AP Computer Science A

Java Programming Essentials [Ver. 2.0]

Unit 3: Basic Data Structure

WEEK 12: CHAPTER 8 2-D ARRAY

DR. ERIC CHOU IEEE SENIOR MEMBER

Objectives

- Motivation of Using 2D Arrays.
- Declare, Instantiate and initialization of a 2D array.
- 2D Array Processing I: 2D Traversal, 2D Max/Min, and 2D Shuffling.
- 2D Array Processing II: 2D Index Space, Column Major/Row Major, area copy, area move, and flip

Objectives

- Image Processing
- Symmetric 2D Array: k-graph, closest points
- Combination Number: Binomial Theorem
- N-D Arrays

Overview

LECTURE 1

2D arrays

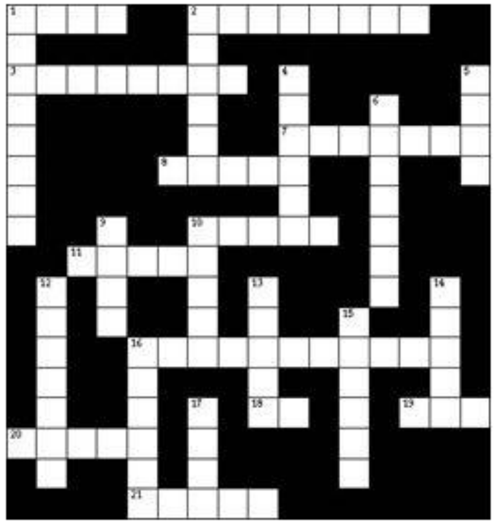
Many applications have multidimensional structures:

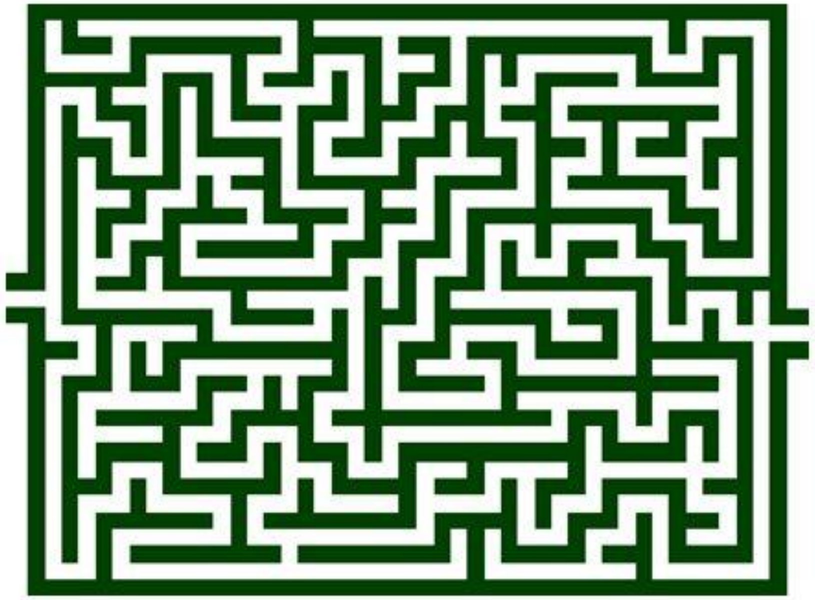
- Matrix operations
- Collection of lists
- Board games (Chess, Checkers)
- Images (rows and columns of pixels)
- ...

$$\mathbf{A}=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Applications

- 2D arrays are useful when data can be represented by a a grid of fixed dimensions
- Often used to represent tables, matrices, images, and game boards
- Examples of games include checkers, chess, tic-tac-toe, crosswords, and mazes





Multidimensional Array

- Thus far, you have used one-dimensional arrays to model linear collections of elements.
- You can use a two-dimensional array to represent a matrix or a table.
- For example, the following table that describes the distances between the cities can be represented using a two-dimensional array.

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

Declare/Create Two-dimensional Arrays

```
// Declare array ref var
ElementType[][] refVar; /*or*/ ElementType refVar[][]; /*not preferred */

// Create array and assign its reference to variable
refVar = new ElementType[10][10];

// Combine declaration and creation in one statement
ElementType[][] refVar = new ElementType[10][10];

// Alternative syntax
ElementType refVar[][] = new ElementType[10][10]; /*not preferred */
```

Declaring Variables of Two-dimensional Arrays and Creating Two-dimensional Arrays

```
int[][] matrix = new int[10][10];
```

or

```
int matrix[][] = new int[10][10];
matrix[0][0] = 3;

for (int i = 0; i < matrix.length; i++)
  for (int j = 0; j < matrix[i].length; j++)
    matrix[i][j] = (int)(Math.random() * 1000);

double[][] x;
```

Two-dimensional Array Illustration

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```
matrix = new int[5][5];
```

matrix.length? 5
matrix[0].length? 5

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	7	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```
matrix[2][1] = 7;
```

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6
[2]	7	8	9
[3]	10	11	12

```
int[][] array = {
   {1, 2, 3},
   {4, 5, 6},
   {7, 8, 9},
   {10, 11, 12}
};
```

array.length? 4
array[0].length? 3

Declaring, Creating, and Initializing Using Shorthand Notations

- You can also use an array initializer to declare, create and initialize a two-dimensional array. For example,

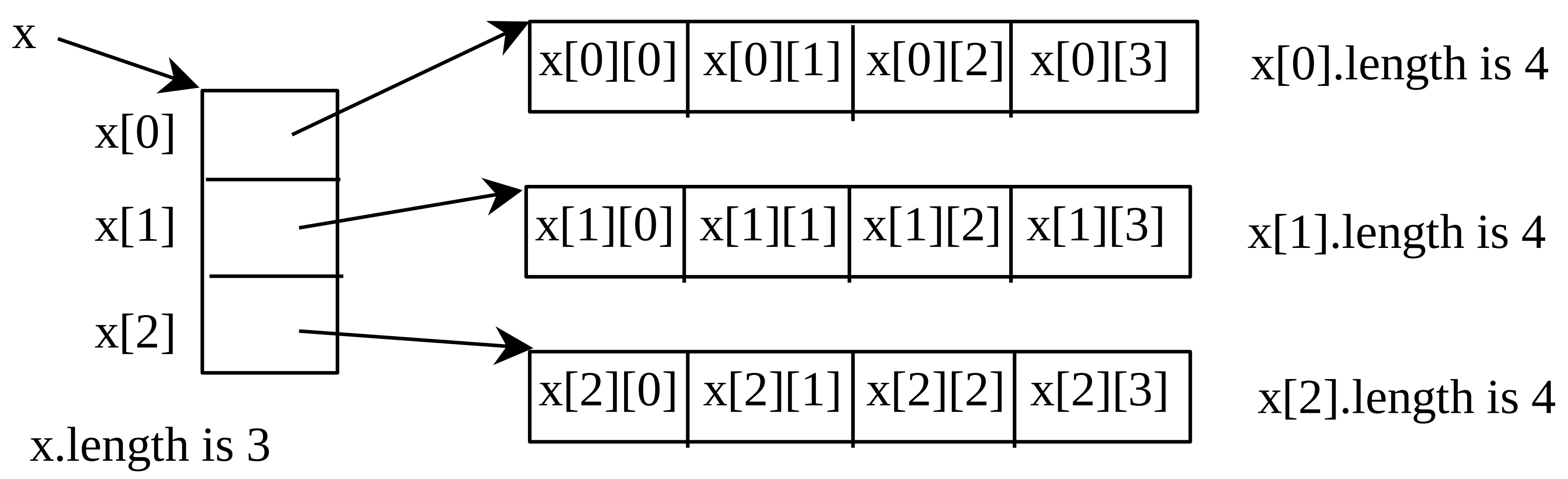
```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Same as

```
int[][] array = new int[4][3];
array[0][0] = 1; array[0][1] = 2; array[0][2] = 3;
array[1][0] = 4; array[1][1] = 5; array[1][2] = 6;
array[2][0] = 7; array[2][1] = 8; array[2][2] = 9;
array[3][0] = 10; array[3][1] = 11; array[3][2] = 12;
```

Lengths of Two-dimensional Arrays

```
int[][] x = new int[3][4];
```


x
x[0]
x[1]
x[2]
x[0][0] x[0][1] x[0][2] x[0][3]
x[1][0] x[1][1] x[1][2] x[1][3]
x[2][0] x[2][1] x[2][2] x[2][3]
x[0].length is 4
x[1].length is 4
x[2].length is 4
x.length is 3

Lengths of Two-dimensional Arrays, cont.

