1.Java 中的几种基本数据类型是什么？对应的包装类型是什么？各自占用多少字节呢？

基本数据类型：byte,short,char,int,long,float,dobule,boolean,

包装类：Byte,Short,Character,Integer,,Long.Float,Double,Boolean

占用字节：

基本类型：1,2,2,4,8,4,8,(boolean在被编译成int类型时和int一样是四个，当在orcale java中boolean是1字节)

2.String、StringBuffer 和StringBuild的区别是什么?String为什么是不可变的?

String是不可变字符串，StringBuffer 和 StringBuild是可变字符串，StringBuffer 和StringBuild的区别是，StringBuffer是安全的，但是StringBuild速度更快，优先选用安全的。

String源码上看是被final修饰的，表示不可变，同时String的参数有两个，一个是字符数字value一个是hash，而字符数组也是用final修饰的 ，且参数都是用private修饰的，并没有提供set方法，让String只能在创建的时候复制，之后无法修改值。当然如果真的想要修改值，也可以使用反射的方式进行强制修改。

3.== 与 equals?hashCode 与 equals ?

==比较的是地址值，而equals比较的是内容。

hashCode在java中可以分成两种，不会创建类对应的散列表，这种情况下hashCode和equals就没有特别大的关系，hashCode并不影响判断。而在会创建类对应的散列表的时候，hashCode就会参与比较，如果两个对象相等，则hashCode一定相等，反之则不一定。所以这时候是先通过hashCode来提高对比效率。

4.Java 反射？反射有什么缺点？你是怎么理解反射的（为什么框架需要反射）？

Java反射是在代码运行期间来实现对项目代码的控制，

反射的缺点会有性能影响，但是只是在超大数据量的时候才会出现，反射会绕过源代码，给维护带来困难，反射能够执行一些正常情况下不允许操作的事情，比如通过发射可以在一个List<Integet>中存入字符串。

像在使用aop时 只有使用反射才能知道调用当前方式时的实时数据，能够更加灵活的操作项目。同时使用泛型时也可以运用到反射来对存入的参数进行操作，不将代码写死，spring中大部分都是配置信息，只有通过反射才能根据配置的不同加载不同类。

5.谈谈对 Java 注解的理解，解决了什么问题？

Java注解分为好几种，一种是java自带的注解，比如@override这种注解最重要的目的是为了标识，一种框架中的注解，比如@Controller这种注解有实际作用，还有一只注解被称为元注解，适用于标注注解的一种注解，主要使用的是两种：@Target 用于设置这个注解的使用范围，还有一种@Rentention，用来标注注解的运行时期，一共分为三个时期：SOUCE,CALSS,RUNTIME，其中SOURCE和CLASS是在项目编译时期使用的注解，当项目编译完成后注解失效，比较著名的有lombok的注解类，还有一种RUNTIME的注解是我们平常用的最多的注解，在项目运行时生效，比如spring的aop或者我们自定义的注解日志类都是通过注解和反射的方式进行实现的。

6.Java 泛型了解么？什么是类型擦除？介绍一下常用的通配符？

Java泛型是通过通配符来标注传入的对象，当我们不知道传入的参数的时候，就可以使用泛型来进行标注，再通过反射来获取其中的数据，泛型的好处是相比较于强转，泛型不会报类型转换异常，而且代码的复用性更高，大多数适用于组件的封装。

类型擦除就是泛型只在编译器生效，到运行期间，泛型所表示的含义都会变化，比如List<Integer>和List<String>他们getClass方法获取的数据是相同的，这就是因为类型擦除让他们的在运行期间变成了Object ，当然如果泛型的表示方法为<? extends T> 那么则会变成T类型。

常用的通配符包括：T表示一般对象 ；E表示警告；K表示key ；V表示Value（一般和K一起用）；S表示Subtype

7.内部类了解吗？匿名内部类了解吗？

内部类其实就是在一个类内部声明的一个类，他有可以访问外部类所有参数和方法，包括private包裹的，外部类想要访问内部类的参数则需要通过创建对象的方式，匿名内部类本质就是继承该类或者实现接口的子类匿名对象

8.BIO,NIO,AIO 有什么区别?

BIO是同步阻塞IO也就是我们，即当请求来时进行处理，只有处理好才会返回响应，是最传统的IO模型。

NIO是同步非阻塞IO，和BIO的区别主要是引入了缓冲区，当请求来时会让数据先加载进入缓冲区，当加载完成后在进行IO操作。监控任务则有select或epoll来完成，select是通过轮询的方式来进行监控，每次都遍历缓冲区，每当有完成的就进行任务，而epoll则是通过将已经完成的放入到指定的区域，每次都去指定区域调取任务就行了。

AIO是在NIO的基础上优化实现异步非阻塞IO，因为每次NIO的轮询操作还是在阻塞的，所以提出了AIO就是当缓冲区加载完成后，主动提醒线程，或者设置回调函数还是先IO操作，但是犹豫实现过于复杂，应用范围没有NIO广。