Java 基础题（上）

1、 反射的理解

答：反射就是把 Java 类中的各种成分映射成相应的 java 类。

a、 变量，方法，构造方法，包等等信息用一个个 java 类来表示。（Field、Method、Contructor、Package）

b、 举个例子：房子中　有衣柜、电视、床类

c、 Java 中提 Class 类中提供了很多方法获得相应的类，并可以通过这个类中的 newIstance 方法获得实例对象，进一步操作。

简单总结一点：反射就其实是对 java 中各个类中共性部分进行封装，这也是面向对象的一种特性。

参考：反射就是把 Java 类中的各种成分映射成相应的 java 类。例如，一个 Java 类中用一个 Class 类的对象来表示，一个类中的组成部分：成员变量，方法，构造方法，包等等信息也用一个个的 Java 类来表示，就像汽车是一个类，汽车中的发动机，变速箱等等也是一个个的类。表示 java 类的 Class 类显然要提供一系列的方法，来获得其中的变量，方法，构造方法，修饰符，包等信息，这些信息就是用相应类的实例对象来表示，它们是 Field、Method、Contructor、Package 等等。

一个类中的每个成员都可以用相应的反射 API 类的一个实例对象来表示，通过调用 Class 类的方法可以得到这些实例对象后，得到这些实例对象后有什么用呢？怎么用呢？这正是学习和应用反射的要点。

2、 集合框架的概述，ArrayList 的特点、Map 集合的特点；Collection 集合下有哪些类，每个类又有那些特点？HashMap 和 Hashtable 的区别？

集合框架的概述：集合类主要负责保存、盛装其他数据，因此集合类也被称为容器类。集合类和数组类不一样，数组元素既可以是基本数据类型，也可以是对象（实际上保存的是对象的引用变量）；而集合只能保存对象（实际上只是保存对象的引用变量）。Java 的集合类主要由两个接口派生而出：Collection 和 Map，这两个是集合框架的根接口。

Map 集合的特点：Map 集合用于保存具有映射关系的数据，因此 Map 集合里保存着两组值，一组使用于保存 Map 里的 key，另外一组用于保存 Map 里的 value，key 和 value 都可以是任何引用类型的数据。Map 的 key 不允许重复。Key 和 value 之间存在单向一对一关系，即通过指定的 key，总能找到唯一的确定的 value。Map 里包含了一个 keySet()方法，用于返回 Map 里所有 key 组成的 Set 集合。

Collection 集合下包含的类：包含 List 集合和 Set 集合。

List 集合包含 ArrayList、LinkedList、Vector。List 集合特点：有序，有索引，可以存放重复元素。

ArrayList：底层是数组结构，是不同步的，线程不安全，增删慢，查寻速度快；

LinkedList：底层是链表结构，增删比较快，查寻比较慢；

Vector：底层是数组结构，是同步的，线程安全，查询速度慢。

Set 集合包含 HashSet、TreeSet、LinkedHashSet。Set 集合特点：无序，无索引，不可存放重复元素。

HashSet：线程不安全，存取速度快；

TreeSet：线程不安全，可以 Set 集合中的元素进行排序。

LinkedHashSet：底层是链表实现的，是 Set 集合中惟一一个能保证怎么存就怎么取的集合对象。

HashMap 和 Hashtable 的区别：

A、共同点：底层都是哈希算法，都是双列结合

B、区别：HashMap 可以存储 null 键和 null 值，是线程不安全的，效率高，JDK1.2 版本；

Hashtable 不可以存储 null 键和 null 值，线程安全，效率低，JDK1.0 版本。

3、 懒汉式和饿汉式之间的区别

饿汉式：线程安全，调用效率高，创建对象再对外提供对象的，不能延时加载；

懒汉式：线程不安全，需要同步调用效率不高，是后创建再对外提供对象的，能延时加载。

4、 进程和线程之间的区别

进程：当一个程序进入内存运行时，即变成一个进程。进程是处于运行过程中的程序，并且具有一定的独立功能，进程是系统进行资源分配和调度的一个独立单位。一般而言，进程包含三个特征：独立性、动态性、并发性。

