Map接口：

Map集合是无序的，每条数据都包含两个信息：Key-Value；

Key：唯一不重复（equals比较为true视为重复）

Value：是可以重复的值

Map接口和Collection是没有交际的，Map接口是独立的接口

Put(K,V)：添加元素

1. 当添加的元素k-v在集合不存时，为添加元素
2. 当添加的元素k-v在集合中存在时，则会根据k值替换与之相等的v
3. 返回的值，为map集合中与key相等的，value值，没有为null

Get(K): equals比较为true视为相同的key该方法根据给定的K获取对应的v值，返回值为对应泛型的v值，如果不存在返回null

Remove(K)：根据给定的K，来删除这一行数据，返回值为被删除的K所对应的v值，不存在返回null

重写equals方法时应当重写hashCode(),重写规则如下：

1. 若两个对象equals方法比较为true，那么hashCode()返回的值必须相同
2. 反之不是必须的，但是这种情况要避免：即两个对象equals比较为false，但是hashCode()返回值一样，这种情况要避免，如果hashCode()返回值一样，则会影响HashMap的性能
3. 注意：hashCode()在当前内容没有发生改变的前提下，多次调用应当是返回相同的值，如果内容修改后，则应当返回不同的值

加载因子，0.75: 平衡检索和空间利用率，没有初始化散列组大小时默认值为16，可以根据数据大小发生变化，保证加载因子为0.75

Map集合遍历：

1. 遍历所有的Key
   1. Set<K> keyset()：可以获取Map集合中所有的Key值并将其存入一个Set集合中返回，遍历该Set集合即可
2. 遍历所有的Key及Value
   1. Set<Entry<K,V>> entrySet(): 讲每一组K-v存入一个Entry<k,v>实例中，并返回Set集合，然后遍历该Set集合即可
3. 遍历所有的Value
   1. Collect<T> Values(): 返回的值时对应的value泛型的值，存入Collection<T>集合，然后迭代器遍历即可

File：文件和目录路径的抽象表现形式，只能获取文件的周边信息

1. 不能对文件内容进行读写操作
2. 想要操作一个文件，需要现有一个文件的对象
3. 跨平台：一次编译到处运行，处处改错，文件分割符号，各个平台表现形式不一样
   1. Windows：\
   2. Linux：/
   3. 为了解决这种跨平台引起的非逻辑性的一场，引入以常量表示文件的分隔符：
   4. Separator：这个常量可以在各个平台上，根据各个平台的属性，自动使用文件分隔符的表现形式
   5. 文件分隔符：separaotr（常量），--忽略平台的差异性
   6. 属于文件类中的常量
   7. 语法：File.separator调用
4. 路径：
   1. 绝对路径：全路径
   2. 相对路径：有一个参照物，参照物是当前的项目，相对项目名
   3. ./ 代表是当前目录
   4. ../ 代表的是当前目录的上一级目录（父级目录）
5. 获取文件的周边信息的方法
   1. getName()获取文件或者目录的名字，返回String字符串
   2. 删除目录，只能删除空目录