不常用但是很基础容易忘记的知识点:

小数默认是double类型,不能直接赋值给float
布尔型不能直接用0和1赋值
long型的数字后加l或L
二进制0b开头,八进制0开头,十六进制0x开头
float类型必须写f或F
e2表示十的二次方
final修饰的变量只能赋值一次,可以在声明的时候赋值,也可以在构造器中赋值.
if else if语句一旦前面条件成立,后面跳过判断语句
多层循环,只能用break跳出一层循环,可以在最内层break前面把true存入boolean变量,再次外层循环下面通过条件语句,再跳出次循环.
标签结束外部循环,最外层循环的上方 标签名: 循环内通过break 标签名 的格式跳出循环
所以如果指定了数组的内容，就不能同时设置数组的长度
int[] c = new int[3]{100,102,444,836,3236};
System.arraycopy(src, srcPos, dest, destPos, length) 原数组,起始位置,目的数组,起始位置,复制长度
copyOfRange 数组复制
toString() 转换为字符串
sort 排序
binarySearch 搜索
equals 判断是否相同
fill 填充

new Hero();
代表创建了一个Hero对象
但是也仅仅是创建了一个对象，没有办法访问它
为了访问这个对象，会使用引用来代表这个对象
Hero h = new Hero();
h这个变量是Hero类型，又叫做引用
=的意思指的h这个引用代表右侧创建的对象
“代表” 在面向对象里，又叫做“指向
引用指向新对象: Hero garen = new Hero(); garen = new Hero();
方法名是一样的，但是参数类型不一样,方法重载
构造器中用this调用其他构造器,但是必须是this放在第一个语句
使用其他包下的类用import
作用范围最小原则:能用private就用private，不行就放大一级，用package,再不行就用protected，最后用public。这样就能把数据尽量的封装起来，没有必要露出来的，就不用露出来了

对象.类属性和类.类属性都可以访问Static修饰的类变量
类方法: 又叫做静态方法 不用建立对象,可以直接访问,两种访问方式和类属性一样,如果一个方法，没有调用任何对象属性，那么就可以考虑设计为类方法,对象属性就是没有static修饰的属性

对象方法： 又叫实例方法，非静态方法

对象属性初始化有3种
1. 声明该属性的时候初始化
2. 构造方法中初始化
3. 初始化块
类属性初始化有2种
1. 声明该属性的时候初始化
2. 静态初始化块

单例方法一饿汉式:1.私有化构造方法,2.准备一个类属性,指向该类的实例,3.静态方法,返回类实例,4.主函数新建引用指向类.类方法
单例方法二懒汉式:1.私有化构造方法,2.准备一个类属性,指向NULL,3.静态方法,if语句判断类属性是否为null,是则指向类的一个实例,然后返回该类属性,4.主函数新建引用指向类.类方法
饿汉式是立即加载的方式，无论是否会用到这个对象，都会加载。
如果在构造方法里写了性能消耗较大，占时较久的代码，比如建立与数据库的连接，那么就会在启动的时候感觉稍微有些卡顿。
这个是面试的时候经常会考的点，面试题通常的问法是:

面试考点:
什么是单例模式？回答的时候，要答到三元素
1. 构造方法私有化
2. 静态属性指向实例
3. public static的 getInstance方法，返回第二步的静态属性

枚举是特殊的类,也是类.

```
public enum Season {
    SPRING,SUMMER,AUTUMN,WINTER
}
public class HelloWorld {
    public static void main(String[] args) {
        Season season = Season.SPRING;
        switch (season) {
        case SPRING:
            System.out.println("春天");
            break;
        case SUMMER:
            System.out.println("夏天");
            break;
        case AUTUMN:
            System.out.println("秋天");
            break;
        case WINTER:
```

```
            System.out.println("冬天");
            break;
        }
    }
}
```

枚举类遍历:

```
public class HelloWorld {
    public static void main(String[] args) {
        for (Season s : Season.values()) {
            System.out.println(s);
        }
    }
}
```

接口里面的方法默认省去了public abstract

对象是有类型的,类型和构造该对象的构造方法名一样.引用也是有类型的,一般来说引用的类型和对象的类型一样

```
Hero h = new Hero();
ADHero ad = new ADHero();
h = ad;
```

右边ad引用所指向的对象的类型是 物理攻击英雄

左边h引用的类型是 普通英雄

把物理攻击英雄 当做 普通英雄，说不说得通？ 说得通，就可以转

父类引用可以指向子类对象(向上转型)

没有继承关系的两个类，互相转换，一定会失败

虽然ADHero和APHero都继承了Hero，但是彼此没有互相继承关系

实现类向上转型为接口,可以成功

instanceof Hero 判断一个引用所指向的对象，是否是Hero类型，或者Hero的子类

```
package charactor;

public class Hero {
    public String name;
    protected float hp;

    public static void main(String[] args) {
        ADHero ad = new ADHero();
        APHero ap = new APHero();

        Hero h1= ad;
        Hero h2= ap;

        //判断引用h1指向的对象，是否是ADHero类型
```

```
        System.out.println(h1 instanceof ADHero);

        //判断引用h2指向的对象，是否是APHero类型
        System.out.println(h2 instanceof APHero);

        //判断引用h1指向的对象，是否是Hero的子类型
        System.out.println(h1 instanceof Hero);
    }
}
```

要实现类的多态，需要如下条件

1. 父类（接口）引用指向子类对象
2. 调用的方法有重写

final 修饰类,不能被继承;修饰方法,子类方法不能被重写;修饰引用,不能修改引用指向的对象,修饰变量,只能赋值一次

abstract 抽象类,可以定义抽象方法(也可以没有抽象方法,但是定义了抽象类,就不能直接实例化),不同的子类可以实现不同的抽象方法

接口和抽象类的区别:

