来看，**一个类可以实现多个接口，但是不能继承多个抽象类**。

3.接口的字段只能是 static 和 final 类型的，而抽象类的字段没有这种限制。

4.接口的成员只能是 public 的，而抽象类的成员可以有多种访问权限。

拓展：JAVA8后接口也可以像抽象类一样，有默认的方法实现，需要加default关键字；JAVA9后的接口也能实现private方法，但是要有具体实现内容

**Q6. 创建线程有几种方式？哪种更好？**

A:

1. 继承thread类，覆写run方法 无返回值
2. 实现runnable接口，创建新线程传入
3. 线程池创建
4. 创建Callable接口的实现类，并实现call()方法，该call()方法将作为线程执行体，并且有返回值。使用futureTask 类实例化FutureTask <String>futureTask=new FutureTask<>(callable);

实现接口的方式会更好一些，因为JAVA不支持多重继承，但是可以实现多个接口。另一方面类可能只要求可执行就行，继承整个Thread类开销过大.

**Q7. 概括说下线程的几种可用状态？**

A：新建，线程新创建还未使用。

可运行：线程位于可运行线程池中，等待被线程调度选中，获取CPU使用权。

运行：获得了CPU时间片，执行程序代码

阻塞：等待阻塞（wait()）,同步阻塞（同步锁），其他阻塞（sleep（）或者join（）等）

死亡：线程结束生命周期。

**Q8. 为什么重写equals还要重写hashcode？**

A: 如果没有重写hashcode方法，不同对象他们的hashcode返回地址值，当使用hashmap时，要同时重写hashcode方法，因为对同一个KEY值，重写hashcode方法可以保证他们的hashcode返回值相同，这样hashmap可以根据key定位到指定的内容，equals方法可以保证对象中的内容是相同的，而且先判断hashcode内容可以一定程度上提升效率，不需要再判断equeals方法。

JVM

**Q1. JVM有哪几种垃圾回收机制？**

1. 1.标记-清除：分为两个阶段，标记所有需要回收的对象，然后统一回收，产生大量碎片；2.复制：把内存划分为两个相等的区域，垃圾回收时，把正在使用的对象复制到另外一个区域中，能够进行相应的整理，不会出现碎片问题，但是耗费两倍空间；

3. 标记-整理：结合前面两个，第一阶段从根节点开始标记所有被引用对象，第二阶段遍历整个堆，清除未标记对象并且把存活对象“压缩”到堆的其中一块，按顺序排放。

4.分代收集算法：基于不同的生命周期，分为年轻代，所有新对象产生的地方，在多次GC周期后，会转移到年老代，年老代，持久代，存放静态文件，如类方法等。

SQL

**Q1. 数据库事务？**

A: 事务就是为了解决一组查询要么全部执行成功，要么全部执行失败。MySQL事务默认是采取自动提交的模式，除非显示开始一个事务。

**Q2. 事务的四大特征？（ACID）**

A: (1)原子性：整个事务的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚（ROLLBACK）到事务开始前的状态，就像这个事务从来没有执行过一样。

（2）一致性：在事务开始之前和事务结束之后，数据库的完整性约束没有被破坏。

（3）隔离性：隔离状态执行事务，使他们好像是系统在给定时间执行的唯一操作。如果有两个事务，运行在相同的时间内，执行相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。这种熟悉有时称为串行化，为了防止事务操作间的混淆，必须串行化或者序列化请求，使得在同一时间仅有一个请求应用同一数据。

（4）持久性：在事务完成以后，该事务对数据库所作的更改便持久的保存在数据库之中，不会被回滚。

**Q3. MySQL中四种隔离级别分别是什么？**

A：

读未提交：未提交读隔离级别也叫读脏，就是事务可以读取其他事务未提交的数据；

读已提交：在其他数据库中如SQL SERVE默认的隔离级别就是提交读，已提交读隔离级别就是在事务未提交之前所作的修改其他事务是不可见的。

可重复读：保证同一个事务中的多次相同查询的结果是一致的。是MYSQL的默认隔离级别。比如一个事务开始查询了一条记录然后过几秒又执行了相同的查询。保证两次查询的结果是相同的。

