package function2;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import scala.Tuple2;

public class Cantime100 implements Serializable{

private final int timeInterval;

private final int preprocessingFlag;

private static final long serialVersionUID = - 2022345678L;

public Cantime100(int timeInterval, int preprocessingFlag) {

this.timeInterval = timeInterval;

this.preprocessingFlag = preprocessingFlag;

}

//参考去年写的代码，传入传出参数是游标(Iterator)类型

//去年代码里重载函数名是(call)

@Override

public Iterator<Tuple2<TelegramHash, CanUnitBean>> execute(

Iterator<Tuple2<TelegramHash, CanUnitBean>> its) throws Exception {

if (preprocessingFlag == 0) {

return tuple;

}

//HashMap<deviceId, TreeMap<时间，数据包>>

//java的TreeMap结构的遍历是有序的(\*重要\*)

HashMap<String, TreeMap<Integer, CanUnitBean>> Telegram = new HashMap<String, TreeMap<Integer, CanUnitBean>>();

//设备混杂,时间无序的数据包 => 设备分类，时间排序的结构 --start

while (its.hasNext()) {

Tuple2<TelegramHash, CanUnitBean> next = its.next();

TreeMap<Integer, CanUnitBean> rvTelegram = Telegram.get(next.\_1.deviceId);

//设备对应TreeMap新建

if(rvTelegram == null) {

Timestamp.put(next.\_1.deviceId, next.\_1.timestamp);

rvTelegram =new TreeMap<Integer, CanUnitBean>();

Telegram.put(next.\_1.deviceId, rvTelegram);

}

int tmStamp = (int)(next.\_2.getCanTime());

rvTelegram.put(tmStamp, next.\_2);

}

//--end

HashMap<String, TreeMap<Integer, CanUnitBean>> Telegram100 = new HashMap<String, TreeMap<Integer, CanUnitBean>>();

//遍历刚才创建的Telegram，以100mm区间顺，取时间最小状态

for (Map.Entry<String, TreeMap<Integer, CanUnitBean>>telEntry : Telegram.entrySet()) {

TelegramHash tel = new TelegramHash(telEntry.getKey(), Timestamp.get(telEntry.getKey()));

rvTelegram100 =new TreeMap<Integer, CanUnitBean>();

for(Map.Entry<Integer,CanUnitBean> mm\_cub : telEntry.getValue().entrySet()) {

mm100 = mm\_cub.getKey() - mm\_cub.getKey() % 100

bean100 = rvTelegram100.getValue(mm100)

if(bean100 == null){

//100mm内最早数据包

//所有CanUnitBean状态存入

rvTelegram100.put(mm100, mm\_cub.getValue())

}else{

//100mm内非最早数据包

//对比CanUnitBean中状态，合并还没有的状态

}

}

}

//

//由Telegram100(设备ID，各100毫秒内最早状态)

//创建返回结构

}

}