```
int[][] array = {
   {1, 2, 3},
   {4, 5, 6},
   {7, 8, 9},
   {10, 11, 12}
};
```

```
array.length
array[0].length
array[1].length
array[2].length
array[3].length
```

array[4].length ArrayIndexOutOfBoundsException

Ragged Arrays

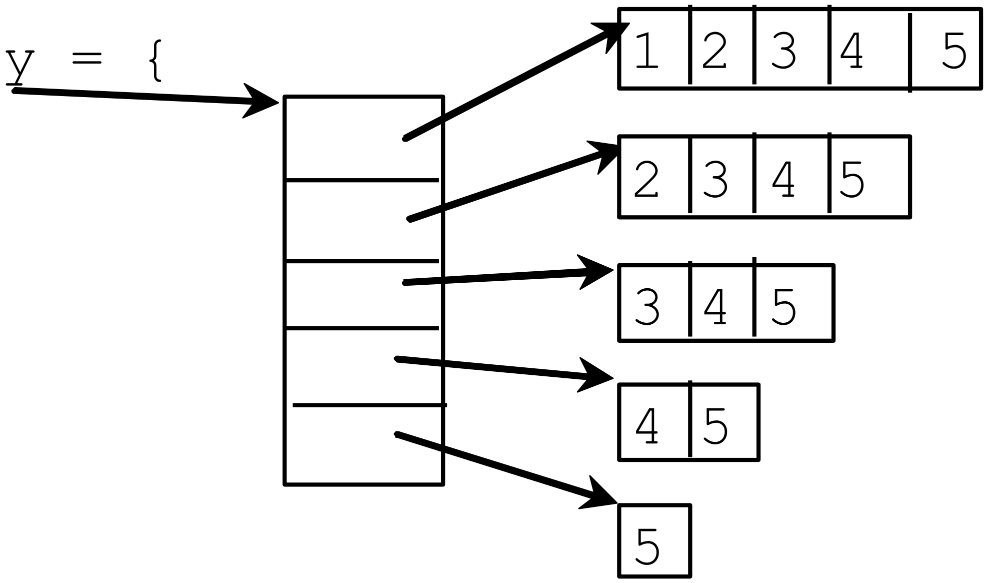
- Each row in a two-dimensional array is itself an array. So, the rows can have different lengths. Such an array is known as *a ragged array*. For example,

```
int[][] matrix = {
   {1, 2, 3, 4, 5},
   {2, 3, 4, 5},
   {3, 4, 5},
   {4, 5},
   {5}
};
```

matrix.length is 5
matrix[0].length is 5
matrix[1].length is 4
matrix[2].length is 3
matrix[3].length is 2
matrix[4].length is 1

Ragged Arrays, cont.

```
int[][] triangleArray = {
   {1, 2, 3, 4, 5},
   {2, 3, 4, 5},
   {3, 4, 5},
   {4, 5},
   {5}
};
```



2D Array Processing I

LECTURE 2

Processing Two-Dimensional Arrays

1. (Initializing arrays with input values)
2. (Printing arrays)
3. (Summing all elements)
4. (Summing all elements by column)
5. (Which row has the largest sum)
6. (Finding the smallest index of the largest element)
7. (*Random shuffling*)

Initializing arrays with input values

```
Scanner input = new Scanner(System.in);
System.out.println("Enter " + matrix.length + " rows and " +
  matrix[0].length + " columns: ");
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column < matrix[row].length; column++) {
    matrix[row][column] = input.nextInt();
  }
}
```

Initializing arrays with random values

```
for (int row = 0; row < matrix.length; row++) {
   for (int column = 0; column <
    matrix[row].length; column++) {
      matrix[row][column] = (int)(Math.random() *
    100);
   }
}
```

Printing arrays

```
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column <
   matrix[row].length; column++) {
    System.out.print(matrix[row][column] + " ");
  }
  System.out.println();
}
```

Summing all elements

```
int total = 0;
for (int row = 0; row < matrix.length; row++) {
  for (int column = 0; column <
   matrix[row].length; column++) {
    total += matrix[row][column];
  }
}
```

Summing elements by column

```
for (int column = 0; column < matrix[0].length;
   column++) {
  int total = 0;
  for (int row = 0; row < matrix.length; row++)
    total += matrix[row][column];
  System.out.println("Sum for column " + column + "
   is " + total);
}
```

Which row has the largest sum

```
int sum = 0;
int maxSum = Integer.MIN_VALUE; int maxRow = 0;
for (int i=0; i<matrix.length; i++){
  sum = 0;
  for (int j=0; j<matrix[i].length; j++){
       sum += matrix[i][j];
    }
  if (sum > maxSum) {maxSum = sum; maxRow = i; }
}
System.out.println("Row: "+maxRow+
                   "has the largest sum="+maxSum);
```

Finding the smallest index of the largest element

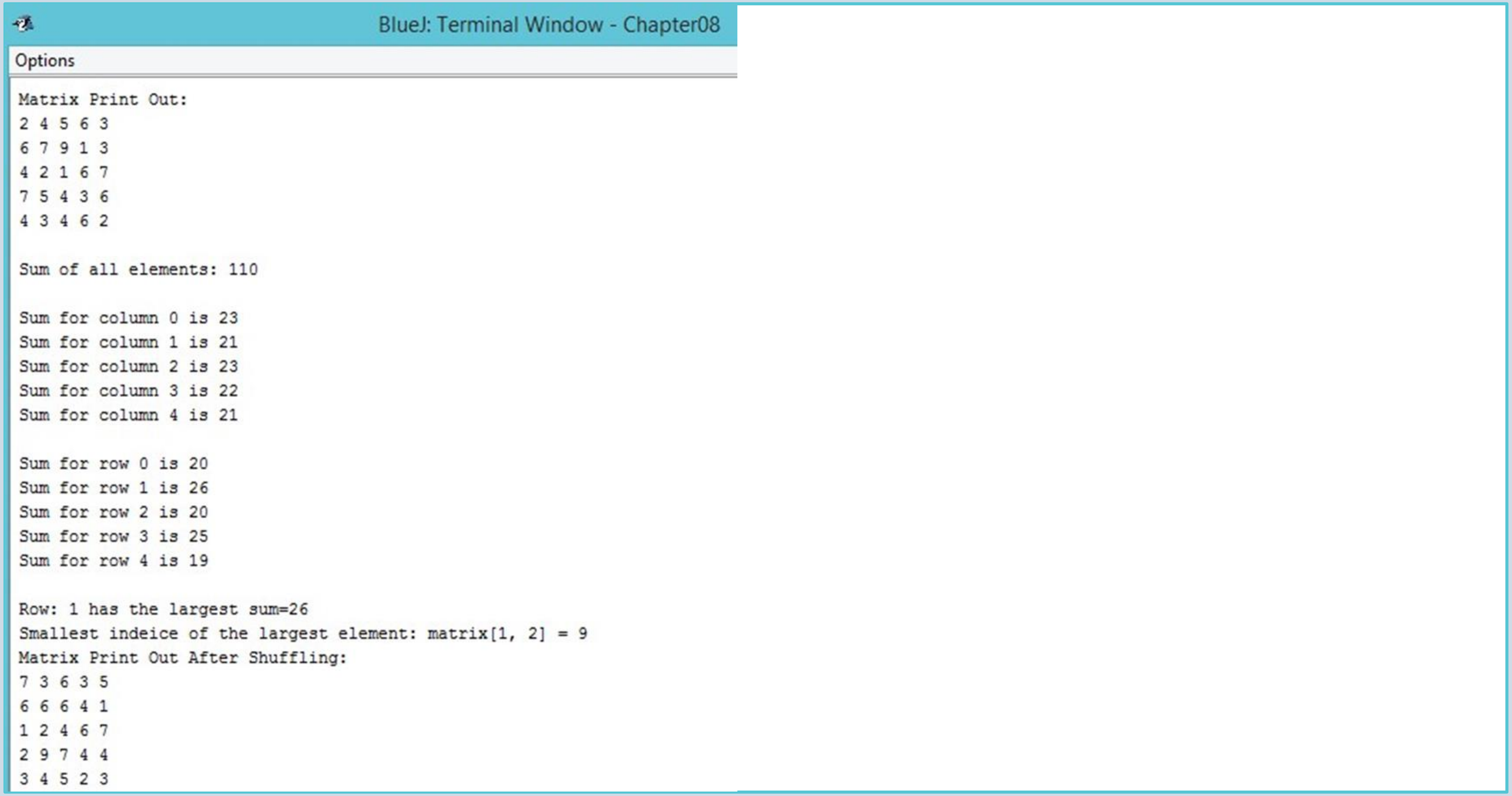
```
int maxi=0; int maxj =0;  int max = matrix[0][0];
for (int i = 0; i < matrix.length; i++) {
  for (int j = 0; j < matrix[i].length; j++) {
              if (max < matrix[i][j]) {
               maxi = i; maxj = j; max = matrix[i][j];
             }
   }
}
```

Random shuffling

```java
for (int i = 0; i < matrix.length; i++) {
  for (int j = 0; j < matrix[i].length; j++) {
    int i1 = (int)(Math.random() * matrix.length);
    int j1 = (int)(Math.random() * matrix[i].length);
    // Swap matrix[i][j] with matrix[i1][j1]
    int temp = matrix[i][j];
    matrix[i][j] = matrix[i1][j1];
    matrix[i1][j1] = temp;
  }
}
```

Demonstration Program

ARRAYPROCESSING2D.JAVA



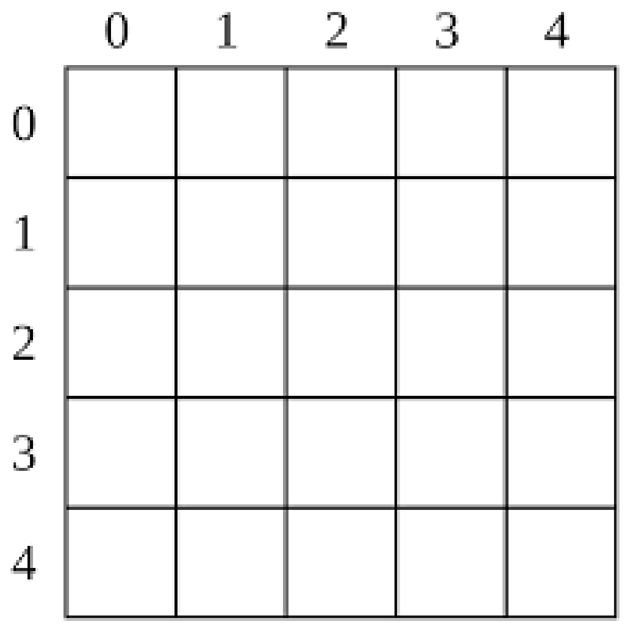
```
Matrix Print Out:
2 4 5 6 3
6 7 9 1 3
4 2 1 6 7
7 5 4 3 6
4 3 4 6 2