线程：也被称为轻量级进程，线程是进程的执行单元。线程在程序中是独立的、并发的执行流，对应绝大多数程序而言，通常仅要求有一个主线程。

线程是进程的组成部分，一个进程可以拥有多个线程，一个线程必有又一个父进程。线程可以拥有自己的堆栈、自己的程序计数器和自己的局部变量，但不拥有系统资源，它与父进程的其他线程共享该进程所拥有的全部资源。

线程是独立运行的，其执行是抢占式的。

线程比进程具有更高的性能。

一个线程可以创建和撤销另一个线程，同一个进程中的多个线程之间可以并发执行。

简而言之，一个程序运行后至少有一个进程，一个进程里可以包含多个线程，但至少要包含一个线程。

5、 线程的创建方式

线程创建有 2 种方式：继承 Thread 类和实现 Runnable 接口

A、继承 Thread：

a、 定义 Thread 类的子类，并重写该类的 run()方法，run()方法称为线程执行体；

b、 创建 Thread 子类的实例，即创建了线程对象；

c、 调用线程对象的 start()方法来启动该线程。

B、实现 Runnable 接口：

a、 定义 Runnable 接口的实现类，并重写该接口的 run()方法；

b、 创建 Runnable 实现类的实例，并以此实例作为 Thread 的 target 来创建 Thread 对象，该 Thread 对象才是真正的线程对象。

c、 调用线程对象的 start()方法来启动该线程。

C、两种方式的优缺点对比：

a、 继承 Thread：

优点：编写简单，如果需要访问当前线程，无须使用 Thread.currentThread()方法，直接使用 this 即可获得当前线程。

缺点：因为线程类已经继承了 Thread 类，所以不能再继承其他父类。

b、 实现 Runnable：

优点：可以继承其他父类；同时，在多个线程中可以共享同一个 target 对象，非常适合多个相同线程来处理同一份资源的情况。

缺点：编程稍稍复杂，如果需要访问当前进程，则必须使用 Thread.currentThread()方法。

6、 线程中 start()方法和 run()区别

启动线程使用 start()方法而不是 run()方法！

调用 start()方法来启动线程，系统会把该 run()方法当成线程执行体来处理；但是如果直接调用线程对象的 run()方法，则 run()方法会立即被执行，而在 run()方法返回之前其他线程无法并发执行，也就是说，如果直接调用线程对象的 run()方法，系统把线程对象当成一个普通对象，而 run()方法也是一个普通方法，而不是线程执行体。

7、 同步代码块和同步方法的区别

同步就是将部分操作功能数据的代码进行加锁；

同步方法是指进入该方法时需要获取 this 对象的同步锁；

同步代码块则是可以指定需要获取哪个对象的同步锁。

相对而言，同步方法比同步代码块更高效，但是它们的功能是一样的。

8、 线程的死锁

当两个线程相互等待对方释放同步监视器时就会发生死锁。一旦出现死锁，整个程序既不会发生任何异常，也不会给出任何提示，只是所有的线程处于阻塞状态，无法继续。

9、 sleep 和 wait 的区别

A、 sleep 必须传入参数，wait 可传入参数，也可以不传入参数，传入指在参数的时间结束后等待；

B、 sleep 在同步方法或者同步代码中，不释放锁，wait 则一定会释放锁。

C、 sleep 是 Thread 类的方法，wait 是 Object 类的方法。

10、线程池的理解

系统启动一个新线程的成本比较高，因为它涉及与操作系统交互。在这种情况下，使用线程池可以很好地提高性能，尤其是当程序中需要创建大量生存期很短暂的线程时，更应该考虑使用线程池。

线程池在系统启动时即创建大量空闲的线程，程序将一个 Runnable 对象传给线程池，线程池就会启动一个线程来执行它们的 run()方法，当 run()方法执行结束后，该线程并不会死亡，而是再次返回线程池中成为空闲状态，等待执行下一个 Runnable 对象的 run()方法。

