Jarrod Parkes

CS 321 – Introduction to Object-Oriented Programming with Java

Programming Assignment #1

Table Class – Assignment Description

Problem Statement: The first programming assignment focuses on the implementation and application of a two-dimensional table class. The table class will require the use of the basic functions of an Array, ArrayList, or both. Along with the basic functions, the table class will provide the requirements listed below and any all other supplementary requirements used for the basic of a type of collection.

Table Class – Requirements

1. The program will be written using the Java programming language
2. The program will be compiled without errors using the NetBeans 6.9.1 IDE
3. The program will execute in a timely manner
4. The program will contain adequate comments and documentation
5. The Table class will provide a function for printing a row (record)
6. The Table class will provide a function for printing a column
7. The Table class will provide a function for printing the entire table
8. The Table class will provide a function for producing the numerical or character sum (in the form of a String) for both a row or a column
9. The Table class will provide a function for adding a row or column
10. The Table class will provide a function for removing a row or column
11. The Table class will provide a function that produces an adequate “deep” copy in the form of a copy constructor
12. (Elective Method #1) The Table class will provide a function that allows users to fill a single row completely are partially with instances of Object type
13. (Elective Method #2) The Table class will provide a function for resetting the table dimensions as well as clearing out element values
14. The Table class shall provide an implementation (.java file) with appropriate method headers and sufficient abstraction by specification
15. The Table class shall be accompanied by a driver file (Main.java)
16. The driver file that accompanies the Table class shall be adequately commented and documented as would any other addition file
17. The driver portion of the program shall be tested for reliability and completion of all stated requirements

Table Class – Specification of General Design for Overall Project

Since the Table class will be implemented as a two-dimensional java.lang.Object array, all iterations and access of table elements is done rather directly through access at indexed locations. Table instances will have names, rectangular dimensions, and format specifications for types of elements that may exist in a table and need to be printed. Formats will be particular to notice the row and column numbers as well as defining a clear column width so that the table lines up evenly. Modifier functions for the Table class include those to reallocate a new table, clear rows and columns, remove rows and columns, and access cells individually. In the case of logical errors within the test cases, simple print statements have been added to notify of incorrect use of the classes and or their underlying data types. The designs for the rudimentary Table class will be similar in respects to a Microsoft excel spreadsheet without the graphical interface or ability to use device input to direct program execution. Print formatting will handle in these ways: (1) empty nodes will be displayed with empty element ‘ ‘ (2) numbers will attempt for a precision of two decimal places, but may lack or exceed that number based on number of digits although the value will not be altered (3) and strings will display at most ten characters before being truncated. Also, in addition to those required methods of the assignment of the Table class, a collection class called Database will be implemented using a Java ArrayList to store multiple instances of the Table class using simple methods of adding, removing, and displaying.

Table Class - UML Diagram

|  |
| --- |
| Table |
| - strFormat: String  - numFormat : String  - nullFormat : String  - output : PrintStream  - name : String  - values : Object[][] |
| + Table()  + Table(int,int)  + Table(string,int,int)  + Table(Table)  - copy() : Object[][]  - copyTo(Object[][])  + resetTable(int,int)  + resetTable(string,int,int)  - noTable() : boolean  + deleteTable()  + addRow()  + clearRow(int)  + fillRow(int, Object[])  + removeRow(int)  + addColumn()  + clearColumn(int)  + removeColumn(int)  + print()  + printRow(int)  + printColumn(int)  + sumRow(int)  + sumColumn(int)  + setCell(Object,int,int)  + getCell(int,int) : String  + getName() : String  + setName(String) |

Table Class – Specification of General Design for Complex Methods

1. addRow() and addColumn()

- These methods both begin by creating empty instances of a two-dimensional Object array that is either one column or row larger than the original array before the function call. Then, the elements from the previous table are systematically added back to the new table disregarding the new empty elements.

2. removeRow() and removeColumn()

- In a similar fashion to the above methods, each of these functions begins by creating a new Object[][] array that has one less row or column. Then loop iterations to refill the new Table are skipped over when the row or column to be copied is equal to the specified row or column to be deleted.

3. print(), printRow(), and printColumn()

- Each of these functions displays in the same way by providing a separate type of format for numerical, mixed, and empty elements. Outside of basic loop operations to make sure each table element is included, the only other notable features are that strings are truncated (only while displayed) at ten characters and numerical values attempt to have at least two decimal place precision.

