一、面向对象

1. 面向对象提纲：
   (1) 面向对象和面向过程的概念
   (2) 面向对象的三大特征：封装、继承、多态
   (3) 面向对象编程常用的关键字：
   this：访问本类的成员变量、成员方法、构造方法
   super：访问父类的成员变量、成员方法、构造方法
   static：随着类加载，被对象共享，可以类名调用、存储在方法区
   final：最终类不可被继承、最终方法不可被重写、变量是常量
   (4) 内部类、匿名内部类
   (5) 抽象类与接口
   (6) 异常：编译时异常和运行时异常；throw 方法体内抛异常对象、throws 方法上抛异常类；try-catch-finally；常见的异常
   (7) 成员变量、局部变量、静态变量
2. 什么是面向对象？
   面向对象是一种符合人类思维习惯的编程思想，面向对象是在程序中使用对象来映射现实中的事物，使用对象的关系来描述事物之间的联系。

3. 面向对象与面向过程的区别

(1) 面向过程就是分析解决问题所需要的步骤，然后用函数把这些步骤一一实现，使用的时候一个一个的调用就可以了。面向过程强调的是功能行为，解决问题的步骤，以函数为基础来完成各种操作。

(2) 面向对象是基于面向过程的，面向对象是把需要解决的问题按照一定规则划分为多个独立的对象，强调的是对象，把功能封装成对象，让对象去调用方法来解决问题。

4. 面向对象的特点：封装性、继承性、多态性

(1) 封装

把对象的属性和具体的实现细节隐藏起来，仅对外提供公共的访问方法。

所谓类的封装是指在定义一个类的时候，将类中的属性私有化，即使用 private 来修饰，私有属性只能在本类中被访问。set、get 方法。

好处：将变化隔离；便于使用；提高重用性；安全性。

(2) 继承

描述的是类与类之间的所属关系，通过继承，可以在无须重新编写原有类的情况下，对原有类的功能进行扩展。

PS：子类方法的权限必须大于等于父类方法权限

好处：提高了代码的重用性；让类与类之间产生了关系，是多态的前提。

弊端：打破了封装性。

重写：子类重新实现父类的方法，方法声明相同：方法名、参数列表、返回值类型相同，方法体不同。

super 关键字：可以理解为父类对象，谁调用代表谁父类。

(1) super.成员变量：父类的成员变量

(2) super.成员方法(形参)：父类的成员方法

(3) super(形参)：父类的构造方法

final 关键字：是一个修饰符，可以修饰类、方法、变量

(1) final 修饰的类是最终类，不可以被继承；

(2) final 修饰的方法最终方法，不可以重写；

(3) final 修饰的变量是常量，只可以赋值一次。

(3) 抽象类

抽象类

一般用于描述一个体系单元，将一组共性内容进行抽取，特点：可以在类中定义抽象内容让子类实现，可以定义非抽象内容让子类直接使用。

接口

一般用于定义对象的扩展功能，是在继承之外还需这个对象具备的一些功能。

抽象类与接口的区别

(1) 抽象类与接口都是不断向上抽取的结果，抽象类用 abstract 修饰，接口用 interface 声明 ，有抽象方法的类一定要声明为抽象类，接口是一个特殊的抽象类，接口中的方法都是抽象方法。两者都不能实例化，继承抽象类或实现接口都需要实现其中的抽象方法

(2) 成员修饰符：前者不固定，接口固定，都是 public 修饰

(3) 成员变量：前者变量或常量；后者 final 修饰，常量

(4) 构造函数：前者有，后者无

(5) 成员方法：前者抽象或非抽象；后者都是抽象方法

(6) 抽象类只能被继承，而且只能单继承。接口需要被实现，而且可以单实现或多实现。

(7) 设计理念：被继承体现的是类与类的所属(is a)的关系，抽象类中定义的是继承体系的共性功能；被实现体现的是 like a 的关系，接口中定义的是继承体系的扩展功能。

多实现的好处：避免了单继承的局限性；还可以将类进行功能的扩展。

1. abstract 与哪些关键字不能共存为什么

   private：私有的方法是不可见的，无法被重写

   final：被 final 修饰的方法是最终方法，无法被重写

   static：被 static 修饰的方法，要随类加载到方法区，由于抽象方法没有方法体所以不能加载

(4) 多态

在程序中允许出现重名现象，它指在一个类中定义的属性和方法被其他类继承后，它们可以具有不同的数据类型或表现出不同的行为，这使得同一个属性和方法在不同的类中具有不同的语义。

多态可以理解为事物存在的多种体现形态。父类的引用指向了自己的子类对象；父类的引用也可以接收自己子类的对象。一个类中定义的属性和方法被其他类继承后，它们可以具有不同的数据类型或表现出不同的行为，这使得同一个属性和方法在不同的类中具有不同的语义。重写、重载多态性的不同表现。

向上转型、向下转型、重写、重载多态性的不同表现。

实现多态的机制

父类或接口的引用变量可以指向子类或实现类的实例对象，而程序调用的方法在运行期间才动态的绑定，具体调用的方法就是引用变量指向的具体实例对象的方法。

好处：提高了代码的扩展性

弊端：父类的引用只能访问父类中有的成员，无法调用子类中特有的方法

5. 类和对象

类：对某一类事物的抽象描述（抽象），用于描述一组对象的共同属性（成员变量）和行为（成员方法）。

对象：表示现实中某类事物的具体个体（具体）

6. this 关键字：代表当前类的对象，谁调用代表谁：哪个对象调用了 this 所在的函数，this 就代表那个对象。
   (1) this.成员变量：解决与局部变量名称冲突的问题
   (2) this.成员方法(形参)：调用本类成员方法
   (3) this(形参)：调用本类构造方法

