**import** org.apache.spark.rdd.RDD  
**import** org.apache.spark.{SparkConf, SparkContext}  
  
**import** scala.collection.Map  
  
*/\*\*  
 \* Created by Administrator on 2017/2/19 0019.  
 \*/***case class** Student(name:String,age:Int) **extends** Ordered[Student] {  
 **override def** compare(that: Student): Int = {  
 **if** (**this**.name == that.name) -1  
 **else** 1  
 }  
}  
  
  
**object** OpRdd {  
 **def** main(args: Array[String]): Unit = {  
 **val** conf = **new** SparkConf().setAppName(**"OpRdd"**).setMaster(**"local"**)  
 **val** sc = **new** SparkContext(conf)  
  
 *//RDD的创建* **val** nums = *Array*(1,2,3,4)  
 **val** numRdd: RDD[Int] = sc.parallelize(nums)  
  
 **val** words = *Array*(**"Hello Kevin"**,**"I'm Jhon"**,**"Nice meeting you!"**)  
 **val** wordRdd: RDD[String] = sc.parallelize(words)  
  
 **val** scores = *Array*((**"Kevin"**,60),(**"Lily"**,70),(**"Lily"**,80),(**"Kevin"**,90))  
 **val** scoreRdd: RDD[(String, Int)] = sc.parallelize(scores)  
  
  
 *//第一部分 常见的transformations, lazy特性  
  
 //map 1.不改变元素类型  
// val mapRdd: RDD[Int] = numRdd.map(x => x \* 2)  
 // 2.改变元素类型* **val** mapRdd: RDD[(Int, Int)] = numRdd.map(x => (x,1))  
  
  
 *//flatMap 不改变元素类型，改变元素数量* **val** flatMapRdd: RDD[String] = wordRdd.flatMap(str => str.split(**" "**))  
  
 *//filter 过滤含义m字符的单词* **val** filterRdd: RDD[String] = wordRdd.filter( str => !str.contains(**"m"**) )  
  
 *//对 (key,value)对RDD的操作  
 //groupByKey 对元素进行分组，返回的是(key,seq[])* **val** groupByKeyRdd: RDD[(String, Iterable[Int])] = scoreRdd.groupByKey()  
  
 *//reduceByKey ,通过key对value进行聚合, v1+ v2 = v,接下来是v + v3 =* **val** reduceByKeyRdd: RDD[(String, Int)] = scoreRdd.reduceByKey(\_ + \_)  
  
 *//和reduceByKey操作相同的效果，但是groupByKey性能比较低，所有数据要加载到内存处理，数据量大的时候内存占用高  
 //reduceByKey性能比较高，会在map先做聚合，类似于combine,减少了reduce的数量，也减少网络开销等  
 //如果reduceByKey处理不了，才会选择groupByKey().map()代替  
// scoreRdd.groupByKey().map(x => (x.\_1,x.\_2.sum)).foreach(println(\_))  
  
 //sortByKey 按照key进行排序* **val** sortByKeyRdd: RDD[(String, Int)] = scoreRdd.sortByKey(**false**)  
  
 **val** students = *Array*(**new** Student(**"Kevin"**,22),  
 **new** Student(**"Lily"**,20))  
  
 **val** studentRdd: RDD[Student] = sc.parallelize(students)  
  
 *//元组对* **val** map: RDD[(Student, Int)] = studentRdd.map(x => (x,1))  
 map.sortByKey().foreach(*println*(\_))  
  
 **val** info = *Array*((**"Kevin"**,22),(**"Lily"**,20))  
 **val** infoRdd: RDD[(String, Int)] = sc.parallelize(info)  
  
 *//join 按照key进行关联,笛卡尔积关联* **val** joinRdd: RDD[(String, (Int, Int))] = infoRdd.join(scoreRdd)  
*// infoRdd.join(scoreRdd.groupByKey()).foreach(println)  
  
 //cogroup  
// val cogroupRdd: RDD[(String, (Iterable[Int], Iterable[Int]))] = infoRdd.cogroup(scoreRdd)* **val** address = *Array*((**"Kevin"**,**"gz"**),(**"Lily"**,**"sz"**))  
 **val** addressRdd: RDD[(String, String)] = sc.parallelize(address)  
  
 **val** cogroupRdd: RDD[(String, (Iterable[Int], Iterable[Int], Iterable[String]))] = infoRdd.cogroup(scoreRdd,addressRdd)  
  
  
 *//第二部分： actions操作  
// wordRdd.foreach(println)  
// println("-------------------------------")  
// cogroupRdd.foreach(println(\_))  
  
  
 //reduce* **val** reduce: Int = numRdd.reduce(\_ + \_)  
*// println(reduce)  
  
  
 //collect和foreache区别  
 // collect将所有参与计算的节点数据拉取到终端，如果数据量大的时候性能比较低  
 //foreach 是参与计算节点的数据自己遍历输出，比较高效* **val** collect: Array[Int] = numRdd.collect()  
*// for(ele <- collect) println(ele)  
// numRdd.foreach(println(\_))  
  
 // take 类似与TopN操作* **val** take: Array[Int] = numRdd.take(3)  
 **for**(i <- 0 until take.length) *println*(take(i))  
  
 *//first 类似take(1)* **val** first: Int = numRdd.first()  
*// println(first)  
// println(numRdd.take(1)(0))  
  
 //count* **val** count: Long = numRdd.count()  
 *println*(count)  
  
  
 *//countByKey* **val** countByKeyRdd: Map[String, Long] = scoreRdd.countByKey()  
 **for**(ele <- countByKeyRdd){  
 *println*(ele)  
 }  
 *//比countByKey高效* **val** reduceByKey: RDD[(String, Int)] = scoreRdd.mapValues(\_ => 1).reduceByKey(\_ + \_)  
 reduceByKey.foreach(*println*(\_))  
  
  
  
 *//saveAsTextFile,一般是hdfs路径，或者linux本地路径，不是windows  
// numRdd.saveAsTextFile("file:\\F:\\OpRdd")* }  
}