11、匿名内部类

答：概念：匿名内部类就是没类名的子类，

前提：必须有继承或者实现。

应用场景：当函数的参数引用接口类时，且这个接口的方法个数不超过三个时；

12、抽象类和接口的区别

抽象类一般用于描述一个体系的共内容，定义好抽象方法，让子类去实现。

接口和抽象类都不能被实例化，都可以包含抽象方法；

接口作为系统与外界交互的窗口，接口体现的是一种规范；

抽象类则不一样，抽象类作为系统中多个子类的共同父类，它所体现的是一种模板式设计；

接口里只能包含抽象方法、静态方法和默认方法，不能为普通方法提供方法实现，抽象类则完全可以包含普通方法。

接口里只能定义静态常量，抽象类里则既可以定义普通成员变量，也可以定义静态常量。

接口里不包含构造器，抽象类里可以包含构造器；

接口里不能包含初始化块，但抽象类则完全可以包含初始化块；

一个类最多只能有一个直接父类，包含抽象类；但一个类可以直接实现多个接口，通过实现多个接口可以弥补 Java 单继承的不足。

13、面向对象和面对过程区别，面向对象的三大特征：封装、继承、多态。

相同点：都是一种编程思想。

区别：面向过程就是解决问题所需要的步骤，用函数去实现每一个步骤，强调细节。

而面向过程是一种对现实世界理解和抽象的方法。对同类事物的概括与总结，隐藏具体属性和行为，强调整体。

三个特征：

A 将不需要对外提供的内容都隐藏起来，把属性都隐藏，提供公共方法对其访问。好处是将变化隔离；便于使用；提高重用性；安全性。

B 继承是代码的一种重用方式，好处是提高了代码的复用性，让类之间有关系，给多态提供了前提。

C 多态的体现是父亲或接口引用指向子类对象，提高了程序的扩展性，前提是子类必须有继承或实现。

14、获取字节码文件的三种方式

A、第一种：使用 Class 类的 forName(String clazzName)静态方法；

B、第二种：调用某个类的 class 属性来获取该类对应的 Class 对象；

C、第三种：调用某个对象的 getClass()方法

15、反射创建对象的两种方式

A、使用 Class 对象的 newInstance()方法来创建该 Class 对象对应的实例，这种方式要求该 Class 对象的对应类有默认构造器；

B、先使用 Class 对象获取指定的 Construtor 对象，再调用 Constructor 对象的 newInstance()方法来创建该 Class 对象对应类的实例。

16、反射的理解

当一个程序在编译时根本无法预知该对象和类可能属于哪个类，程序只依靠运行时信息来发现该对象和类的真实信息，这就必须使用反射；一旦获得了某个类所对应的 Class 对象之后，程序就可以调用 Class 对象的方法来获得该对象和该类的真实信息了。

17、数组的理解

A、在一个数组中，数组的元素类型是唯一的，即一个数组里只能存储一种数据类型的数据，而不能存储多种数据类型的数据；

B、一旦数组初始化完成，数组在内存中所占的空间将被固定下来，因此数组的长度将不可改变；

C、数组既可以存储基本类型的数据，也可以存储引用类型的数据。

18、equals 与==的区别

==比较两个对象在内存里是不是同一个对象，就是说在内存里的存储位置一致。两个 String 对象存储的值是一样的，但有可能在内存里存储在不同的地方 。

==比较的是引用而 equals 方法比较的是内容。public boolean equals(Object obj) 这个方法是由 Object 对象提供的，可以由子类进行重写。默认的实现只有当对象和自身进行比较时才会返回 true,这个时候和==是等价的。String, BitSet, Date, 和 File 都对 equals 方法进行了重写，对两个 String 对象 而言，值相等意味着它们包含同样的字符序列。对于基本类型的包装类来说，值相等意味着对应的基本类型的值一样。

19、finalize、final、finally 三者之间的关系

final: 常量声明。接口里声明的变量默认是 final 的。final 类无法继承，也就是没有子类。这么做是出于基础类型的安全考虑，比如 String 和 Integer。这样也使得编译器进行一些优化，更容易保证线程的安全性。final 方法无法重写。final 变量的值不能改变。

finalize: 帮助进行垃圾回收。finalize()方法在一个对象被销毁和回收前会被调用。

finally: 通常用于异常处理，不管有没有异常被抛出都会执行到。比如，关闭连接通常放到 finally 块中完成。

20、在 System.out.println()里面,System, out, println 分别是什么?

System 是系统提供的预定义的 final 类，out 是一个 PrintStream 对象，println 是 out 对象里面一个重载的方法。

21、Java 的访问修饰符是什么?

访问权限修饰符是表明类成员的访问权限类型的关键字。使用这些关键字来限定程序的方法或者变量的访问权限。它们包含：

public: 所有类都可以访问

protected: 同一个包内以及所有子类都可以访问

private: 只有归属的类才能访问

默认: 归属类及相同包下的子类可以访问。

22、静态类型有什么特点?