7. static 关键字的特点

(1) 静态变量、静态方法随着类的加载而加载，优先于对象而存在，被类的所有对象所共享，可以通过类名直接调用。

(2) 静态方法只能访问静态成员，非静态方法可以访问静态或非静态成员

(3) 静态方法中不可以使用 this、super 关键字

因为 this 代表对象，而静态方法随着类加载时，有可能没有对象，所以 this 无法使用。

(4) 静态代码块 static{代码}：只执行一次，对类的成员变量进行初始化。

8. final、finally、finalize 的区别
   (1) final：是一个修饰符，可以修饰类、方法和变量，
   被 final 修饰的类是最终类，不可以被继承，
   被 final 修饰的方法是最终方法，不可以被重写，
   被 final 修饰的变量是常量，只能赋值一次
   (2) finally：用于异常处理，和 try、catch 结合使用，可以添加也可以不添加，用于

执行一些必须执行的代码，如释放资源等，除了 try 代码块有 system.exit(0)，否则 finally 的代码一定会被执行。

(3) finalize：Object 类中的方法，用于垃圾资源回收。

9. 局部变量、成员变量、静态变量的区别

(1) 成员变量：作用于整个类中，随对象存储在堆内存中，生命周期跟对象一样

(2) 局部变量：作用于方法、参数、语句中，方法或语句结束则局部变量生命周期结束，存放在栈内存中。

(3) 静态变量：静态变量属于类，随着类的加载而加载，随着类的消失而消失，被类的所有对象共享，可以通过类名直接调用；存储在方法区中。

成员变量、局部变量、静态变量的区别	
位置的不同	成员变量直接定义在类中 局部变量定义在方法中，参数上，语句中
有效范围的不同	成员变量在这个类中有效。 局部变量只在自己所属的大括号内有效，大括号结束，局部变量失去作用域。
存储的不同	成员变量存在于堆内存中，随着对象的产生而存在，消失而消失 局部变量存在于栈内存中，随着所属区域的运行而存在，结束而释放
成员变量和静态变量的区别	
所属不同	成员变量所属于对象。所以也称为实例变量。 静态变量所属于类。所以也称为类变量。
存储不同	成员变量存在于堆内存中。 静态变量存在于方法区中。
生命周期不同	成员变量随着对象创建而存在。随着对象被回收而消失。 静态变量随着类的加载而存在。随着类的消失而消失。
调用方式不同	成员变量只能被对象所调用 。 静态变量可以被对象调用，也可以被类名调用。 所以，成员变量可以称为对象的特有数据，静态变量称为对象的共享数据。

10. 标识符：由字母、数字、下划线(_)、美元符号($)组成，但不能由数字开头，不能是 Java 关键字

11. 数据类型

8 中基本数据类型 :整型 byte、short、int、long ,浮点型 float、double ,字符型 char，布尔型 boolean

3 中引用类型：class、interface、数组

12. switch 的取值

byte、short、int、char ，JDK5 以后可以是枚举，JDK7 以后可以是字符串。

13. break、continue、return 的区别

    break：用于 switch 和循环语句，用于跳出或结束

    continue：用于循环语句，结束本次循环，继续下一次循环。

    return：结束函数，结束功能。

14. 一个类的实例化过程有哪些步骤？Student s = new Student(); 在内存中到底执行了哪些步骤。

    (1) 类加载器把 Sutdent.class 文件加载进内存

    (2) 在栈内存为 s 变量申请一个空间

    (3) 在堆内存为 Student 对象申请空间

    (4) 对类中的成员变量进行默认初始化

    (5) 对类中的成员变量进行显式初始化

    (6) 有构造代码块就先执行构造代码块，如果没有，则省略

    (7) 执行构造方法，通过构造方法对对象数据进行初始化

    (8) 堆内存中的数据初始化完毕，把内存值复制给 s 变量

15. 内部类

    如果 A 类需要直接访问 B 类中的成员，而 B 类又需要建立 A 类的对象。这时,为了方便设计和访问，直接将 A 类定义在 B 类中就可以了。A 类就称为内部类。内部类可以直接访问外部类中的成员，包括私有的。而外部类想要访问内部类，必须要建立内部类的对象。

16. 匿名内部类

    匿名内部类就是没有名字的内部类，是内部类的简化形式。一般只用一次就可以用这种形式。匿名内部类其实就是一个匿名子类对象。定义匿名内部类的前提：内部类必须继承一个类或者实现接口。

17. 异常

常见异常	
IndexOutOfBoundsException	数组角标越界异常
NullPointerException	空指针异常
ConcurrentModificationException	并发修改异常
ClassCastException	类型转换异常
UnsupportedOperationException	不支持操作异常
NullPointerException	没有元素异常
IllegalArgumentException	非法参数异常
IllegalAccessException	非法的访问异常