4. sumRow() and sumColumn()

- These functions operate by iterating through the row or column while summing any numerical values, tracking null values, and concatenating character values to a string. After this is complete, the type of summation is determined by the number of null values, string length, or subsequently the mathematical sum.

Table Class – Testing

The test cases listed below are also found commented in the Main.java file provided in the soft copy of this project. To make sure that testing was unbiased, a user who never saw the development of the project was allowed to test the classes for reliability.

1a. Test #1

/\* Test Case #1 - Table Class

\*

\* Functions being Tested:

\* constructor

\* setCell

\* setName

\* getCell

\* getNam

\* print

\*/

Table t1 = new Table("Table 1", 2, 2);

t1.print();

t1.setCell(4, 0, 1);

t1.setCell("james", 3, 5);

t1.getCell(0, 0);

t1.getCell(0, 1);

t1.getCell(4, 5);

System.out.print(t1.getName() + "\n");

t1.setName("Table 001");

System.out.print(t1.getName() + "\n");

t1.print();

1b. Associated Output

Table 1

0 1

0 [ ][ ]

1 [ ][ ]

Error in 'setCell(Object, int, int)' InvalidRow

Handled: Fails Silently...

Error in 'getCell(int, int)' EmptyCell

Handled: Fails Silently...

Error in 'getCell(int, int)' InvalidRow

Handled: Fails Silently...

Table 1

Table 001

Table 001

0 1

0 [ ][ 4.00]

1 [ ][ ]

2a. Test #2

/\* Test Case #2 - Table Class

\*

\* Functions being Tested:

\* constructor

\* addColumn

\* addRow

\* clearColumn

\* clearRow

\* fillRow

\* removeColumn

\* removeRow

\* sumColumn

\* sumRow

\* print

\*/

Table t2 = new Table();

t2.print();

t2.addColumn();

t2.addRow();

t2.addColumn();

t2.addRow();

t2.addColumn();

t2.addColumn();

t2.fillRow(4, new Object[]{4 ,4, "dog", 'h'});

t2.fillRow(2, new Object[]{1.0, 2.0, 3.0, 4.0});

t2.fillRow(0, new Object[]{7.4f, "james"});

t2.print();

t2.printColumn(7);

t2.printColumn(2);

t2.printRow(2);

t2.printRow(9);

t2.removeRow(5);

t2.removeColumn(8);

t2.removeColumn(2);

t2.removeRow(1);

t2.print();

t2.sumRow(5);

t2.sumRow(1);

t2.sumRow(0);

t2.sumColumn(2);

t2.sumColumn(0);

t2.clearColumn(1);

t2.print();

t2.sumRow(1);

t2.sumColumn(1);

t2.clearRow(1);

t2.print();

2b. Associated Output

Error in 'print()' NoTableExists

Handled: Fails Silently...

Error in 'addColumn()' NoTableExists

Handled: (1 x 1) Table created

Error in 'fillRow(int, Object[])' InvalidRow

Handled: Fails Silently...

Table 0

0 1 2 3

0 [ 7.40][ james][ ][ ]

1 [ ][ ][ ][ ]

2 [ 1.00][ 2.00][ 3.00][ 4.00]

Error in 'printColumn(int)' InvalidColumn

Handled: Fails Silently...

Column 2

2

0 [ ]

1 [ ]

2 [ 3.00]

Row 2

0 1 2 3

2 [ 1.00][ 2.00][ 3.00][ 4.00]

Error in 'printRow(int)' InvalidRow

Handled: Fails Silently...

Error in 'removeRow(int)' InvalidRow

Handled: Fails Silently...

Error in 'removeColumn(int)' InvalidColumn

Handled: Fails Silently...

Table 0

0 1 2

0 [ 7.40][ james][ ]

1 [ 1.00][ 2.00][ 4.00]

Error in 'sumRow(int)' InvalidRow

Handled: Fails Silently...