静态变量是和类绑定到一起的，而不是类的实例对象。每一个实例对象都共享同样一份静态变量。也就是说，一个类的静态变量只有一份，不管它有多少个对象。类变量或者说静态变量是通过 static 这个关键字来声明的。类变量通常被用作常量。静态变量通常通过类名字来进行访问。当程序运行的时候这个变量就会创建直到程序结束后才会被销毁。类变量的作用域和实例变量是一样的。它的初始值和成员变量也是一样的，当变量没被初始化的时候根据它的数据类型，会有一个默认值。类似的，静态方法是属于类的方法，而不是类对象，它的调用并不作用于类对象，也不需要创建任何的类实例。静态方法本身就是 final 的，因为重写只会发生在类实例上，静态方法是和类绑定在一起的，不是对象。父类的静态方法会被子类的静态方法屏蔽，只要原来方法没有声明为 final。非静态方法不能重写静态方法，也就是说，你不能在子类中把一个静态方法改成实例方法。

非静态变量在每一个对象实例上都有单独的一份值。

23、StringBuffer 和 StringBuilder 的区别

相同点：都表示一个字符序列可变的字符串；

区别：StringBuffer 是线程安全的，StringBuilder 是线程不安全的，性能略高。

24、增强 for 循环与普通 for 循环的区别

增强 for 循环必须有被遍历的目标（如集合或数组）。

普通 for 循环遍历数组的时候需要索引。

增强 for 循环不能获取下标，所以遍历数组时最好使用普通 for 循环。

特点：书写简洁。对集合进行遍历，只能获取集合元素，不能对集合进行操作，类似迭代器的简写形式，但是迭代器可以对元素进行 remove 操作（ListIterator 可以进行增删改查的操作）。

25、throws 和 throw 的区别

throws： 声明抛出只能在方法签名中使用，可以声明抛出多个异常类，一旦使用了 throws 语句声明抛出该异常，程序就无须使用 try…catch 块来捕获该异常了。

throw： 如果需要在程序中自行抛出异常，则应该使用 throw 语句，throw 语句可以单独使用，throw 语句抛出的不是异常类，而是一个异常实例，而且每次只能抛出一个异常实例。

26、什么是泛型

泛型就是允许在定义类、接口、方法时 类型形参，这个类型形参在声明变量、创建对象、调用方法时动态地指定（即传入实际的类型参数，也可称为类型实参）。

包含泛型声明的类型可以在定义变量。创建对象时传入一个类型实参，从而可以动态地生成无数多个逻辑上的子类，但这种子类在物理上并不存在。

27、多线程编程的好处是什么？

在多线程程序中，多个线程被并发的执行以提高程序的效率，CPU 不会因为某个线程需要等待资源而进入空闲状态。多个线程共享堆内存(heap memory)，因此创建多个线程去执行一些任务会比创建多个进程更好。举个例子，Servlets 比 CGI 更好，是因为 Servlets 支持多线程而 CGI 不支持。

28、为什么线程通信的方法 wait(), notify()和 notifyAll()被定义在 Object 类里？

Java 的每个对象中都有一个锁(monitor，也可以成为监视器) 并且 wait()，notify()等方法用于等待对象的锁或者通知其他线程对象的监视器可用。在 Java 的线程中并没有可供任何对象使用的锁和同步器。这就是为什么这些方法是 Object 类的一部分，这样 Java 的每一个类都有用于线程间通信的基本方法。

29、同步方法和同步块，哪个是更好的选择？

同步块是更好的选择，因为它不会锁住整个对象（当然你也可以让它锁住整个对象）。同步方法会锁住整个对象，哪怕这个类中有多个不相关联的同步块，这通常会导致他们停止执行并需要等待获得这个对象上的锁。

30、集合框架中的泛型有什么优点？

Java1.5 引入了泛型，所有的集合接口和实现都大量地使用它。泛型允许我们为集合提供一个可以容纳的对象类型，因此，如果你添加其它类型的任何元素，它会在编译时报错。这避免了在运行时出现 ClassCastException，因为你将会在编译时得到报错信息。泛型也使得代码整洁，我们不需要使用显式转换和 instanceOf 操作符。它也给运行时带来好处，因为不会产生类型检查的字节码指令。