18. throw 和 throws 关键字的区别：

(1) throw 用于抛出异常对象，后面跟的是异常对象名，throw 用在方法体内。

(2) throws 用于抛出异常类，后面跟的异常类名，可以跟多个，用逗号隔开。throws 用在方法上。

19. 编译时异常和运行时异常的区别：

(1) Java 编译器会对编译时异常进行检查，如果出现异常就必须对异常进行处理，否则程序无法编译通过。

编译时异常有 2 中处理方式：

使用 try-catch 语句对异常进行捕获

throws 声明抛出异常，让调用者对其进行处理

编译被检查的异常在函数内被抛出，函数必须要声明，否编译失败。

声明的原因：是需要调用者对该异常进行处理。

(2) 运行时异常一般由程序的逻辑错误引起的，在程序运行时无法恢复，所以，不让调用者处理，直接让程序停止运行，由调用者对代码进行修正。

运行时异常如果在函数内被抛出，在函数上不需要声明。

不声明的原因：不需要调用者处理

20. 定义异常处理时，什么时候定义 try，什么时候定义 throws 呢？

功能内部如果出现异常，如果内部可以处理，就用 try；

如果功能内部处理不了，就必须声明出来，让调用者处理。

21. 重写与重载

重写：子类重新实现父类的方法，方法声明相同，方法体不同。与返回值有关

重载：一个类内，函数名相同，参数数据类型，参数个数，排列不同的函数。与返回值无关。

(1) 父类中的私有方法不可以被重写

(2) 子类方法访问权限一定要大于等于父类的访问权限

(3) 静态的方法只能被静态的方法重写

二、多线程

1. 多线程提纲
   (1) 进程、线程、多线程的概念
   (2) 实现多线程的三种方法
   (3) 线程的 5 中状态
   (4) 线程不安全的原因和解决的办法
   (5) 线程间通信机制：等待唤醒机制，wait()、notify()、notifyAll()
   (6) sleep 和 wait 的区别
   (7) 死锁问题
   (8) 单例设计模式
   (9) 同步代码块与同步方法的区别
2. 什么是进程、线程、多线程？
   首先，线程是依赖于进程而存在的。
   进程：正在运行的程序
   线程：进程中一个程序执行控制单元，一条执行路径，是程序使用 CPU 的最基本单位。
   多线程：一个进程有多条执行路径。
   线程：其实就是进程中一个程序执行控制单元，一条执行路径。进程负责的是应用程序的空间的标示。线程负责的是应用程序的执行顺序。

   首先，线程是依赖于进程而存在的。而进程就是操作系统上正在运行的程序；线程是进程中一个程序执行控制单元，一条执行路径，是程序使用 CPU 的最基本单位；如果一个进程有多条执行路径，那么这个程序就是多线程的。

3. 实现多线程的三种方式？
(1) 继承 Thread 类，重写 run()方法，将被线程执行的代码封装在 run()方法中，通过 start()启动线程
(2) 实现 Runnable 接口，重写 run()方法，将 Runnable 接口的实现类对象作为实参传递给 thread 类的构造函数
   好处：将程序代码与数据进行有效分离；避免单继承的局限性。通常使用这种方法。
(3) 实现 Callable 接口，需要和线程池结合使用。
4. 解决线程不安全方法：
   同步代码快：同步锁是任意对象，但必须是同一个对象
   同步方法：同步锁是 this，静态方法的同步锁是类的字节码文件
   Jdk5 以后，可以加 Lock 锁
5. 线程间通信：等待唤醒机制，提供了三个方法：wait()、notify()、notifyAll()
6. 死锁问题：是指两个或者两个以上的线程在执行的过程中，因争夺资源产生的一种互相等待现象
7. 线程不安全的原因：多线程操作共享数据；多条语句操作共享数据。

8. sleep 和 wait 的区别

(1) sleep 是 Thread 类的方法，wait 是 Object 类的方法

(2) sleep 必须指定时间，wait 可以指定时间也可以不指定

(3) sleep 释放 CPU 执行权不释放同步锁，wait 释放 CPU 执行权和同步锁

(4) wait 只能在同步代码块或同步方法里面使用，而 sleep 可以在任何地方使用。

(5) sleep 必需捕获异常，wait 不需要捕获异常

9. 线程状态

新建：创建线程对象

就绪：有执行资格但没有 CPU 执行权

运行：有执行资格和 CPU 执行权

阻塞：sleep()、wait()、等待同步锁

死亡：run()执行完毕或 exception、error

10. 同步方法与同步代码块的区别

(1) 它们的作用都是封装多条操作共享数据的语句，只能让一个线程都执行完，在执行过程中，其他线程不可参与进来。

(2) 同步代码块：位置比较灵活，封装了操作共享数据的语句，多个线程中只有持有锁的才可以操作共享数据，需要指定一个对象作为锁，锁可以是任意对象，但必须是同一对象。

(3) 同步方法：声明方法时加 synchronized 关键字修饰，同步方法使用的锁是 this，静态方法的锁是类的字节码文件，持有锁的线程调用这个方法时其他线程无法调用。

11. 单例设计模式：单例模式就是要确保类在内存中只有一个对象，该实例必须自动创建，并且对外提供。

饿汉式：类一加载就创建对象，实际开发中用这种方法。

懒汉式：用的时候才去创建对象，延迟加载和线程安全问题，getInstance()要声明为同步方法

(1) 保证类在内存中只有一个对象

(2) 构造方法私有化

(3) 在类的内部创建一个该类的实例对象

(4) 通过公共的方法让外界访问

```
public class Single {

    private static Single s = null;
    private Single(){}
    public static Single getInstance(){ //锁是谁？字节码文件对象；
        if(s == null){
            synchronized(Single.class){
                if(s == null)
                    s = new Single();
            }
        }
        return s;
    }


}
```

