**redis**

查看集合大小：

Scard key 如果set是空或者key不存在，返回0

创建**Git**版本库：

git init

Git 相关命令：git status 查看结果

执行两步提交到仓库：

1. git add XXX(文件名)
2. git commit -m “explain”

版本回退： git log 查看历史修改

git checkout -- file可以丢弃工作区的修改

要设置用户名以及邮箱，否则公钥私钥不匹配，

git remote add origin [git@github.com:](mailto:git@github.com:) 路径.git

然后，本地库的所有内容远程推送到Github:（若更改，需要先commit）

git push -u origin master //远程库初始是空的就需要加-u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令以后就不用了

Eclipse下使用PyDev搭建Python 开发环境：<http://www.cnblogs.com/wangcp-2014/p/5531130.html>

HashMap中threshold=capacity\*loadFactor ---故一般情况下threshold 会更小一些。当HashMap的size大于threshold时会执行resize()操作;

常规情况来讲就是: size<threshold<capacity

讨论容器的类型，而不是容器持有的类型。

**class** Food{}

**class** Fruit **extends** Food{}

**class** Apple **extends** Fruit{}

//定了上界，但不能维持list中是同样的类型，如 Apple、与orange 虽然都继承于Fruit 但是是不同的类，写入会失败

如 orange 是继承于 Fruit ，你填入一个 Apple()，是错误的，即使是可以的，但编译器是没有那么智能

List<? extends Fruit> flist=new ArrayList<>();

//!- Flist.add(new Apple()); Apple is Fruit or it’s childern (orange)?Xerror

//!- Flist.add(new Fruit()); Fruit is Fruit or it’s children (orange) ?X error

//!- Flist.add(new Object()); error

flist.add(**null**);// 可以但无意义~

List<? **extends** Fruit>flist=Arrays.*asList*(**new** Apple());

//!-List<? **extends** Fruit>flist=Arrays.*asList*(**new** Food());

List<? **super** Fruit>flist=Arrays.*asList*(**new** Apple()); 也可以

这样却可以

//定了下界，中都是Fruit的超类，存入Fruit总没错。

List<? **super** Fruit>flist=**new** ArrayList<>();

flist.add(**new** Apple());//ok-- Apple is Fruit or it’s parents

flist.add(**new** Fruit());//ok --Fruit is Fruit

//!-flist.add(**new** Food());//可能会有其他类转型为Food() 作为参数传入，-- Food is Fruit -error

--20170802

hashMap的modCount: 记录结构的变化次数，value更改不作数,添加则会++；

1.8中扩容（resize()）的一个改进(?)是扩容之后不用重新再计算key在hash桶中的位置了，（扩容为原来的2倍 capacity\*2），要么是原始位置index，要么是index= oldCap + index,

这两个不要弄混了：

---计算桶中的位置是index&(len-1)

---计算hashCode的值是：(h=k.hashCode())^(h>>>16) 保证高低位都参与到Hash的计算中，

<http://blog.csdn.net/u011240877/article/category/6447444>讲的挺详细的。

HashMap允许key，Value为null，同时他们都保存在第一个桶中,可查看源码得:

Put()方法： return putVal(hash(key),key,value,false,true)

而hash()方法：

return (key==null) ? 0: (h=key.hashCode())^(h>>>16)

**为什么哈希表的容量一定是2的整数次幂？**--很有道理啊

1、capacity为2的整数次幂的话， h& (len-1) 就相当于对len去模，提升了计算效率，

1. capacity为2的整数次幂，为偶数，因此capacity-1为奇数，可以保证h&(len-1)最后一位可能为1也可能为0， 可以保证散列的均匀性，

而如果 capacity 为奇数的话，很明显 capacity-1 为偶数，它的最后一位是 0，这样 h&(capacity-1) 的最后一位肯定为 0，即只能为偶数，这样任何 hash 值都只会被散列到数组的偶数下标位置上，这便浪费了近一半的空间。

---20170803