31、什么是装饰设计模式？装饰设计模式和继承有何区别？

装饰设计模式:当想要对已有的对象进行功能增强时，可以定义类，将已有对象传入，基于已有的功能，并提供加强功能。那么自定义的该类称为装饰类。

装饰类通常会通过构造方法接收被装饰的对象。并基于被装饰的对象的功能，提供更强的功能。

和继承的区别：

装饰模式比继承要灵活。避免了继承体系臃肿。而且降低了类于类之间的关系。装饰类因为增强已有对象，具备的功能和已有的是相同的，只不过提供了更强功能。所以装饰类和被装饰类通常是都属于一个体系中的。

32、枚举简介

A、在程序中可以使用一个枚举来指定对象的取值范围；

B、在 Java 中使用 enum 关键字定义一个枚举类，每一个枚举类都是继承 Enum 类；

C、在枚举类中可以通过 values()方法取得枚举中的全部内容；

D、在枚举类中可以定义构造方法，但在设置枚举范围时必须显式地调用构造方法；

E、所有的枚举类都可以直接使用 Comparable 进行排序，因为 Enum 类实现了 Comparable 接口；

F、Java 类集中提供枚举的支持类 EnumMap、EnumSet；

G、一个枚举类可以实现一个接口或者直接定义一个抽象方法，但是枚举中的每个元素都必须分别实现全部的抽象方法。

33、什么是构造函数，它与一般函数有什么区别？

构造函数：用于给对象进行初始化，是给与之对应的对象进行初始化，它具有针对性，函数中的一种。

特点：

1：该函数的名称和所在类的名称相同。

2：不需要定义返回值类型。

3：该函数没有具体的返回值。

记住：所有对象创建时，都需要初始化才可以使用。

构造函数和一般函数有什么区别呢？

1：两个函数定义格式不同。

2：构造函数是在对象创建时，就被调用，用于初始化，而且初始化动作只执行一次。

一般函数，是对象创建后，需要调用才执行，可以被调用多次。

34、构造代码块和构造函数有什么区别？

构造代码块：是给所有的对象进行初始化，也就是说，所有的对象都会调用一个代码块。只要对象一建立。就会调用这个代码块。

构造函数：是给与之对应的对象进行初始化。它具有针对性。

静态代码块：就是一个有静态关键字标示的一个代码块区域。定义在类中。

作用：可以完成类的初始化。静态代码块随着类的加载而执行，而且只执行一次（new 多个对象就只执行一次）。

如果和主函数在同一类中，优先于主函数执行。

静态代码块、构造代码块、构造函数同时存在时的执行顺序：静态代码块 -->构造代码块 --> 构造函数；

35、重载和重写的区别

1.重写必须继承，重载不用。

2.重写的方法名，参数数目相同，参数类型兼容，重载的方法名相同，参数列表不同。

3.重写的方法修饰符大于等于父类的方法，重载和修饰符无关。

4.重写不可以抛出父类没有抛出的一般异常，可以抛出运行时异常

36、|、&与||、&&的区别

首先 &和| 称为布尔运算符，&&和|| 称为条件布尔运算符。

两种运算符得到的结果完全相同，但得到结果的方式又一个重要区别：条件布尔运算符性能比较好。他检查第一个操作数的值，再根据该操作数的值进行操作，可能根本就不处理第二个操作数。

如果&&运算符的第一个操作数是 false，就不需要考虑第二个操作数的值了，因为无论第二个操作数的值是什么，其结果都是 false。同样，如果第一个操作数是 true，||运算符就返回 true，无需考虑第二个操作数的值。但&和|却不是这样，它们总是要计算两个操作数。

因为操作数的就算是有条件的，如果使用&&和||运算符来代替&和|，性能会有一定提高。

37、类和对象的区别

类是拥有相同行为特征对象的一个抽象概念，而对象是类这个抽象概念中事实存在的个体。

打个比方：类;人，对象：你。人这个类可以是黄种人、黑种人、白种人等等，但是对象实实在在的就是指类当中的一个个体。

38、JDK1.5 新特性有哪些

泛型、增强的 for 循环、自动装箱/ 自动拆箱、类型安全的枚举、静态导入、可变参数。

38、为什么配置 path，什么时候配置 classpath？

PATH 为了便捷。系统变量 Path 告诉操作系统可执行文件(*.exe、*.bat 等)所在的路径；

不把 JDK 的 bin 目录加到 PATH 的话，需要 path\to\jdk\bin\javac Xxx.java 这样编译程序，加到 PATH 后，可以直接 javac Xxxx.java， 输入省事多了。