12. 设计模式：单例设计模式、装饰者模式、适配器模式、工厂模式
13. 线程的默认优先级是多少？范围是多少？

    5 ， 1-10

三、Java 怎么实现跨平台？

1. JDK、JRE、JVM

(1) JDK

Java development kit，Java 开发工具包，jdk 包括 Java 编译器、Java 运行工具、Java 文档生成工具、Java 打包工具，是提供给开发者使用。

(2) JRE

Java Runtime Environment，Java 运行环境，jre 包括 jvm 虚拟机和 Java 核心类库(lib)，是提供给普通用户使用。

(3) JVM

Java Virtual Machine，虚拟机，作用是把类加载到内存，解析执行字节码文件，实现 Java 跨平台性。要运行 Java 程序只需要在操作系统中安装一个对应版本的 Java 虚拟机即可，由 JVM 来调用操作系统底层指令解析、执行 Java 程序，从而让 Java 程序在该操作系统中运行。

Java 语言具有跨平台性，但 jvm 不具有跨平台性。

2. path、classpath

path：命令路径，classpath：class 文件路径；先在当前路径下查找，再到环境变量定义的路径查找。

为了能在 DOS 命令行窗口中，在任何目录下都能够执行 java 相关命令，就需要将 java 命令文件所在目录的路径放入 path 环境变量中。

原理：当在 DOS 命令提示符窗口中输入某个命令后，Windows 系统会首先在当前目录下查找是否存在可以执行的该命令文件。如果没有，Windows 系统就会在 path 环境变量路径中查找。如果查找到，就会执行该命令。如果还没有找到，那么就会提示命令不存在等信息。

3. javac 编译命令： .java 文件编译成 .class 文件

Java 运行命令：执行 .class 文件

四、String

1、String 有哪些特点

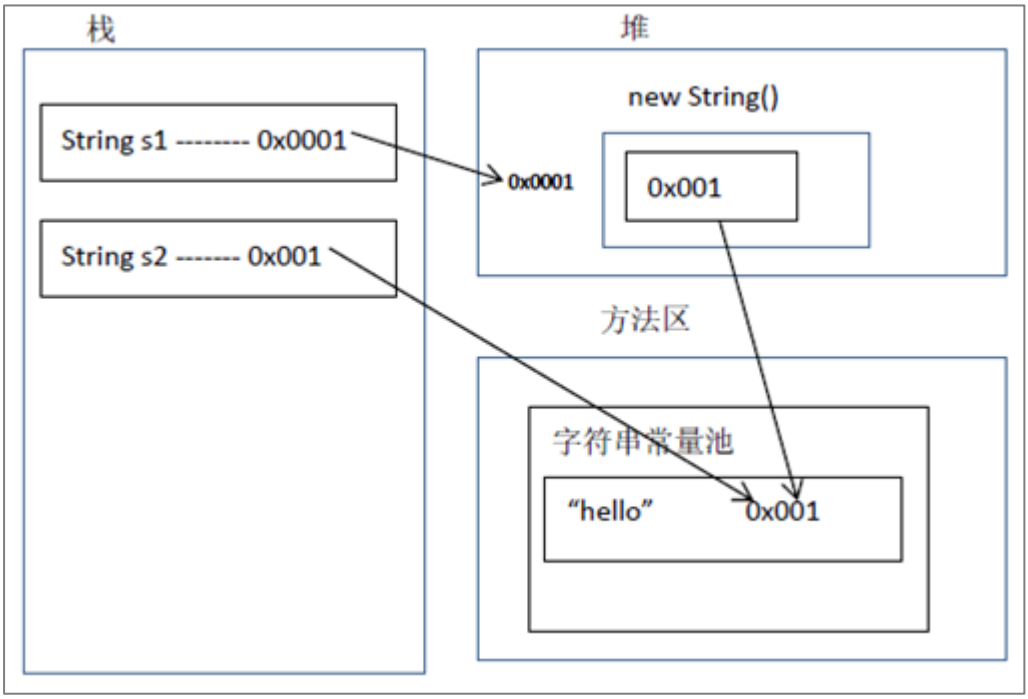
字符串对象一旦被初始化就不会被改变，存放在方法区的字符串常量池中。

2、String s1 = “abc”与 String s2 = new String(“abc”)的区别

s1 创建后，是在字符串常量池中创建了一个"abc"字符串对象。而 s2 是先在堆内存中 new String()对象，然后在方法区的字符串常量池创建了另外一个"abc"字符串对象。

前者创建了 0 或 1 个对象，后者创建了 1 或 2 个对象。

字符串直接赋值是先到方法区的字符串常量池找，有就直接返回，没有就创建并返回



4. 说明 Java 中 String str=null 与 String str=""的区别？

(1) String str = null 表示声明了一个 String 对象的引用 str ,但是没有为其分配内存空间。

(2) String str = "" 表示创建了一个长度等于 0 的空字符串，并在内存中为其分配了内存空间。

(3) String str = new String("tw"); str 在内存中有两个对象 ,1.堆内存有一个 new Sting 2. 方法区的字符串常量池中有一个字符串。