区别1:

子类只能继承一个抽象类，不能继承多个

子类可以实现多个接口

区别2:

抽象类可以定义

public,protected,package,private

静态和非静态属性

final和非final属性

但是接口中声明的属性，只能是

public

静态

final的

即便没有显式的声明

非静态内部类，只有一个外部类对象存在的时候，才有意义,因为没有一个外部类的实例，所以在静态内部类里面不可以访问外部类的实例属性和方法

除了可以访问外部类的私有静态成员外，静态内部类和普通类没什么大的区别 new 外部类.静态内部类();

匿名类指的是在声明一个类的同时实例化它，使代码更加简洁精练

通常情况下，要使用一个接口或者抽象类，都必须创建一个子类,在匿名类中使用外部的局部变量，外部的局部变量必须修饰为final

int的最大值可以通过其对应的封装类Integer.MAX_VALUE获取

不需要调用Integer的intValue方法，通过=就自动转换成int类型，就叫拆箱 自动拆箱

数字转字符串:

方法1： 使用String类的静态方法valueOf

方法2：先把基本类型装箱为对象，然后调用对象的toString

```
//方法1
String str = String.valueOf(i);

//方法2
Integer it = i;
String str2 = it.toString();
```

Math.round 四舍五入
Math.random 随机数0~1
Math.sqrt;开方
Math.pow(a,b);a的b次方
Math.Pi 圆周率π
Math.E 自然常数
格式化输出,和C语言一样 printf format
创建字符串,1.new String("");2.new String(字符数组);3.两个字符或字符串用"+"连接;
String类不能被继承 final修饰
immutable 是指不可改变的
比如创建了一个字符串对象
String garen ="盖伦";
不可改变的具体含义是指:
不能增加长度
不能减少长度
不能插入字符
不能删除字符
不能修改字符
一旦创建好这个字符串，里面的内容 永远 不能改变

String 的表现就像是一个常量
字符串一旦创建,就不能改变内容,只能修改其引用指向新的字符串;
字符串格式化(不熟)

charAt　获取字符
toCharArray　获取对应的字符数组
subString 截取子字符串
split 分隔
trim 去掉首尾空格

toLowerCase
toUpperCase 大小写

indexOf
lastIndexOf

contains 定位

replaceAll
replaceFirst 替换
比较字符串是否是同一个对象(不同引用指向的是否同一个对象),用==
String str1 = "the light";
String str2 = "the light";
因为编译器发现已经存在现成的"the light",所以不会再创建新的字符串,两个引用指向的是一个对象
判断两个字符串的内容是否相同,用equals方法
StringBuffer是可变长的字符串,通过调用方法实现增删改差

1.将可能抛出FileNotFoundException 文件不存在异常的代码放在try里
2.如果文件存在，就会顺序往下执行，并且不执行catch块中的代码
3. 如果文件不存在，try 里的代码会立即终止，程序流程会运行到对应的catch块中
4. e.printStackTrace(); 会打印出方法的调用痕迹，如此例，会打印出异常开始于TestException的第16行，这样就便于定位和分析到底哪里出了异常
FileNotFoundException是Exception的子类，使用Exception也可以catch住FileNotFoundException

```
try {
    System.out.println("试图打开 d:/LOL.exe");
    new FileInputStream(f);
    System.out.println("成功打开");
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    Date d = sdf.parse("2016-06-03");
} catch (FileNotFoundException e) {
    System.out.println("d:/LOL.exe不存在");
    e.printStackTrace();
} catch (ParseException e) {
    System.out.println("日期格式解析错误");
    e.printStackTrace();
}

        try {
    System.out.println("试图打开 d:/LOL.exe");
    new FileInputStream(f);
    System.out.println("成功打开");
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    Date d = sdf.parse("2016-06-03");
} catch (FileNotFoundException | ParseException e) {
    if (e instanceof FileNotFoundException)
        System.out.println("d:/LOL.exe不存在");
    if (e instanceof ParseException)
```

```
            System.out.println("日期格式解析错误");

        e.printStackTrace();
    }
```

无论是否出现异常，finally中的代码都会被执行

```
    finally{
        System.out.println("无论文件是否存在， 都会执行的代码");
    }
```

考虑如下情况：

主方法调用method1
method1调用method2
method2中打开文件

method2中需要进行异常处理
但是method2不打算处理，而是把这个异常通过throws抛出去
那么method1就会接到该异常。 处理办法也是两种，要么是try catch处理掉，要么也是抛出去。
method1选择本地try catch住 一旦try catch住了，就相当于把这个异常消化掉了，主方法在调用method1的时候，就不需要进行异常处理了

```
    try {
        new FileInputStream(f);
        //使用Throwable进行异常捕捉
    } catch (Throwable t) {
        // TODO Auto-generated catch block
        t.printStackTrace();
    }
```

f1.getAbsolutePath() 绝对路径

注意1： 需要在D:\LOLFolder确实存在一个LOL.exe,才可以看到对应的文件长度、修改时间等信息

注意2： renameTo方法用于对物理文件名称进行修改，但是并不会修改File对象的name属性。

```
    File f = new File("d:/LOLFolder/LOL.exe");
    //文件是否存在
    System.out.println("判断是否存在："+f.exists());

    //是否是文件夹
    System.out.println("判断是否是文件夹："+f.isDirectory());

    //是否是文件（非文件夹）
```