# WORKSHEET

# Two-Dimensional Arrays

1. Determine the output of the following code using the provided text file (*data.txt*) on page two:

**import** java.util.Scanner;

**public class** TwoDArray {

**final** **private static int** NUM = 6;

**public** TwoDArray(){

}

**public** **void** load(int[][] grid){

**int** row, col;

String fileName = "data.txt";

try{

Scanner inFile = **new** Scanner(new File(filename));

**for** (row = 0; row < NUM; row++){

**for** (col = 0; col < NUM; col++){

**grid**[row][col] = inFile.readInt();

}

}

}catch(IOException e){

System.out.println(“Error: “ + e.getMessage());

}

}

**public** **void** display(int[][] grid){

**int** row, col;

**for** (row = 0; row < NUM; row++){

**for** (col =0; col < NUM; col++){

System.out.print(grid[row][col] + " ");

}

System.out.println();

}

System.out.println();

}

**public** void fun(int[][] grid){

**int** row, col;

**for** (row = 0; row < NUM; row++){

**for** (col = 0; col < NUM; col++)

**if** ((grid[row][col] % 2) == 0){

grid[row][col] = 0;

}

}

}

}

}

**public** **class** driver{

**public** **static void main**(String[] args){

TwoDArray app = **new** TwoDArray();

**int** [][] matrix = **new** **int**[NUM][NUM];

app.load(matrix);

app.display(matrix);

app.fun(matrix);

app.display(matrix);

}

}

*data.txt*

5 8 4 3 9 5

6 4 9 5 3 2

2 2 0 9 7 3

7 4 5 6 9 5

8 8 3 2 6 4

9 5 6 3 7 6

2. A cell in any array can have up to four diagonal neighbors (i.e., in the northwest, northeast, southwest and southeast directions). Using the results from the *fun* method, write code that directs each cell to simultaneously replace its value with its number of diagonal neighbors that hold a value of zero. Since this action is simultaneous, make sure that you check each cell against a copy of the current array.

For example, assume NUM was changed to 4 and the following array was read into the program:

2 5 4 9

0 5 6 3

1 9 4 6

7 2 6 9

After execution of the original *fun* method, the array would appear as:

0 5 0 9

0 5 0 3

1 9 0 0

7 0 0 9

After executing the code required for this problem, the array should appear as:

0 2 0 1

0 3 1 2

1 3 1 2

0 1 1 1

Using the original *data.txt* file and implementing the enhancements from this problem would create the following array:

1 1 1 0 1 0

2 3 2 2 0 0

2 1 3 0 2 0

2 3 3 2 2 1

1 1 2 1 2 0

1 1 2 1 2 1