import java.io.RandomAccessFile;

import java.io.File;

import java.io.FileReader;

import java.util.Scanner;

import java.util.SortedMap;

import java.util.ArrayList;

import java.util.Arrays;

/\*\*

\* @author Chris Irwin Davis

\* @version 1.0

\* <b>

\* <p>This is an example of how to create an interactive prompt</p>

\* <p>There is also some guidance to get started wiht read/write of

\* binary data files using RandomAccessFile class</p>

\* </b>

\*

\*/

public class DavisBasePromptExample {

/\* This can be changed to whatever you like \*/

static String prompt = "davisql> ";

static String version = "v1.0b(example)";

static String copyright = "Â©2016 Chris Irwin Davis";

static boolean isExit = false;

/\*

\* Page size for alll files is 512 bytes by default.

\* You may choose to make it user modifiable

\*/

static long pageSize = 512;

/\*

\* The Scanner class is used to collect user commands from the prompt

\* There are many ways to do this. This is just one.

\*

\* Each time the semicolon (;) delimiter is entered, the userCommand

\* String is re-populated.

\*/

static Scanner scanner = new Scanner(System.in).useDelimiter(";");

/\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Main method

\*/

public static void main(String[] args) {

/\* Display the welcome screen \*/

splashScreen();

/\* Variable to collect user input from the prompt \*/

String userCommand = "";

while(!isExit) {

System.out.print(prompt);

/\* toLowerCase() renders command case insensitive \*/

userCommand = scanner.next().replace("\n", "").replace("\r", "").trim().toLowerCase();

// userCommand = userCommand.replace("\n", "").replace("\r", "");

parseUserCommand(userCommand);

}

System.out.println("Exiting...");

}

/\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Method definitions

\*/

/\*\*

\* Display the splash screen

\*/

public static void splashScreen() {

System.out.println(line("-",80));

System.out.println("Welcome to DavisBaseLite"); // Display the string.

System.out.println("DavisBaseLite Version " + getVersion());

System.out.println(getCopyright());

System.out.println("\nType \"help;\" to display supported commands.");

System.out.println(line("-",80));

}

/\*\*

\* @param s The String to be repeated

\* @param num The number of time to repeat String s.

\* @return String A String object, which is the String s appended to itself num times.

\*/

public static String line(String s,int num) {

String a = "";

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

a += s;

}

return a;

}

/\*\*

\* Help: Display supported commands

\*/

public static void help() {

System.out.println(line("\*",80));

System.out.println("SUPPORTED COMMANDS");

System.out.println("All commands below are case insensitive");

System.out.println();

System.out.println("\tSELECT \* FROM table\_name; Display all records in the table.");

System.out.println("\tSELECT \* FROM table\_name WHERE rowid = <value>; Display records whose rowid is <id>.");

System.out.println("\tDROP TABLE table\_name; Remove table data and its schema.");

System.out.println("\tVERSION; Show the program version.");

System.out.println("\tHELP; Show this help information");

System.out.println("\tEXIT; Exit the program");

System.out.println();

System.out.println();

System.out.println(line("\*",80));

}

/\*\* return the DavisBase version \*/

public static String getVersion() {

return version;

}

public static String getCopyright() {

return copyright;

}

public static void displayVersion() {

System.out.println("DavisBaseLite Version " + getVersion());

System.out.println(getCopyright());

}

public static void parseUserCommand (String userCommand) {

/\* commandTokens is an array of Strings that contains one token per array element

\* The first token can be used to determine the type of command

\* The other tokens can be used to pass relevant parameters to each command-specific

\* method inside each case statement \*/

// String[] commandTokens = userCommand.split(" ");

ArrayList<String> commandTokens = new ArrayList<String>(Arrays.asList(userCommand.split(" ")));

/\*

\* This switch handles a very small list of hardcoded commands of known syntax.

\* You will want to rewrite this method to interpret more complex commands.

\*/

switch (commandTokens.get(0)) {

case "select":

parseQueryString(userCommand);

break;

case "drop":

System.out.println("STUB: Calling your method to drop items");

dropTable(userCommand);

break;

case "create":

parseCreateString(userCommand);

break;

case "help":

help();

break;

case "version":

displayVersion();

break;

case "exit":

isExit = true;

break;

case "quit":

isExit = true;

default:

System.out.println("I didn't understand the command: \"" + userCommand + "\"");

break;

}

}

/\*\*

\* Stub method for dropping tables

\* @param dropTableString is a String of the user input

\*/

public static void dropTable(String dropTableString) {

System.out.println("STUB: Calling parseQueryString(String s) to process queries");

System.out.println("Parsing the string:\"" + dropTableString + "\"");

}

/\*\*

\* Stub method for executing queries

\* @param queryString is a String of the user input

\*/

public static void parseQueryString(String queryString) {

System.out.println("STUB: Calling parseQueryString(String s) to process queries");

System.out.println("Parsing the string:\"" + queryString + "\"");

}

/\*\*

\* Stub method for creating new tables

\* @param queryString is a String of the user input

\*/

public static void parseCreateString(String createTableString) {

System.out.println("STUB: Calling your method to create a table");

System.out.println("Parsing the string:\"" + createTableString + "\"");

ArrayList<String> createTableTokens = new ArrayList<String>(Arrays.asList(createTableString.split(" ")));

/\* Define table file name \*/

String tableFileName = createTableTokens.get(2) + ".tbl";

/\* YOUR CODE GOES HERE \*/

/\* Code to create a .tbl file to contain table data \*/

try {

/\* Create RandomAccessFile tableFile in read-write mode.

\* Note that this doesn't create the table file in the correct directory structure

\*/

RandomAccessFile tableFile = new RandomAccessFile(tableFileName, "rw");

tableFile.setLength(pageSize);

}

catch(Exception e) {

System.out.println(e);

}

/\* Code to insert a row in the davisbase\_tables table

\* i.e. database catalog meta-data

\*/

/\* Code to insert rows in the davisbase\_columns table

\* for each column in the new table

\* i.e. database catalog meta-data

\*/

}

}