五、集合体系

1、说说集合体系

Collection：单列集合	
List：元素有序，可重复，有索引	Set：无序，唯一
ArrayList 底层数据结构是数组，查询快，增删慢。线程不安全，效率高。数据增长为原来的 50% **Vector** 底层数据结构是数组，查询快，增删慢。线程安全，效率低。数据增长为原来的一倍 **LinkedList** 底层数据结构是链表，查询慢，增删快。线程不安全，效率高	**HashSet** 底层数据结构是哈希表。依赖两个方法：hashCode()和 equals()保证元素的唯一性 **--LinkedHashSet** 底层数据结构是链表和哈希表。由链表保证元素有序，由哈希表保证元素唯一 **TreeSet** 底层数据结构是红黑树(自平衡的二叉树)，由自然排序和比较器排序保证元素的排序；根据比较的返回值是否是 0 来证元素唯一性

Map：双列集合，存储的是键值对形式的元素，键唯一，值可以重复

|--HashMap：键是哈希表结构，可以保证键的唯一性

|--LinkedHashMap :Map 接口的哈希表(保证元素的唯一性)和链表(保证元素的有序性)实现

|--TreeMap：键是自平衡的二叉树结构，可以保证键的排序和唯一性

|--Properties：跟 IO 流结合使用，通常用于读写配置文件

load()：把流中的数据读取到集合中

getProperty()：根据键获取值

setProperty()：添加元素

store()：存储到文件

遍历：

List：普通 for 遍历，增强 for 遍历，迭代器遍历

Set：增强 for 和迭代器遍历

Map：

1. 根据键找值
   (1) 用 keySet()获取所有键的集合
   (2) 遍历键的集合，得到每一个键
   (3) 根据键找值 get()
2. 根据键值对对象找键和值
   (1) 用 entrySet()获取所有的键值对对象的集合
   (2) 遍历键值对对象的集合，得到每一个键值对对象
   (3) 通过键值对对象找键和值，getKey()、getValue ()

Hashtable 和 HashMap 的区别

Hashtable 底层数据结构是哈希表，线程安全，效率低，不允许空键值
HashMap 底层数据结构是哈希表，线程不安全，效率高，允许空键值

3. 哈希表的原理：先用 hashCode()方法比较对象的哈希值是否相等，不等再用 equals()方法比较对象是否相同。
4. Collection 和 Collections 的区别

(1) Collections 是针对集合类操作的一个工具类，提供一系列静态方法，实现对集合的查找、排序、替换、线程安全化等操作。
(2) Collection 是集合类的顶层接口，有 List 和 Set 两个子接口，提供了关于集合的一些通用方法。

5. Map 和 Collection 集合的区别

(1) Map 是双列集合，存储的是键值对形式的元素。
Collection 是单列集合，存储的是单个的元素。
(2) Map 集合的键是唯一的
Collection 的儿子 Set 集合元素是唯一的。
(3) Map 集合的值是可以重复的。
Collection 的儿子 List 集合的元素是可以重复的。

6．泛型是什么?有什么用?在哪里用?泛型有什么好处和弊端？

泛型是一种把明确数据类型的工作推迟到创建对象或者调用方法的时时候才去明确的特殊的数据类型。

好处：

优化了程序的设计，解决了黄色警告线的问题。
把运行时期的问题提前到了编译时期间解决了。
避免了强制类型转换。

弊端：让类型统一了，不能存储不同的数据类型了。

六、网络编程

就是用来实现网络互连的不同计算机上运行的程序间可以进行数据交换。

1. 五层网络模型：应用层 http、网络层 ip、传输层 tcp/udp、物理层、数据链路层
2. 三要素：

   Ip：网络中计算机的唯一标识，点分十进制，IP 地址 = 网络号码+主机地址，5 种

   端口：正在运行的程序的唯一标识，0~2^{16}-1，0~1023 系统占用

   传输协议：通信规则（tcp/ip、udp）：
3. tcp 与 udp 的区别

   udp：不需要建立连接，不可靠协议，传输速度快；发送的数据需要打包；数据包有大小限制，小于 64K

   tcp：需要通过三次握手建立连接，可靠协议，数据没有大小限制，效率稍低。
4. udp：DatagramPacket 数据包、DatagramSocket
5. tcp：发送端 Socket、接收端 ServerSocket

七、反射

1. 什么是反射？

   在运行状态下，通过类的字节码文件解剖一个类，去使用类的构造方法，成员变量，成员方法。

   反射就是把 Java 类中的各种成分映射成相应的 Java 类。

   一般情况下我们要解决某个问题，先找到相关的类，创建该类的对象，然后通过该对象调用对应的方法来解决问题。

   反射是一个正好相反的过程，开始可能并没有类可以解决这个问题，而我们却先用一个当时可能并不存在的方法解决了这个问题，后来才有的这个类。
2. 反射获取字节码对象方式

   获取字节码方式三种：

   (1) 用任意数据类型的静态 class 属性可以得到

   类名.class，例如：System.class

   (2) 用 Object 类的 getClass 方法得到

   对象.getClass()，例如：new Date().getClass();

   (3) 用 Class 类的静态方法 forName()方法得到

   Class.forName("类名")，例如：Class.forName("java.util.Date");

3. 创建对象的两种方式：

   (1)直接用字节码创建对象，只能调用默认的构造方法：字节码.newInstance();