39、栈内存和堆内存的区别

最主要的区别就是**栈内存用来存储局部变量和方法调用；堆内存用来存储 Java 中的对象。**

无论是成员变量，局部变量，还是类变量，它们指向的对象都存储在堆内存中。

栈内存归属于单个线程，每个线程都会有一个栈内存，其存储的变量只能在其所属线程中可见，即栈内存可以理解成线程的私有内存。

堆内存中的对象对所有线程可见。堆内存中的对象可以被所有线程访问。

栈的内存要远远小于堆内存，如果你使用递归的话，那么你的栈很快就会充满，导致内存溢出。

40、抽象类为什么不能与 private、final、static 共存？

抽象方法的最实质的意义在于被未来的子类覆盖实现掉。它自己是个空方法。

private 的实质意义在于本类其他方法调用它。你自己是个空方法，别人调用你有什么用？所以 abstract 和 private 在一起毫无意义。

final 规定子类不能再覆盖它。abstract 是专等着要别人来覆盖，二者矛盾。所以不能放在一起使用。

static 的成员是在程序执行到 main 时就已经确定的。动态方法调度就是指在运行时才决定执行哪个方法（是子类的还是父类的）。 比如运行时如果用户输入自行车，就执行自行车的驾驶方法。而 abstract 一定用到动态方法调度。所以这静态和动态二者是矛盾的，不能一起用。

41、为什么接口中的常量必须使用 public static final 修饰

首先接口是一种高度抽象的"模版"，而接口中的属性也就是“模版”的成员，就应当是所有实现"模版"的实现类的共有特性，所以它是 public 的 ,是所有实现类共有的。否则具体实现类不能使用该变量，则就没有了定义的意义。

static 的原因是，假如可以是非 static 的话，因一个类可以继承多个接口，出现重名的变量，如何区分呢？所以必须定义成 static 的，以便区分。

其次,接口中如果可能定义非 final 的变量的话，而方法又都是 abstract 的，这就自相矛盾了，有可变成员变量但对应的方法却无法操作这些变量，虽然可以直接修改这些静态成员变量的值，但所有实现类对应的值都被修改了，这跟抽象类有何区别？又接口是一种更高层面的抽象，是一种规范、功能定义的声明，所有可变的东西都应该归属到实现类中，这样接口才能起到标准化、规范化的作用。所以接口中的属性必然是 final 的。

42、关于 compartor 和 comparable 接口的区别

Comparable & Comparator 都是用来实现集合中元素的比较、排序的，只是 Comparable 是在集合内部定义的方法实现的排序，Comparator 是在集合外部实现的排序，所以，如想实现排序，就需要在集合外定义 Comparator 接口的方法或在集合内实现 Comparable 接口的方法。

Comparator 位于包 Java.util 下，而 Comparable 位于包 java.lang 下

Comparable 是一个对象本身就已经支持自比较所需要实现的接口（如 String、Integer 自己就可以完成比较大小操作，已经实现了 Comparable 接口）

而 Comparator 是一个专用的比较器，当这个对象不支持自比较或者自比较函数不能满足你的要求时，你可以写一个比较器来完成两个对象之间大小的比较。

43、TCP 与 UDP 区别总结：

A、TCP 面向连接（如打电话要先拨号建立连接）;UDP 是无连接的，即发送数据之前不需要建立连接

B、TCP 提供可靠的服务。也就是说，通过 TCP 连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP 尽最大努力交付，即不保证可靠交付

C、TCP 面向字节流，实际上是 TCP 把数据看成一连串无结构的字节流;UDP 是面向报文的

UDP 没有拥塞控制，因此网络出现拥塞不会使源主机的发送速率降低（对实时应用很有用，如 IP 电话，实时视频会议等）

D、每一条 TCP 连接只能是点到点的;UDP 支持一对一，一对多，多对一和多对多的交互通信

E、TCP 首部开销 20 字节;UDP 的首部开销小，只有 8 个字节

F、TCP 的逻辑通信信道是全双工的可靠信道，UDP 则是不可靠信道