可串行化：可串行化就是保证读取的范围内没有新的数据插入，比如事务第一次查询得到某个范围的数据，第二次查询也得到相同范围的数据，中间没有新的数据插入到该范围中。

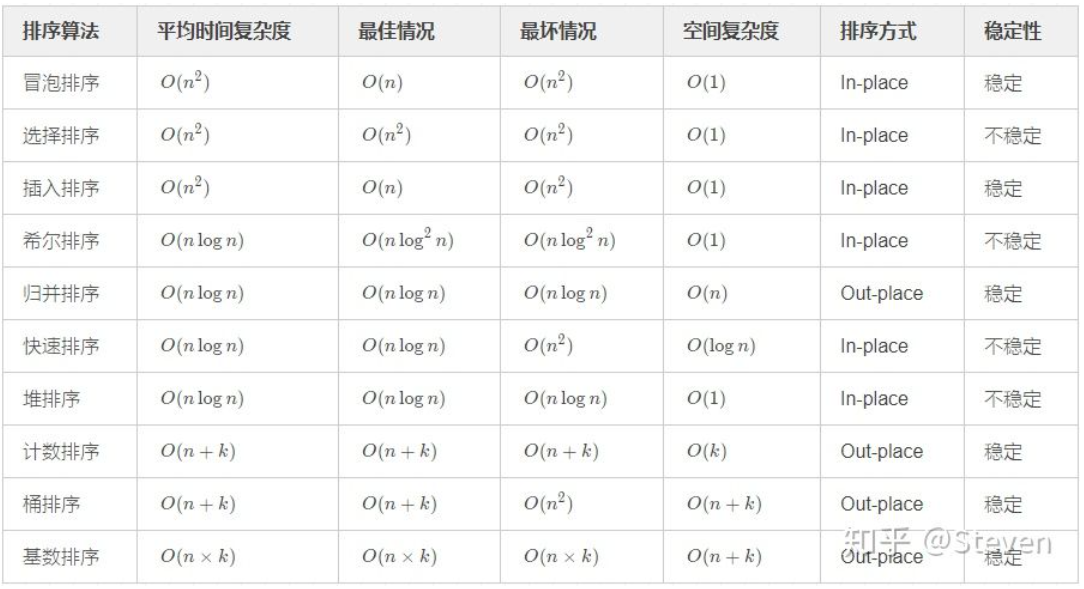
**Q4. Where和having的区别？**

A:1.where在分组之前进行限定，如果不满足条件，则不参与分组；Having则在分组之后进行限定，如果不满足结果，则不会被查询出来。

2.where后不可以跟聚合函数，having可以进行聚合函数的判断。

算法

**Q1. 各种排序的时间/空间复杂度？**

A：

计算机网络

**Q1. 为什么需要三次握手？**

A: TCP连接是全双工的，三次握手才能保证两个方向都是联通的；为了防止已失效的连接请求报文段(发出的第一个连接请求报文段并没有丢失，而是在某个网络结点长时间的滞留了，以致延误到连接释放以后的某个时间才到达server),突然又传送到了服务端，因而产生错误.

操作系统

Q1. 进程和线程的区别？

A: 进程有独立的地址空间，而线程只是一个进程中的不同执行路径。进程之间切换时耗费资源较大效率较低，多线程适合高并发操作；一个持续至少有一个进程，一个进程至少有一个线程；线程划分尺度小于进程，使得多线程程序并发性高；进程在执行过程中拥有独立的内存单元，而多个线程共享内存；虽然多线程在一个应用程序中，可以看起来时有多个执行部分可以同时执行，但是操作系统并没有将多个线程看作独立的应用来实现进程的调度和管理以及资源发呢配。

两个有序[链表](https://www.nowcoder.com/jump/super-jump/word?word=%E9%93%BE%E8%A1%A8)合并

[岛屿数量](https://www.nowcoder.com/jump/super-jump/word?word=%E5%B2%9B%E5%B1%BF%E6%95%B0%E9%87%8F)问题

有效的括号