   (2)获取构造方法 Constructor，然后调用构造方法创建，可以通过参数不同调用不同的构造方式

4. 暴力反射

   java 的特性之一就是封装，将对象的属性和具体实现细节隐藏起来，只对外提供公共的方法访问，private 修饰的内部属性和方法对我们是不可见的。

我们通过正常的方法是无法获取以及修改的,可是通过反射却可以强制获取并做一些修改，这就破坏了封装性，这就是所谓的暴力反射

八、IO 流

1. IO 体系，流向分：输入输出流，内容分：字节流、字符流。

3. 字节流输入流 InputStream，读数据

FileInputStream 文件字节流，用于文件的读操作

BufferedInputStream 带缓冲区的字节流，用于提高效率

字节流输出流 OutputStream，写数据

FileOutputStream

BufferedOutputStream

4. 字符流 Reader/Writer

|--FileReader/FileWriter：文件字符流，用于文本文件的读写操作

|--BufferedReader/BufferedWriter：加缓冲区的字符流，用于提高效率

5. 转换流 InputStreamReader/OutputStreamWriter：是字节流和字符流之间的桥梁

6. 配置文件 Properties

super	(1) super.成员变量：父类的成员变量 (1) super.成员方法(形参)：父类的成员方法 (2) super(形参)：父类的构造方法
final	(1) final 修饰的类是最终类，不可以被继承 (2) final 修饰的方法最终方法，不可以重写 (3) final 修饰的变量是常量，只可以赋值一次
this	(1) 代表当前类的对象，谁调用代表谁：哪个对象调用了 this 所在的函数，this 就代表那个对象 (2) this.成员变量：本类的成员变量 (3) this.成员方法(形参)：本类的成员方法 (4) this(形参)：本类的构造方法
static	(1) 随着类的加载而加载；优先于对象存在；被所有对象所共享；可以直接被类名所调用。 (2) 静态方法只能访问静态成员，非静态方法既可以访问静态也可访问非静态成员 (3) 静态方法中不可以定义 this、super 关键字，因为静态优先于对象存在，this 和 super 所代表的当前类对象或父类对象可能还不存在。

1，封装是什么？
2，什么是面向对象，举例说明面向对象思想
3，this 和 super 的区别
4，单例设计模式懒汉式
5，重写和重载的区别
6，异常的分类，运行和编译。。。
7，异常的处理机制　:try...catch...finally
8，throws 和 throw 的区别
9，什么是进程，什么是线程？
10，实现多线程的方法？具体解释一下两种方法
(1)继承 Thread　(2)实现 Runnable 接口；
11，线程的几种状态
12，sleep()和 wait()的区别
13，同步的两种表现形式
14，线程的默认优先级是多少？范围是多少？
15，run()和 start()方法的区别是什么？
16，同步锁对象里面放的是什么？synchronized（锁对象）{//该对象可以是任意对象
17，同步方法，以及同步静态方法锁里面是什么？
18，线程的生命周期图，记清楚
19，字符串什么方法转为数组？toCharArray()
20，StringBuffer 和 StringBuilder 有什么区别？
21，StringBuffer 可不可以添加字符串，
22，StringBuffer 和 String 的区别
23，阐述一下集合框架
24，ArrayList 底层的数据结构是什么?
25,ArrayList 和 LinkedList 和 Vector 之间增删速度，和查询速度如何/?为什么？
26,ArrayList 和 Vector 之间那个安全？
27，Set 和 List 的区别？
28，HashSet 和 TreeSet 如何实现元素唯一的？
30，HashMap 集合的两种遍历方式？
31，HashMap 和 HashTable 的区别？
32，ArrayList 和 HashMap 添加元素的方法是什么？
33，泛型的好处？
34，IO 流分类
35，转换流，字节流向字符是哪个？字符转换成字节是哪种？
36，BufferedReader 的两种特有方法？
37，字符流和字节流的区别？
38，为什么用字符流？

39，反射和暴力反射。
40，文件的复制方法

1. 求从 10 到 100 中能被 3 或 5 整除的数的和

```
int sum = 0;
for(int i = 10; i <= 100; i++) if( i % 3 == 0 || i % 5 == 0) sum += i;
System.out.println(sum);
```

2. 将一个字符串逆序，不要使用反转函数

```
String message = "he saw a racecar";
StringBuilder rev = new StringBuilder();
for(int i = message.length()-1; i >= 0; i--) rev.append(message.charAt(i));
System.out.println(rev.toString());
```

```
    Scanner sc = new Scanner(System.in);
        System.out.println("请输入学生数据：");
        while (sc.hasNext()) {
            String str = sc.nextLine();
            if (str.equals("over")) {
                sc.close();
                break;
            }
    }
```

22. 买票程序

```
1.  public class Test28 {
2.   public static void main(String[] args) {
3.         TicketResource tr = new TicketResource();
4.         TicketWindow tw = new TicketWindow(tr);
5.         new Thread(tw,"窗口 1").start();
6.         new Thread(tw,"窗口 2").start();
7.         new Thread(tw,"窗口 3").start();
8.         new Thread(tw,"窗口 4").start();
9.         new Thread(tw,"窗口 5").start();
10.       }
11. }
12. // 售票窗口类，随机出售 1 到 5 张车票
13. class TicketWindow implements Runnable {
14.  private TicketResource tr;
15.  public TicketWindow(TicketResource tr) {
16.       this.tr = tr;
17.  }
18.  @Override
19.  public void run() {
20.       while (tr.getTickets() > 0) {
21.            tr.setTickets(new Random().nextInt(5) + 1);
22.       }
23.   }
24. }
25. // 车票资源类，封装了车票数，定义了查询余票张数的方法和票数减少的方法
26. class TicketResource {
27.  private int tickets = 100;
28.  public int getTickets() {//获取余票
29.       return tickets;
30.  }
31.  public synchronized void setTickets(int num) {
32.       if (tickets >= num) {
33.            try {
34.                 Thread.sleep(100);
35.            } catch (InterruptedException e) {
36.                 e.printStackTrace();
37.            }
```

```
38.             tickets -=num;
39.             System.out.println(Thread.currentThread().getName() + "卖出去了" +
    num
40.                     + "张票，剩余：" + tickets);
41.         } else {
42.             try {
43.                 Thread.sleep(100);
44.             } catch (InterruptedException e) {
45.                 e.printStackTrace();
46.             }
47.             System.out.println(Thread.currentThread().getName()+"余票不足");
48.         }
49. }
50. }
```