Sum of all elements: 110

Sum for column 0 is 23
Sum for column 1 is 21
Sum for column 2 is 23
Sum for column 3 is 22
Sum for column 4 is 21

Sum for row 0 is 20
Sum for row 1 is 26
Sum for row 2 is 20
Sum for row 3 is 25
Sum for row 4 is 19

Row: 1 has the largest sum=26
Smallest indeice of the largest element: matrix[1, 2] = 9
Matrix Print Out After Shuffling:
7 3 6 3 5
6 6 6 4 1
1 2 4 6 7
2 9 7 4 4
3 4 5 2 3
```

2D Array Processing II

LECTURE 3

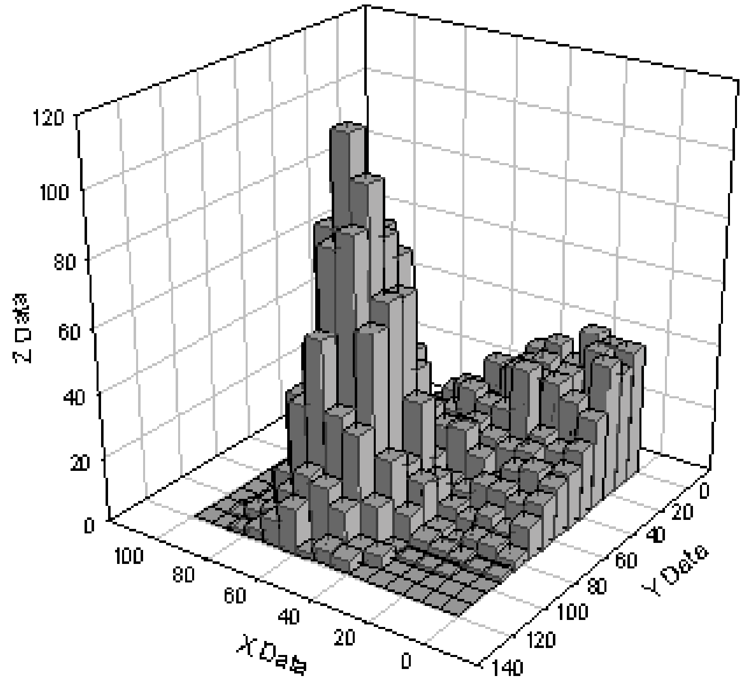
Discrete Functional Model

int[][] m = new int[5][5];



2-D Discrete Functional Model:

f(x, y) = m[i][j];



Nested Loop

```
int i = 0;     // outer loop initial condition
/*  Before Outter Loop Processing */
while (i< m.length){
     int j = 0;          // inner loop initial condition
     /* Before Inner Loop Processing */
     while (j<m[0].length){
         /* cell processing */
         j++;
         // index update for inner loop
    } /* after inner loop processing (in-between rows(cols)) */
    i++;
         // index update of outer loop
}
/* after outer loop processing  (grand total )*/
```

9 x 9 multiplication table

```
int i= 1;
int sum = 0;
int[] rsum = new int[9+1];    // rsum[0] not used
while  (i<=9){
   int j = 0;
   rsum[i] = 0;    // can be omitted
   while  (j<= 9){
        rsum[i] += i * j;
        System.out.printf("%3d  ", i*j);
        j++;
   }
   System.out.println();
   sum += rsum[i];
   i++;
}
System.out.println(sum);
```

Use (i, j) to create 2D index space

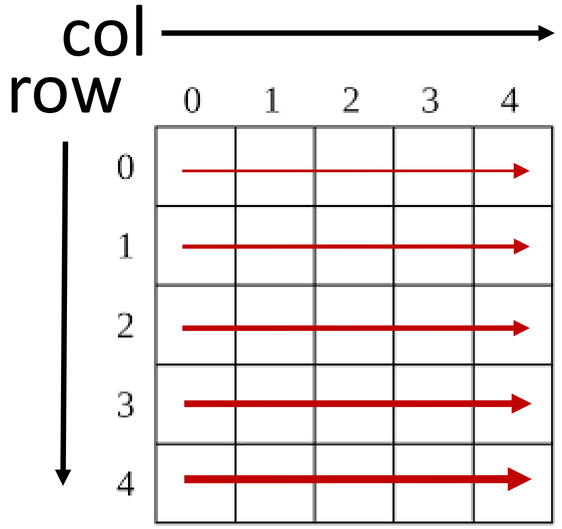
```
int i = 0;
/*  Before Outter Loop Processing */
while (i< m.length){
     int j = 0;
     /* Before Inner Loop Processing */
     while (j<m[0].length){
         /* cell processing */
         j++;
     }
     /*after inner loop processing (in-between rows(cols))*/
     i++;
}
/* after outer loop processing  (grand total )*/
```

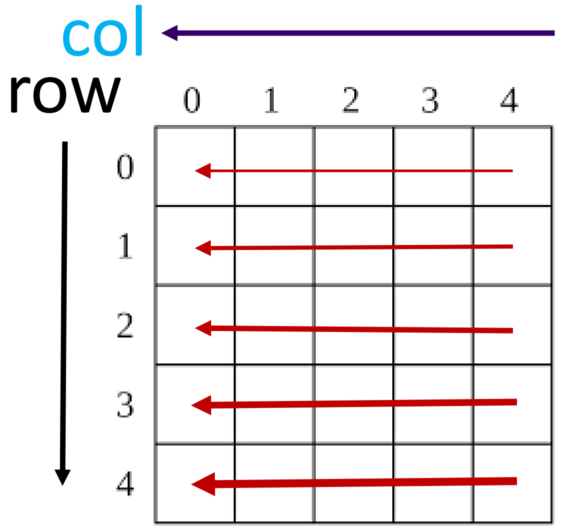
Column Major versus Row Major

- If the column index is used at outer loop, it is called a column major system.
- If the row index is used at the outer loop, it is called a row major system.

2-D Array Indexing for Traversal

int row, col; int[][] m=new int[5][5];

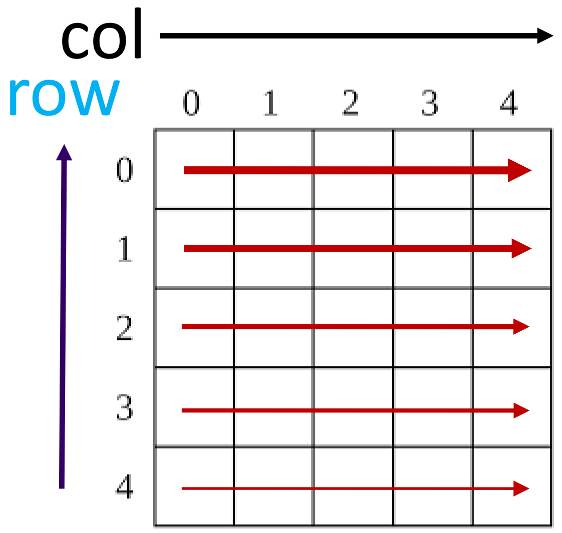


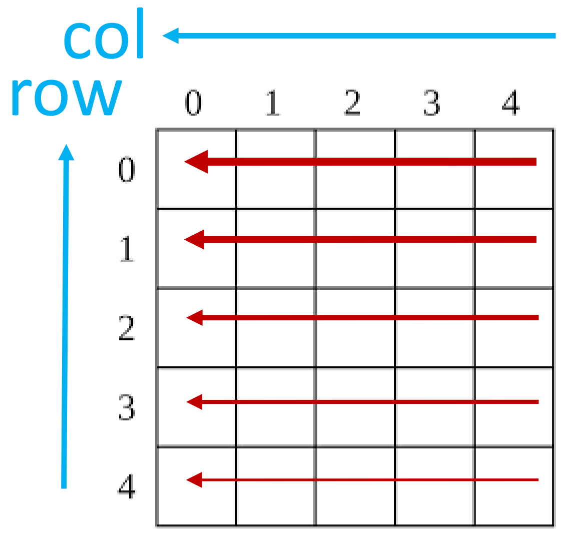


```
for(row=0; row<m.length; row++)
   for(col=0; col<m[0].length; col++)
          System.out.println(m[row][col]);
```

