Lab 3 Objectives

1. Example of creating a comparator

For your first analytic response that is to show the

* **average income per location**

you may want to create a comparator class to sort by location

Create a comparator java class for your project as follows for sorting by location

**import** java.util.Comparator;

**public** **class** LocationComparator **implements** Comparator<BankRecords>{

@Override

**public** **int** compare(BankRecords o1, BankRecords o2) {

// use compareTo to compare strings

**int** result = o1.getRegion().compareTo(o2.getRegion());

**return** result;

}

}

This comparator can be useful for other analytic results involving location sorts.

Think of comparators you can set up with not only a primary sort but secondary sorts as well. Example maybe a female sort as primary and mortgages as secondary.

2. Call up your comparators in your Records.java file.

Ex. in main(), you can call a series of functions to perform your analytic results.

Set up your file to also not only write to the console but to a text file.

See example that follows…

import java.io.FileWriter;

import java.io.IOException;

import java.util.Arrays;

public class Records extends BankRecords{

//create formatted object to write output directly to the

//console and to a file

static FileWriter fw = null;

public Records(){

try {

fw = new FileWriter("bankrecords.txt");

} catch (IOException e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

Records br = new Records();

br.readData();

//call functions to perform analytics

LocationComp(); // analyze average income per loc

//MaxMinComp(); //compare max and min incomes per loc

//femsComp(); // analyze females w. mort/savings accounts per loc

//malesComp(); // analyze male count w. car and 1 child per loc

// \*\*\* close out file object \*\*\*//

try {

fw.close();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

private static void LocationComp() {

Arrays.sort(robjs,new LocationComparator());

//set up needed vars for region counts & incomes per loc

double townCt = 0, innerCt=0, suburbanCt=0,ruralCt=0,

townIncSum = 0, innerIncSum=0, suburbanIncSum=0,ruralIncSum=0;

for (int i=0;i<robjs.length;i++)

if (robjs[i].getRegion().equals("RURAL")) {

ruralIncSum += robjs[i].getIncome();

++ruralCt;

}

else {

//...

}

//setup resulting averages to print to console and to file

double ruralAvg = ruralIncSum/(ruralCt);

System.out.printf(" ... ");

try {

fw.write("Avg inc. for rural region " + ruralAvg);

fw.write("...");

} catch (IOException e) {

e.printStackTrace();

}

}

}

Continue in like manner as above to finish up three other method definitions called

in main().