23. 上传文件

(1) 服务器端

```
1.  public class UploadServer {
2.   public static void main(String[] args) throws IOException {
3.         // 创建服务器端的 Socket 对象
4.         ServerSocket ss = new ServerSocket(11111);
5.         // 监听客户端连接
6.         Socket s = ss.accept();//
7.         // 封装通道内的流
8.         BufferedReader br = new BufferedReader(new InputStreamReader(
9.                 s.getInputStream()));
10.        // 封装文本文件
11.        BufferedWriter bw = new BufferedWriter(new FileWriter("Copy.java"));
12.        String line = null;
13.        while ((line = br.readLine()) != null) {
14.            bw.write(line);
15.            bw.newLine();
16.            bw.flush();
17.        }
18.        // 给出反馈
19.        BufferedWriter bwServer = new BufferedWriter(new OutputStreamWriter(
20.                s.getOutputStream()));
21.        bwServer.write("文件上传成功");
22.        bwServer.newLine();
```

```
23.         bwServer.flush();
24.         // 释放资源
25.         bw.close();
26.         s.close();
27.  }
28. }
```

(2) 客户端

```
1.  public class UploadClient {
2.   public static void main(String[] args) throws IOException {
3.         // 创建客户端 Socket 对象
4.         Socket s = new Socket("192.168.12.92", 11111);
5.         // 封装文本文件
6.         BufferedReader br = new BufferedReader(new FileReader(
7.                   "InetAddressDemo.java"));
8.         // 封装通道内流
9.         BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(
10.                  s.getOutputStream()));
11.        String line = null;
12.        while ((line = br.readLine()) != null) {
13.              bw.write(line);
14.              bw.newLine();
15.              bw.flush();
16.        }
17.        //关闭客户端输出流
18.        s.shutdownOutput();
19.        // 接收反馈
20.        BufferedReader brClient = new BufferedReader(new InputStreamReader(
21.                  s.getInputStream()));
22.        String client = brClient.readLine();
23.        System.out.println(client);
24.        // 释放资源
25.        br.close();
26.        s.close();
27.  }
28. }
```

24. 已知文件 a.txt 文件中的内容为“bcdeadferwplkou”，请编写程序读取该文件内容，并按照自然顺序排序后输出到 b.txt 文件中。即 b.txt 中的文件内容应为“abcd……”这样的顺序。

```
1.  public class Test {
2.   public static void main(String[] args) throws IOException {
3.         // 封装文本文件
4.         FileInputStream fis = fis = new FileInputStream("a.txt");
5.         FileOutputStream fos = new FileOutputStream("b.txt");
6.         // 创建了一个关联文件一样大小的缓冲区
7.         byte[] buf = new byte[fis.available()];
8.         // 读取文件
9.         fis.read(buf);
10.        // 排序
11.        Arrays.sort(buf);
12.        // 写入文件
13.        fos.write(buf);
14.        // 刷新缓冲区
15.        fos.flush();
16.        // 释放资源
17.        fos.close();
18.        fis.close();
19.   }
20. }
```

25. 把以下 IP 存入一个 txt 文件，编写程序把这些 IP 按数值大小，从小到达排序并打印出来。

61.54.231.245
61.54.231.9
61.54.231.246
61.54.231.48
61.53.231.249

```
1.  public class Test6 {
2.   public static void main(String[] args) throws Exception {
3.         // 创建字符缓冲输入流对象，读取 IP 数据
4.         BufferedReader buf = new BufferedReader(new FileReader("IP.txt"));
5.         // 创建一个 TreeSet 集合对象，实现比较器排序（升序）
6.         Set<String> set = new TreeSet<String>(new Comparator<String>() {
7.               @Override
8.               public int compare(String ip1, String ip2) {
9.                     // 利用正则表达式将 IP 地址拆分为四段比较
10.                    String[] strArr1 = ip1.split("\\.");
11.                    String[] strArr2 = ip2.split("\\.");
12.                    // 分四段比较 IP 地址的大小
```

```
                for (int i = 0; i < strArr1.length; i++) {
                    // 相等则继续比较下一段的大小
                    if (strArr1[i].equals(strArr2[i])) {
                        continue;
                    }
                    // 不相等,则可以确定顺序
                    return new Integer(strArr1[i]).compareTo(new Integer(
                            strArr2[i]));
                }
                // IP 完全相同,则不加入
                return 0;
            }
        });
        // 每次读取一行数据，即每次读取一个 IP，并添加到 TreeMap 集合中
        String ip = null;
        while ((ip = buf.readLine()) != null) {
            set.add(ip.trim());//去掉两端空格
        }
        // 释放资源
        buf.close();
        // 增强 for 打印排序后的 IP 地址
        for (String str : set) {
            System.out.println(str);
        }
    }
}
```