```
for(row=0; row<m.length; row++)
   for(col=m[0].length-1; col>=0; col--)
          System.out.println(m[row][col]);
```

2-D Array Indexing for Traversal
int row, col; int[][] m=new int[5][5];



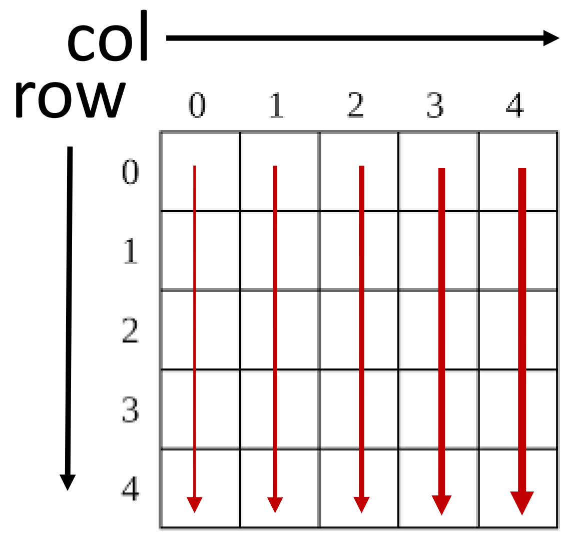


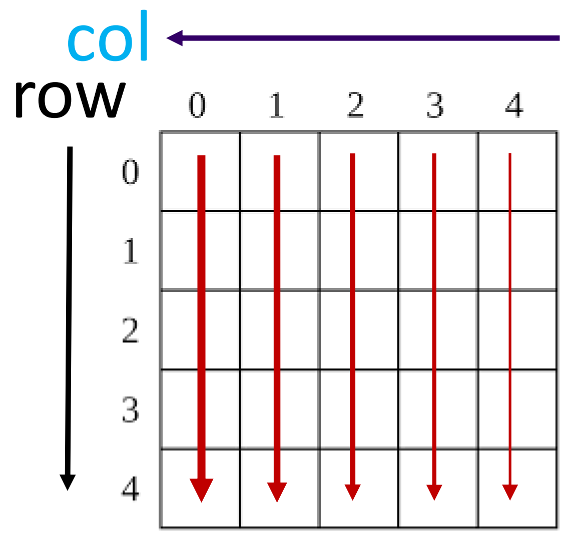
```
for(row=m.length-1; row>=0; row--)
    for(col=0; col<m[0].length; col++)
        System.out.println(m[row][col]);
```

```
for(row=m.length-1; row>=0; row--)
    for(col=m[0].length-1; col>=0; col--)
        System.out.println(m[row][col]);
```

2-D Array Indexing for Traversal

int row, col; int[][] m=new int[5][5];

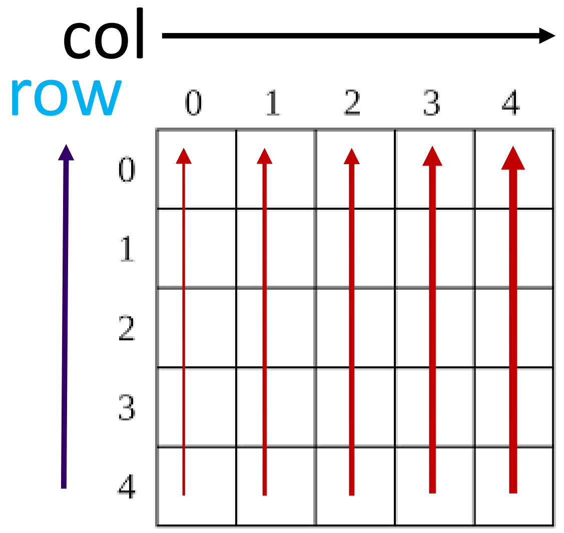


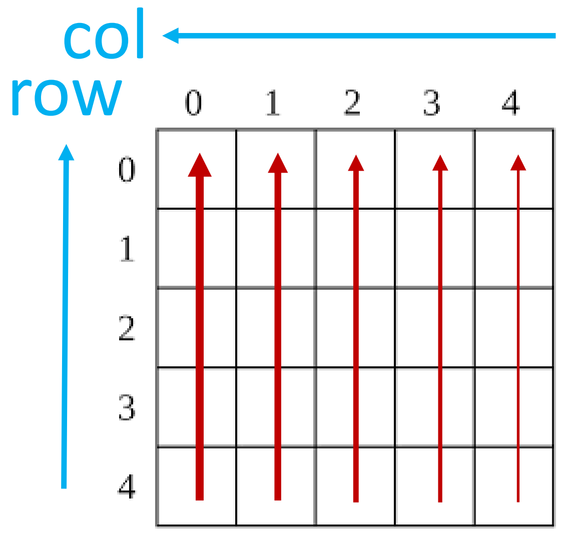


```
for(col=0; col<m[0].length; col++)
   for(row=0; row<m.length; row++)
          System.out.println(m[row][col]);
```

```
for(col=m[0].length-1; col>=0; col--)
    for(row=0; row<m.length; row++)
        System.out.println(m[row][col]);
```

2-D Array Indexing for Traversal
int row, col; int[][] m=new int[5][5];

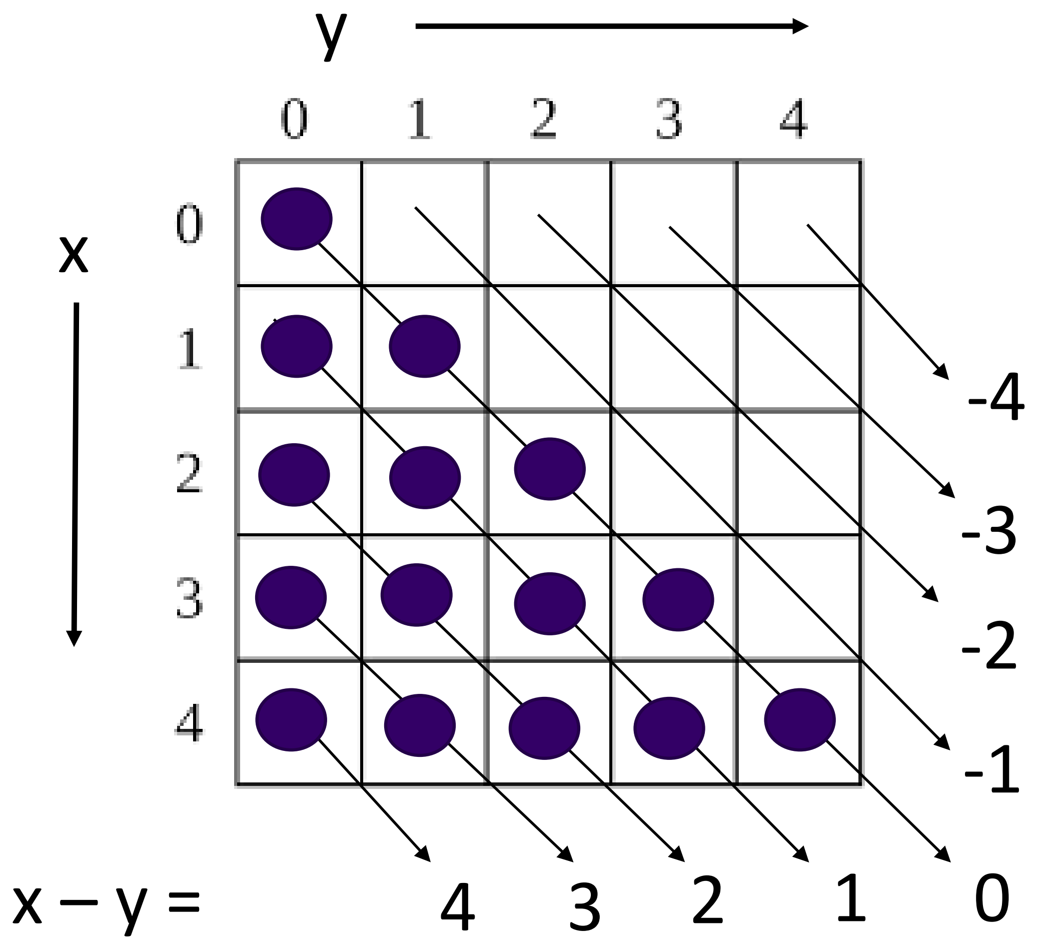




```
for(col=0; col<m[0].length; col++)
    for(row=m.length-1; row>=0; row--)
        System.out.println(m[row][col]);
```