Sum Row 1

7.00000

Sum Row 0

7.4james

Sum Column 2

4.00000

Sum Column 0

8.40000

Table 0

0 1 2

0 [ 7.40][ ][ ]

1 [ 1.00][ ][ 4.00]

Sum Row 1

5.00000

Sum Column 1

Empty Column

Table 0

0 1 2

0 [ 7.40][ ][ ]

1 [ ][ ][ ]

3a. Test #3

/\* Test Case #3 - Table Class

\*

\* Functions being Tested:

\* constructor

\* deleteTable

\* resetTable

\* print

\*/

Table t3 = new Table(4, 4);

t3.print();

t3.deleteTable();

t3.printColumn(4);

t3.printRow(3);

t3.print();

t3.resetTable(2, 2);

t3.print();

t3.resetTable("My First Table", 4, 4);

t3.print();

3b. Associated Output

Table 0

0 1 2 3

0 [ ][ ][ ][ ]

1 [ ][ ][ ][ ]

2 [ ][ ][ ][ ]

3 [ ][ ][ ][ ]

Error in 'printColumn(int)' NoTableExists

Handled: Fails Silently...

Error in 'printRow(int)' NoTableExists

Handled: Fails Silently...

Error in 'print()' NoTableExists

Handled: Fails Silently...

Table 0

0 1

0 [ ][ ]

1 [ ][ ]

My First Table

0 1 2 3

0 [ ][ ][ ][ ]

1 [ ][ ][ ][ ]

2 [ ][ ][ ][ ]

3 [ ][ ][ ][ ]

4a. Test #4

/\* Test Case #4 - Database Class

\*

\* Functions being Tested:

\* constructor

\* printTable

\* printAll

\* removeTable

\* clearDatabase

\*/

Table t4 = new Table("Table 01", 5, 5);

Table t5 = new Table("Table 02", 1, 1);

Table t6 = new Table("Table 03", 3, 4);

Database d1 = new Database(new Table[]{t4, t5, t6});

d1.printAll();

d1.printTable(2);

d1.printTable(5);

d1.removeTable(9);

d1.removeTable(0);

d1.printAll();

d1.clearDatabase();

d1.printTable(0);

d1.printAll();

4b. Associated Output

Table 01

0 1 2 3 4

0 [ ][ ][ ][ ][ ]

1 [ ][ ][ ][ ][ ]

2 [ ][ ][ ][ ][ ]

3 [ ][ ][ ][ ][ ]

4 [ ][ ][ ][ ][ ]

Table 02

0

0 [ ]

Table 03

0 1 2 3

0 [ ][ ][ ][ ]

1 [ ][ ][ ][ ]

2 [ ][ ][ ][ ]

Table 03

0 1 2 3

0 [ ][ ][ ][ ]

1 [ ][ ][ ][ ]

2 [ ][ ][ ][ ]

Error in 'printTable(int)' InvalidRow

Handled: Fails Silently...

Error in 'removeTable(int)' InvalidRow

Handled: Fails Silently...

Table 02

0

0 [ ]

Table 03

0 1 2 3

0 [ ][ ][ ][ ]

1 [ ][ ][ ][ ]

2 [ ][ ][ ][ ]

Error in 'prinTable(int)' Empty Database

Handled: Fails Silently...

Error in 'printAll()' Empty Database

Handled: Fails Silently...

Table Class – Intended Modifications Summary

The Table class is far from perfect, but at this point in the development contains many of the base functions for which the class may be expanded on in the future. The most intriguing modification would involve some type of graphic interface in which the user would create tables using buttons instead of modified source code. If this were to be the case, new functions would probably be added for filling a column full of values, doing other mathematical operations on rows, columns, or elements that are not within the same rows and columns. The Table class would also need support for display options and functions that the end user would be allowed to access for preference of display, etc. Other fixes and modifications include, but are not limited to, a Table constructor for a two dimensional object array, sorting algorithms for rows and columns, formatting options such as cell background color or font color, a sophisticated debugger, Java exception handling, and providing official bounds or limitations for table sizes and values.

Table Class – Intended Modifications Listed

1. Graphical Interface
2. More formatting options and functions
3. Column or table fill functions
4. A library of mathematical operations for tables, row, and columns
5. Sorting algorithms
6. Sophisticated debugger
7. Java exception handling
8. Table size bounds and value bounds
9. Use of ArrayLists for underlying table data structure
10. Ability to select elements to be used in math operations, etc.
11. Ability to store other forms of media instead of just text into tables
12. Etc., but hopefully feature creeping is minimized!