26. 复制指定目录下的指定文件，并修改后缀名。

```
1.  public class CopyFolderDemo {
2.   public static void main(String[] args) throws IOException {
3.       // 封装目录
4.       File srcFolder = new File("e:\\java");
5.       // 封装目的地
6.       File destFolder = new File("e:\\jad");
7.       // 如果目的地目录不存在，就创建
8.       if (!destFolder.exists()) {
9.           destFolder.mkdir();
10.      }
11.
```

```
12.       // 获取该目录下的 java 文件的 File 数组
13.       File[] fileArray = srcFolder.listFiles(new FilenameFilter() {
14.            @Override
15.            public boolean accept(File dir, String name) {
16.                 return new File(dir, name).isFile() && name.endsWith(".java");
17.            }
18.       });
19. // 遍历该 File 数组，得到每一个 File 对象
20.       for (File file : fileArray) {
21.            String name = file.getName();
22.            File newFile = new File(destFolder, name);
23.            copyFile(file, newFile);
24.       }
25.
26.       // 在目的地目录下改名
27.       File[] destFileArray = destFolder.listFiles();
28.       for (File destFile : destFileArray) {
29.            String name =destFile.getName();
30.            String newName = name.replace(".java", ".jad");
31.            File newFile = new File(destFolder,newName);
32.            destFile.renameTo(newFile);
33.       }
34. }
35.
36.  private static void copyFile(File file, File newFile) throws IOException {
37.       BufferedInputStream bis = new BufferedInputStream(new
     FileInputStream(
38.                 file));
39.       BufferedOutputStream bos = new BufferedOutputStream(
40.                 new FileOutputStream(newFile));
41.       byte[] bys = new byte[1024];
42.       int len = 0;
43.       while ((len = bis.read(bys)) != -1) {
44.            bos.write(bys, 0, len);
45.       }
46.       bos.close();
47.       bis.close();
48. }
```

```
49. }
50.
```

27. txt 文件中有这样的一个字符串：“hcexfgijkamdnoqrzstuvwybpl”，请编写程序读取数据内容，把数据排序后写入 b.txt 中。

分析：

A:把 a.txt 这个文件给做出来

B:读取该文件的内容，存储到一个字符串中

C:把字符串转换为字符数组

D:对字符数组进行排序

E:把排序后的字符数组转换为字符串

F:把字符串再次写入 b.txt 中

```
1.  public class StringDemo {
2.   public static void main(String[] args) throws IOException {
3.       // 读取该文件的内容，存储到一个字符串中
4.       BufferedReader br = new BufferedReader(new FileReader("a.txt"));
5.       String line = br.readLine();
6.       br.close();
7.       // 把字符串转换为字符数组
8.       char[] chs = line.toCharArray();
9.       // 对字符数组进行排序
10.      Arrays.sort(chs);
11.      // 把排序后的字符数组转换为字符串
12.      String s = new String(chs);
13.      // 把字符串再次写入 b.txt 中
14.      BufferedWriter bw = new BufferedWriter(new FileWriter("b.txt"));
15.      bw.write(s);
16.      bw.newLine();
17.      bw.flush();
18.      bw.close();
19. }
20. }
21.
```

28. 复制多极文件夹

分析：

A:封装数据源 File

B:封装目的地 File

C:判断该 File 是文件夹还是文件

a:是文件夹

就在目的地目录下创建该文件夹

获取该 File 对象下的所有文件或者文件夹 File 对象
遍历得到每一个 File 对象
回到 C
b:是文件
就复制(字节流)

```
1.  public class CopyFoldersDemo {
2.   public static void main(String[] args) throws IOException {
3.       // 封装数据源 File
4.       File srcFile = new File("E:\\JavaSE\\day21\\code\\demos");
5.       // 封装目的地 File
6.       File destFile = new File("E:\\");
7.       // 复制文件夹的功能
8.       copyFolder(srcFile, destFile);
9.   }
10.  private static void copyFolder(File srcFile, File destFile)
11.          throws IOException {
12.      // 判断该 File 是文件夹还是文件
13.      if (srcFile.isDirectory()) {
14.          // 文件夹
15.          File newFolder = new File(destFile, srcFile.getName());
16.          newFolder.mkdir();
17.          // 获取该 File 对象下的所有文件或者文件夹 File 对象
18.          File[] fileArray = srcFile.listFiles();
19.          for (File file : fileArray) {
20.              copyFolder(file, newFolder);
21.          }
22.      } else {
23.          // 文件
24.          File newFile = new File(destFile, srcFile.getName());
25.          copyFile(srcFile, newFile);
26.      }
27.  }
28.  private static void copyFile(File srcFile, File newFile) throws IOException {
29.      BufferedInputStream bis = new BufferedInputStream(new
    FileInputStream(
30.              srcFile));
31.      BufferedOutputStream bos = new BufferedOutputStream(
32.              new FileOutputStream(newFile));
```

```
33.         byte[] bys = new byte[1024];
34.         int len = 0;
35.         while ((len = bis.read(bys)) != -1) {
36.             bos.write(bys, 0, len);
37.         }
38.         bos.close();
39.         bis.close();
40.     }
41. }
```