```
for(col=m[0].length-1; col>=0; col--)
    for(row=m.length-1; row>=0; row--)
        System.out.println(m[row][col]);
```

Partial Array Traversal



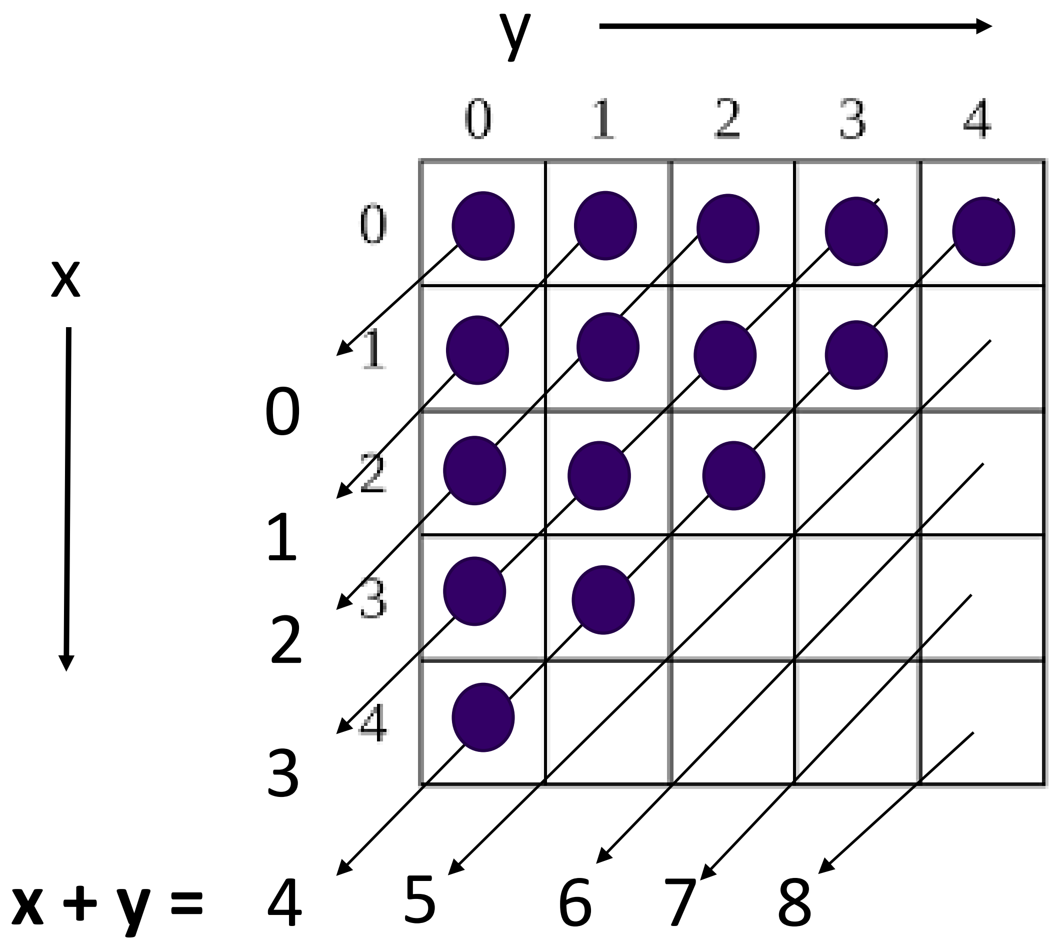
```
for (int I = 0; i<m.length; i++)
   for (int j =0; j< i +1;  j ++)
      { /* do something */}
```

Index:

Stop Condition: j stop at j = i.

i – j = 0;

Partial Array Traversal



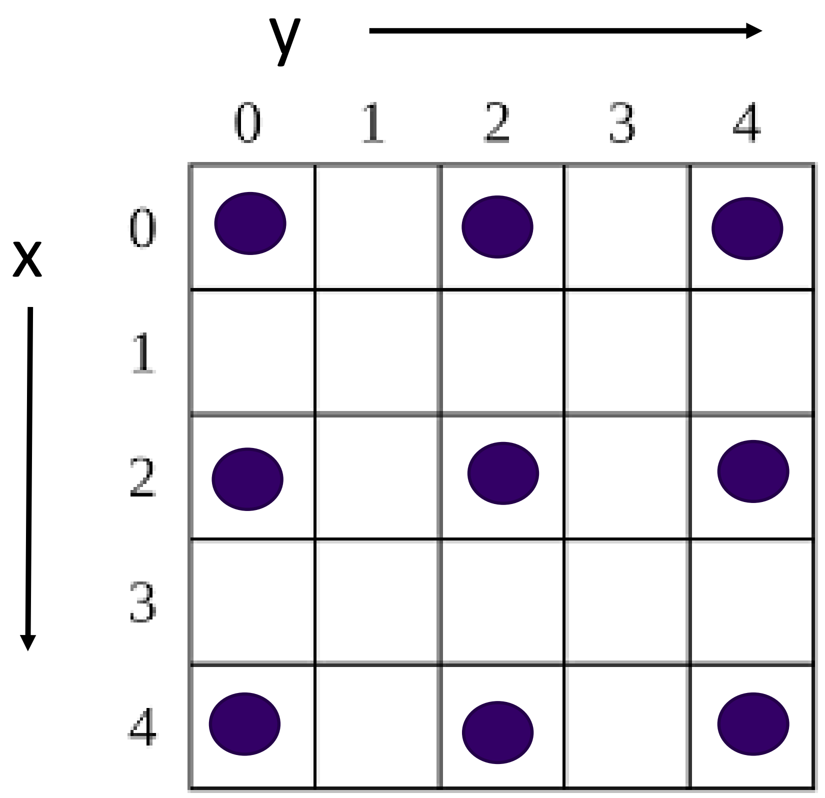
```
for (int i = 0; i<m.length; i++)
    for (int j =m.length-1-i; j>=0;  j --)
        { /* do something */}
```

Index:

Start Condition: i + j = m.length-1;

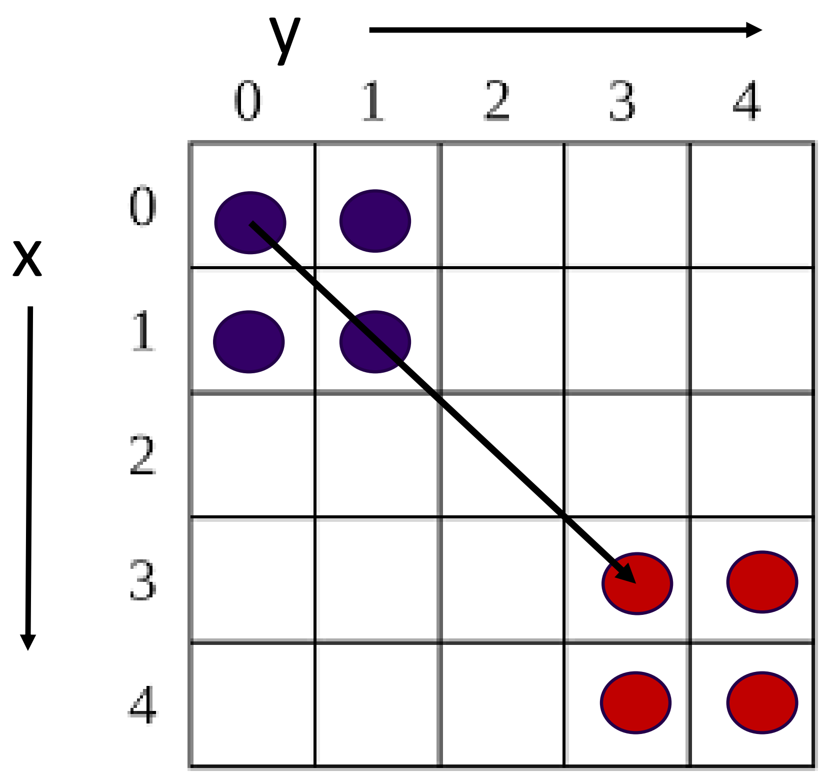
i + j = 4;

Partial Array Traversal

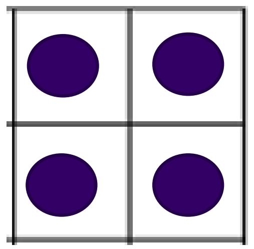


```
for (int i = 0; i<m.length; i+=2)
    for (int j =0; j<m[0].length;  j+=2)
        { /* do something */}
```

Vector Operation (Area Copy)



