Name: Sanika Zende

Roll No-57

Practical 1: Implementation code for map

class Table {

String[][] data;

int rowCount;

Table() {

data = new String[10][10];

}

Table(String filename) {

String[] rows = loadStrings(filename);

data = new String[rows.length][];

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

if (trim(rows[i]).length() == 0) {

continue; // skip empty rows

}

if (rows[i].startsWith("#")) {

continue; // skip comment lines

}

// split the row on the tabs

String[] pieces = split(rows[i], TAB);

// copy to the table array

data[rowCount] = pieces;

rowCount++;

// this could be done in one fell swoop via:

//data[rowCount++] = split(rows[i], TAB);

}

// resize the 'data' array as necessary

data = (String[][]) subset(data, 0, rowCount);

}

int getRowCount() {

return rowCount;

}

// find a row by its name, returns -1 if no row found

int getRowIndex(String name) {

for (int i = 0; i < rowCount; i++) {

if (data[i][0].equals(name)) {

return i;

}

}

println("No row named '" + name + "' was found");

return -1;

}

String getRowName(int row) {

return getString(row, 0);

}

String getString(int rowIndex, int column) {

return data[rowIndex][column];

}

String getString(String rowName, int column) {

return getString(getRowIndex(rowName), column);

}

int getInt(String rowName, int column) {

return parseInt(getString(rowName, column));

}

int getInt(int rowIndex, int column) {

return parseInt(getString(rowIndex, column));

}

float getFloat(String rowName, int column) {

return parseFloat(getString(rowName, column));

}

float getFloat(int rowIndex, int column) {

return parseFloat(getString(rowIndex, column));

}

void setRowName(int row, String what) {

data[row][0] = what;

}

void setString(int rowIndex, int column, String what) {

data[rowIndex][column] = what;

}

void setString(String rowName, int column, String what) {

int rowIndex = getRowIndex(rowName);

data[rowIndex][column] = what;

}

void setInt(int rowIndex, int column, int what) {

data[rowIndex][column] = str(what);

}

void setInt(String rowName, int column, int what) {

int rowIndex = getRowIndex(rowName);

data[rowIndex][column] = str(what);

}

void setFloat(int rowIndex, int column, float what) {

data[rowIndex][column] = str(what);

}

void setFloat(String rowName, int column, float what) {

int rowIndex = getRowIndex(rowName);

data[rowIndex][column] = str(what);

}

// Write this table as a TSV file

void write(PrintWriter writer) {

for (int i = 0; i < rowCount; i++) {

for (int j = 0; j < data[i].length; j++) {

if (j != 0) {

writer.print(TAB);

}

if (data[i][j] != null) {

writer.print(data[i][j]);

}

}

writer.println();

}

writer.flush();

}

}

PImage mapImage;

Table locationTable;

int rowCount;

void setup( ) {

size(640, 400);

mapImage = loadImage("map.png");

// Make a data table from a file that contains

// the coordinates of each state.

locationTable = new Table("locations.tsv");

// The row count will be used a lot, so store it globally.

rowCount = locationTable.getRowCount( );

}

void draw( ) {

background(255);

image(mapImage, 0, 0);

// Drawing attributes for the ellipses.

//smooth( );

fill(192, 0, 0);

noStroke( );

// Loop through the rows of the locations file and draw the points.

for (int row = 0; row < rowCount; row++) {

float x = locationTable.getFloat(row, 1); // column 1

float y = locationTable.getFloat(row, 2); // column 2

ellipse(x, y, 9, 9);

}

}

Output:

