/\*Aaron Knestaut

\* 11.15.15

\* Period: A

\*

\* \*\*\*\*\*Program Description\*\*\*\*\*

\* The program generated a user specified amount of fibonacci numbers and places

\* them into a 2D array

\* \*\*\*\*\*Variable Dictionary\*\*\*\*\*

\* int col - keeps track of what column is being written to

\* String colamount - holds the value for the amount of columns the user wants

\* int colnum - the int version of the amount of columns the user wants

\* int count - a counter for when data is being stored to the 2D array

\* int fib - the int that holds the current fibonacci number

\* int fibmatrix[][] - the 2D array

\* int row - keeps track of what row is being written to

\* int rowamount - holds the value for the amount of rows the user wants

\* int rownum - the int version of the amount of rows the user wants

\*/

import javax.swing.JOptionPane;

public class FibonacciMatrix

{

public static void main (String args [])

{

String rowamount = JOptionPane.showInputDialog (null, "How many rows should be generated?");

String colamount = JOptionPane.showInputDialog (null, "How many columns should be generated?");

int rownum = Integer.parseInt (rowamount);

int colnum = Integer.parseInt (colamount);

int fibmatrix [] [] = new int [(rownum + 1)] [(colnum + 1)];

matrix (rownum, colnum, fibmatrix);

}

public static int fibonacci (int fib)

{

if (fib == 0 || fib == 1)

return fib;

else

return fibonacci(fib - 1) + fibonacci(fib - 2);

}

public static void matrix (int rownum, int colnum, int fibmatrix [] [])

{

int total = rownum \* colnum;

int count = 1;

for (int row = 1; row <= rownum; row++)

for (int col = 1; col <= colnum; col++)

{

fibmatrix [row] [col] = fibonacci(count);

count = count + 1;

}

System.out.println("\n\nFibonacci series upto " + total +" numbers : ");

for(int i = 1; i <= rownum; i++){

for (int j = 1; j <= colnum; j++){

System.out.print(fibmatrix[i][j] + " ");

}

System.out.println("");

}

}

}