```
for (int i = 0; i<2; i++)
    for (int j=0; j<2; j++)
        m[3+i][3+j] = m[0+i][0+j];
             /* 0 is not needed */
```

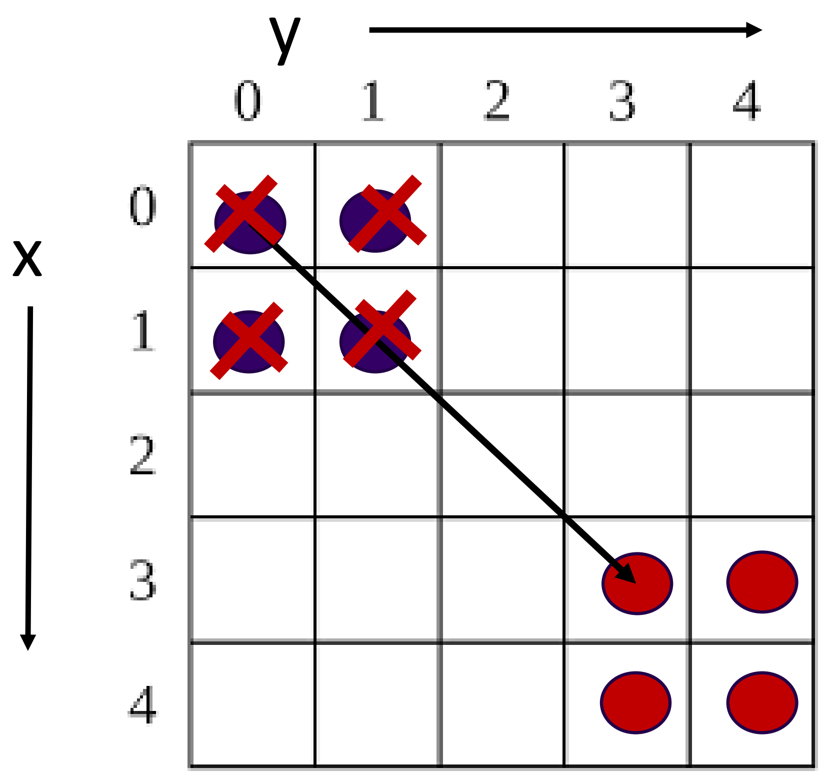


Area to be Copied: **2 x 2** block

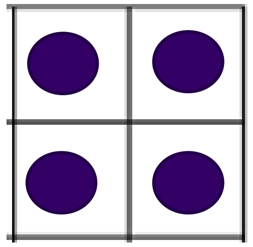
From (0, 0) to (3, 3)

If there is overlap, we need to put the area into a buffer first.

Vector Operation (Area Move)



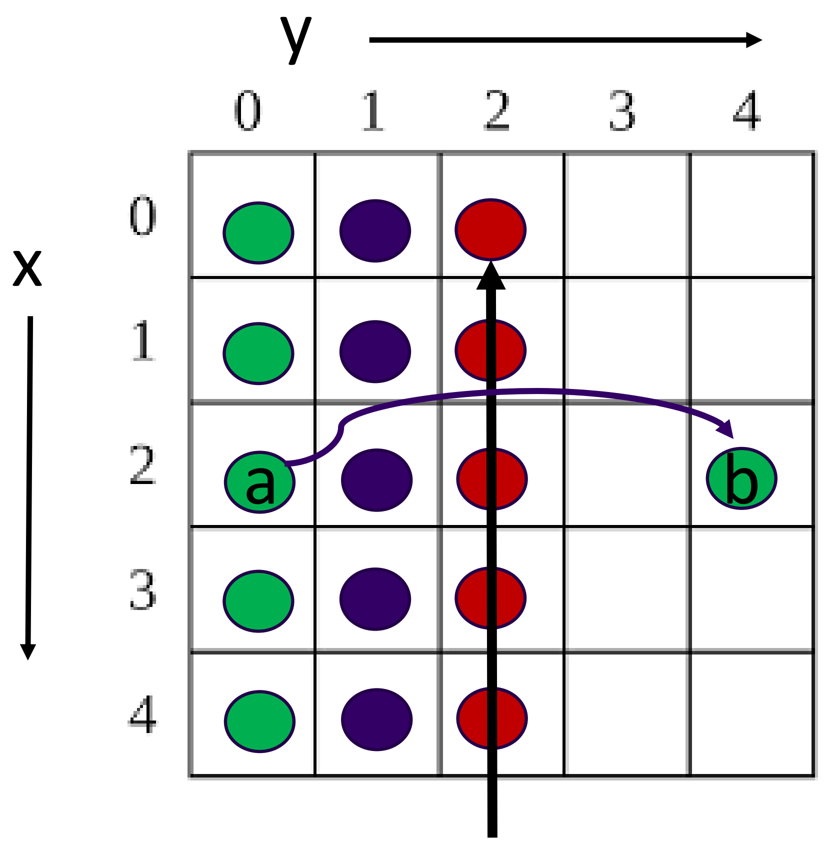
```
for (int i = 0; i<2; i++)
    for (int j=0; j<2; j++)
        m[3+i][3+j] = m[0+i][0+j];
for (int i = 0; i<2; i++)
    for (int j=0; j<2; j++)
        m[0+i][0+j]=0;
```



Area to be Copied: 2 x 2 block

From (0, 0) to (3, 3)

Flip

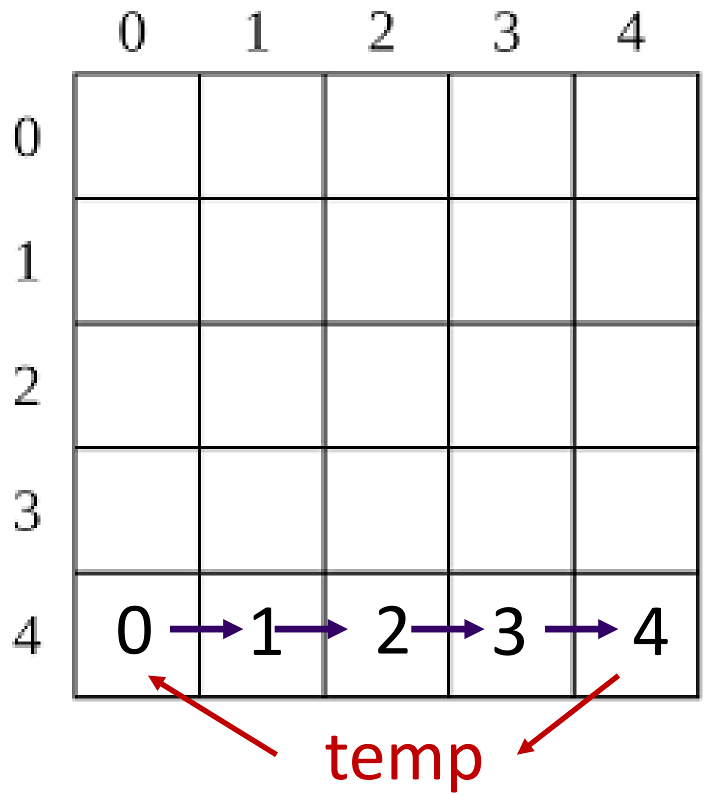


Symmetric Line for Flipping:

```
j = 2;    // j = m.length/2

(a + b)/2 = 2;
b == 4-a;    // b = m.length -a-1;

for (int i=0; i<m.length; i++){
    for (int j=0; j<m.length/2; j++){
        m[i][m.length-j-1] = m[i][j];
    }
}
```

Area Shift

```
for (int i=0; i<m.length; i++){
    int temp = m[i][m.length-1];
    for (int j=m[0].length-2; j<=0; j--){
          m[i][j+1] = m[i][j];
    }
    m[i][0] =temp;
}
```



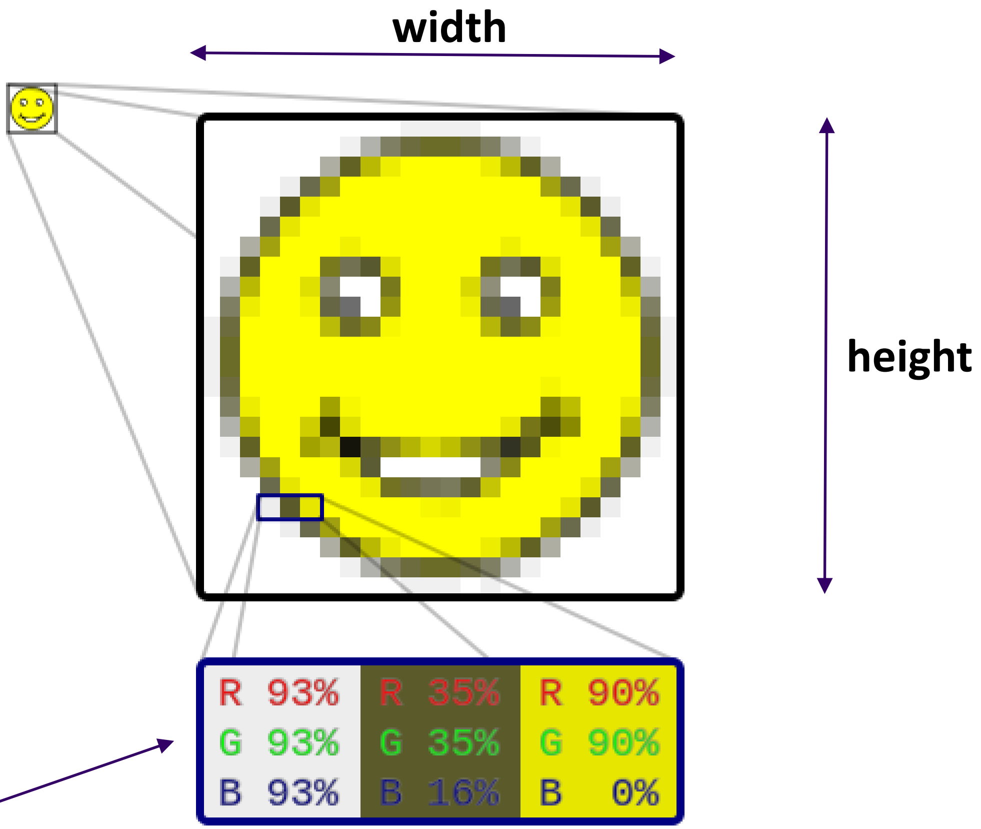
One Row after Shift: 4 0 1 2 3

2-D Array Image Processing

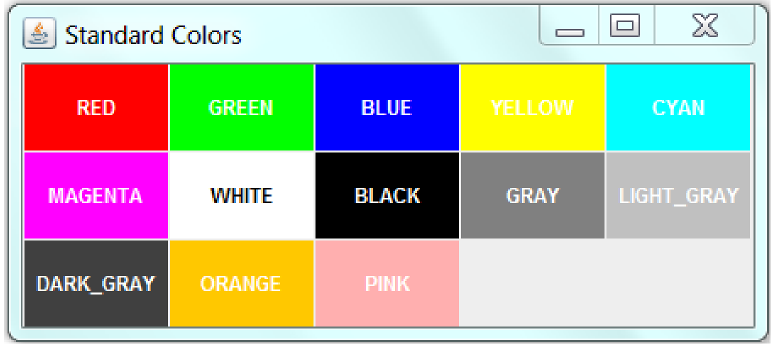
LECTURE 4

An Raster-based Image is an 2-D Array of Pixels with Color

https://en.wikipedia.org/wiki/Raster_graphics



3D-Color vector (by percentage, or by strength (0-255), if alpha channel (opacity) is also included.



java.awt.Color

- The class java.awt.Color provides 13 standard colors as named-constants. They are: Color.RED, GREEN, BLUE, MAGENTA, CYAN, YELLOW, BLACK, WHITE, GRAY, DARK_GRAY, LIGHT_GRAY, ORANGE, and PINK. (In JDK 1.1, these constant names are in lowercase, e.g., red. This violates the Java naming convention for constants. In JDK 1.2, the uppercase names are added. The lowercase names were not removed for backward compatibility.)
- You can use the toString() to print the RGB values of these color (e.g., System.out.println(Color.RED)):

Color Vector (4D, Red/Green/Blue/Alpha)

(Alpha channel is not shown here.)

