冒泡排序：

public class Test3 {  
 public static void main(String[] args) {  
 int[] arr = {23,45,12,46,56,123,45,32,76,79,56};  
 int temp;  
 for(int i=0;i<arr.length-1;i++){  
 for(int j=0;j<arr.length-i-1;j++){  
 if(arr[j]>arr[j+1]){  
 temp = arr[j];  
 arr[j] = arr[j+1];  
 arr[j+1] = temp;  
 }  
 }  
 }  
 for(int i : arr){  
 System.*out*.print(i + " ");  
 }  
 }  
}

饿汉模式：

package compare;  
  
public class Hungry {  
 private static Hungry *demo* = new Hungry();  
 private Hungry(){  
  
 }  
 public static Hungry getDemo() {  
 return *demo*;  
 }  
  
}

懒汉模式：

package compare;  
  
public class Lazy {  
 private static volatile Lazy *demo*;  
 private Lazy(){  
  
 }  
 public static Lazy getDemo(){  
 if(*demo*==null){  
 synchronized (Lazy.class){  
 if(*demo*==null){  
 *demo* = new Lazy();  
 }  
 }  
 }  
 return *demo*;  
 }  
}

优化之静态内部类：

package compare;  
  
public class inner {  
 private inner(){  
  
 }  
 private static class Demo{  
 private static inner *demo* = new inner();  
 }  
 public static inner getDemo(){  
 return Demo.*demo*;  
 }  
}

底层模块不能依赖于高层的，否则耦合度会很高

public String substring(int beginIndex, int endIndex)

返回一个新字符串，它是此字符串的一个子字符串。该子[字符串](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E4%B8%B2)从指定的 beginIndex 处开始， endIndex:到指定的 endIndex-1处结束。

参数：beginIndex - 开始处的索引（包括）

    　　endindex 结尾处索引（不包括）。

Number类:

Interger.parseInt(“字符串”,”多少进制”)

枚举类型的遍历

public class Test5 {  
 enum food{*rice*,*chicken*,*drinking*}  
 public static void main(String[] args) {  
 for(food i : food.*values*()){  
 System.*out*.println(i.toString());  
 }  
 }  
}

转义字符;

“/n”换行

“/t”tab

按占字节大小排序八种基本数据类型

1个字节Boolean byte 2个short char 4个int float 8个long doule

命名规范：

项目名全部小写

类名每个单词首字母大写

变量方法第一个单词首字母小写，后面单词的首字母都大写

注释规范：

类注释：

/\*\*

\* Copyright (C), 2006-2010, ChengDu Lovo info. Co., Ltd.

\* FileName: Test.java

\* 类的详细说明

\*

\* @author 类创建者姓名

\* @Date 创建日期

\* @version 1.00

\*/

属性注释：

/\*\* 提示信息 \*/

private String strMsg = null;

方法注释：

/\*\*

\* 类方法的详细使用说明

\*

\* @param 参数1 参数1的使用说明

\* @return 返回结果的说明

\* @throws 异常类型.错误代码 注明从此类方法中抛出异常的说明

\*/

向上转型是自动，不用强转

public class Test6 {  
 public static void main(String[] args) {  
 Fruit fruit = new Apple();  
 ((Apple) fruit).appearance();  
 }  
}

fruit想要引用子类中的appearance方法，必须要向下强转成apple类型

关于protected修饰符：

* 基类的 protected 成员是包内可见的，并且对子类可见；
* 若子类与基类不在同一包中，那么在子类中，子类实例可以访问其从基类继承而来的protected方法，而不能访问基类实例的protected方法。
* 父类中声明为 public 的方法在子类中也必须为 public。
* 父类中声明为 protected 的方法在子类中要么声明为 protected，要么声明为 public，不能声明为 private。
* 父类中声明为 private 的方法，不能够被继承。

Vehicle v2 = new Car();    // v2 是 Car 类型

int a=2;

System.out.println(--a/2+(++a\*2)); //4

System.out.println(++a\*2+--a/2); //7

直接插入排序：将数组的元素和前面已经排好的元素对比

希尔排序，按照步长去对比，第二次步长为一半，直到步长为1

简单选择排序：依次找到最小的

快速排序，选一个基准然后去对比

Date dNow = new Date( ); SimpleDateFormat ft = new SimpleDateFormat ("yyyy-MM-dd hh:mm:ss"); System.out.println("当前时间为: " + ft.format(dNow));