```
RED            : java.awt.Color[r=255, g=0,   b=0]
GREEN          : java.awt.Color[r=0,   g=255, b=0]
BLUE           : java.awt.Color[r=0,   g=0,   b=255]
YELLOW         : java.awt.Color[r=255, g=255, b=0]
MAGENTA        : java.awt.Color[r=255, g=0,   b=255]
CYAN           : java.awt.Color[r=0,   g=255, b=255]
WHITE          : java.awt.Color[r=255, g=255, b=255]
BLACK          : java.awt.Color[r=0,   g=0,   b=0]
GRAY           : java.awt.Color[r=128, g=128, b=128]
LIGHT_GRAY     : java.awt.Color[r=192, g=192, b=192]
DARK_GRAY      : java.awt.Color[r=64,  g=64,  b=64]
PINK           : java.awt.Color[r=255, g=175, b=175]
ORANGE         : java.awt.Color[r=255, g=200, b=0]
```

Gray Level Image
(R=G=B, all three channels have the same strength)

- A **grayscale** (or graylevel) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a `gray' color is one in which the red, green and blue components all have **equal intensity** in **RGB space**, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image.

- Often, the grayscale intensity is stored as an **8-bit** integer giving **256** possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive graylevels is significantly better than the graylevel resolving power of the human eye.

Demo Program: Graylevel.zip

(include Picture.java Luminance.java, ShowColor.java, ColorToGray.java, Graylevel.java)

Picture.java: handling basic image functions. Do not worry about it right now.

Luminance.java: convert a color into a gray level color (still color but with same intensity for all three RGB channels.)

ShowColor.java: show a color image potentialColor.png

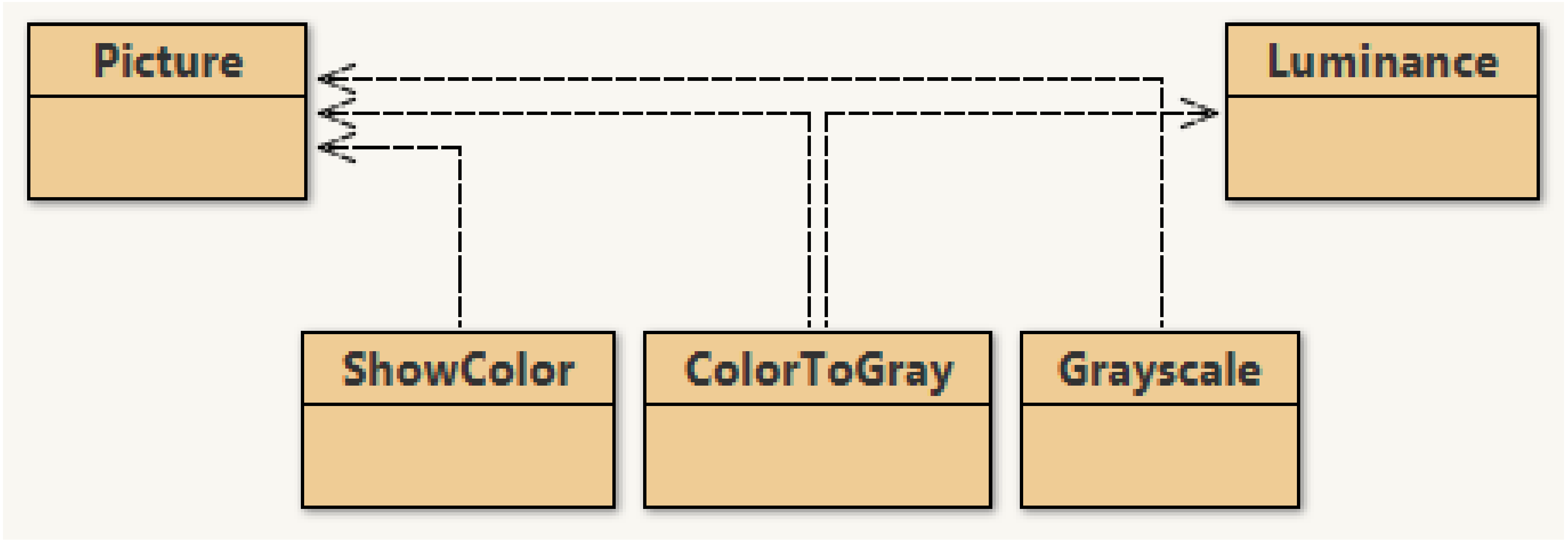
ColorToGray.java: convert the color image to gray level image.

Graylevel.java: our program of interests.

adjust the brightness level of an image by increase the brightness level

or darken it by adding a negative brightness level.

The potentialColor.png and potentialGray.png both are images size of 250 by 250 pixels.


Picture
Luminance
ShowColor
ColorToGray
Grayscale

Demonstration Program

PICTURE.JAVA +LUMINANCE.JAVA

SHOWCOLOR.JAVA

COLORTOGRAY.JAVA

GRAYSCALE.JAVA

Passing 2D Arrays to Methods

LECTURE 5

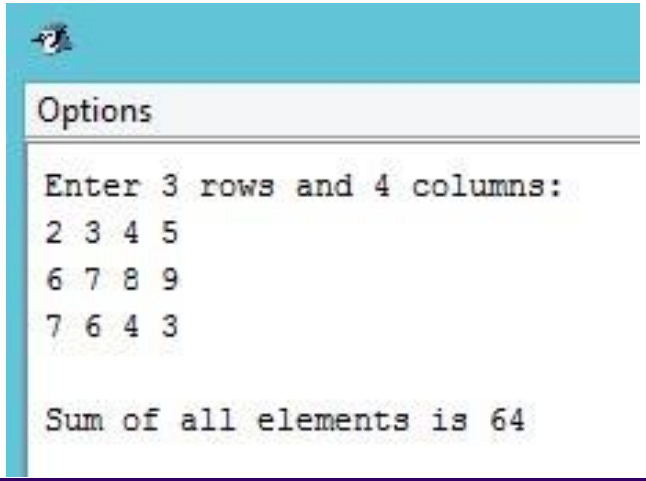
Passing Two-Dimensional Arrays to Methods

- You pass a two-dimensional array to a method just as you pass a one-dimensional array. You also return an array from a method. **Pass2DArray.java** program gives an example with two methods. The first method, getArray(), returns a two-dimensional array, and the second method, sum(int[][] m), returns the sum of all elements in a matrix.

Go BlueJ …

Passing Two-Dimensional Arrays to Methods

- The method getArray prompts the user to enter values for the array (lines 11-24) and returns the array (line 23).
- The method sum (lines 26-35) has a two-dimensional array argument. You can obtain the number of rows using m.length (line 28) and the number of columns in a specified row using m[row].length (line 29).



```
Enter 3 rows and 4 columns:
2 3 4 5
6 7 8 9
7 6 4 3

Sum of all elements is 64
```

Demonstration Program

PASS2DARRAY.JAVA

Application Grading Multiple-Choice Tests

LECTURE 6

Problem: Grading Multiple-Choice Test

StudentAnswer.java (Grading for a class on a subject)

Students' Answers to the Questions:

	0	1	2	3	4	5	6	7	8	9
Student 0	A	B	A	C	C	D	E	E	A	D
Student 1	D	B	A	B	C	A	E	E	A	D
Student 2	E	D	D	A	C	B	E	E	A	D
Student 3	C	B	A	E	D	C	E	E	A	D
Student 4	A	B	D	C	C	D	E	E	A	D
Student 5	B	B	E	C	C	D	E	E	A	D
Student 6	B	B	A	C	C	D	E	E	A	D
Student 7	E	B	E	C	C	D	E	E	A	D

- Objective: write a program that grades multiple-choice test.

Key to the Questions:

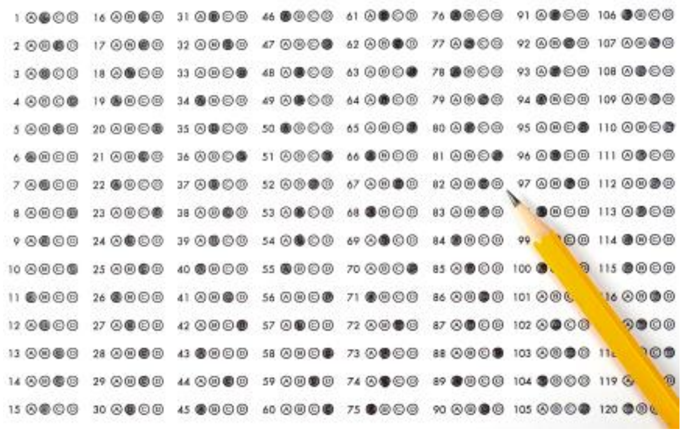
	0	1	2	3	4	5	6	7	8	9
Key	D	B	D	C	C	D	A	E	A	D

Demonstration Program

STUDENTANSWER.JAVA

Ragged Array for Student's Score for Multiple Subjects

StudentScoreMultiple.java



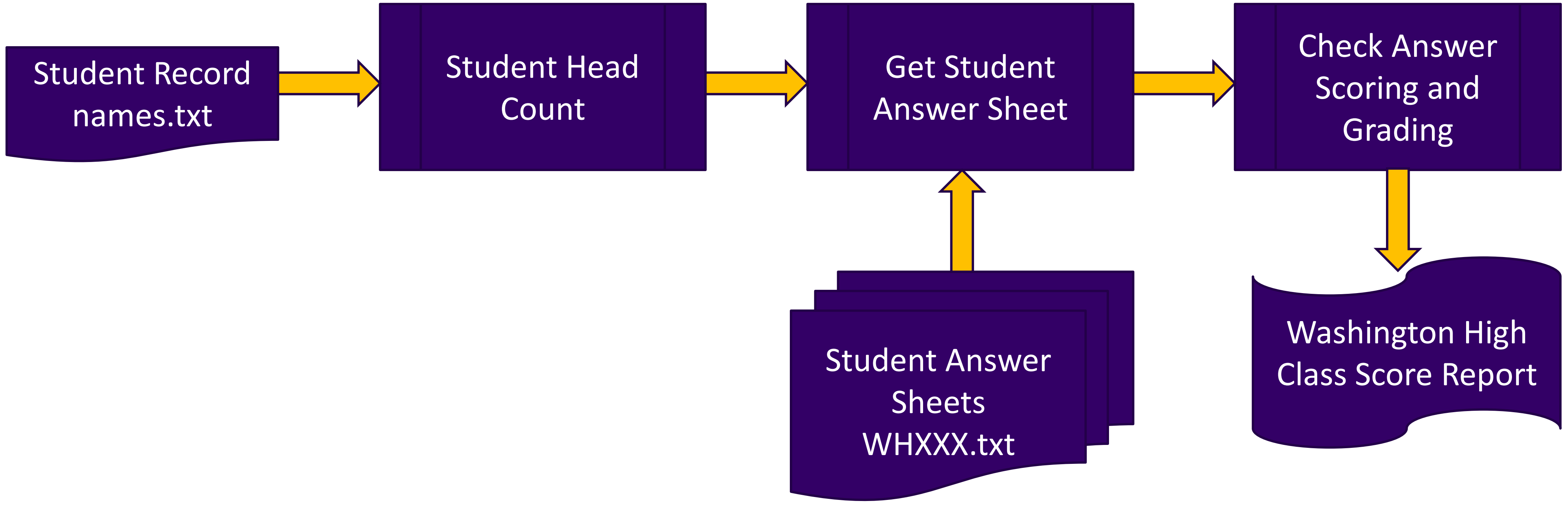
Student Answer Sheet

Answer Sheet Scanning Reader

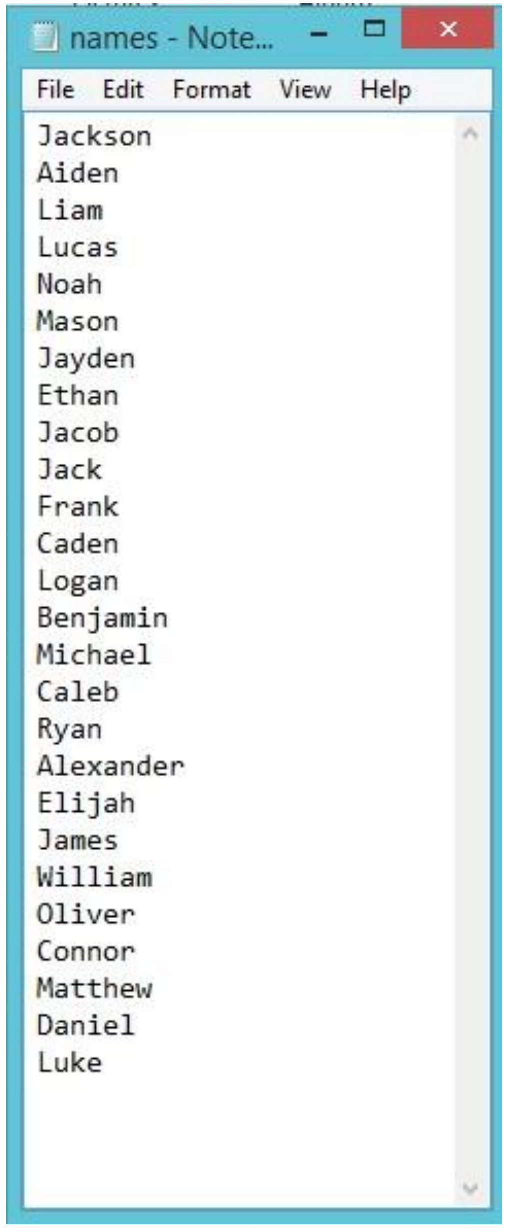
Student Answer Sheet Text File
WH000.txt

Data Flow Diagram

StudentScoreMultiple.java


Student Record names.txt
Student Head Count
Get Student Answer Sheet
Check Answer Scoring and Grading
Student Answer Sheets WHXXX.txt
Washington High Class Score Report

Student Record and Answer Sheet

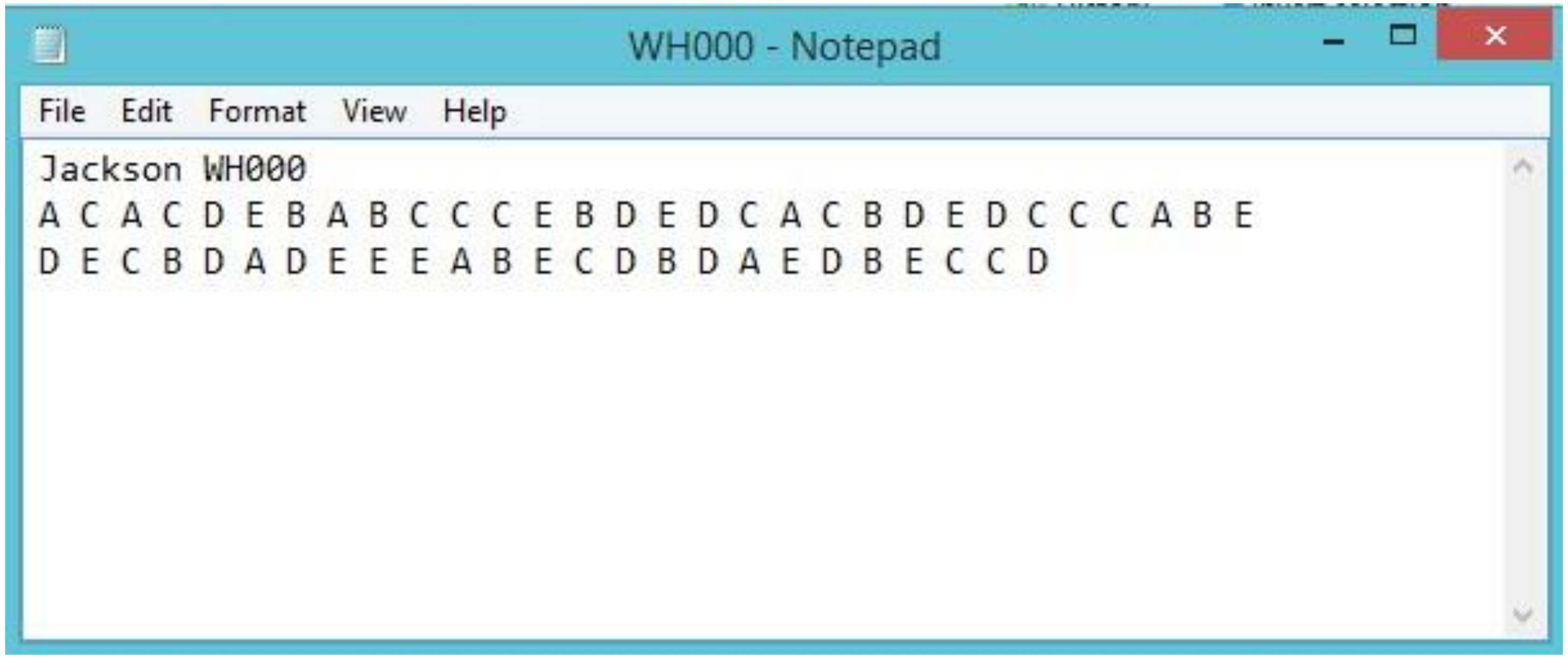


names - Note...

File Edit Format View Help

```
Jackson
Aiden
Liam
Lucas
Noah
Mason
Jayden
Ethan
Jacob
Jack
Frank
Caden
Logan
Benjamin
Michael
Caleb
Ryan
Alexander
Elijah
James
William
Oliver
Connor
Matthew
Daniel
Luke
```

Student Record names.txt



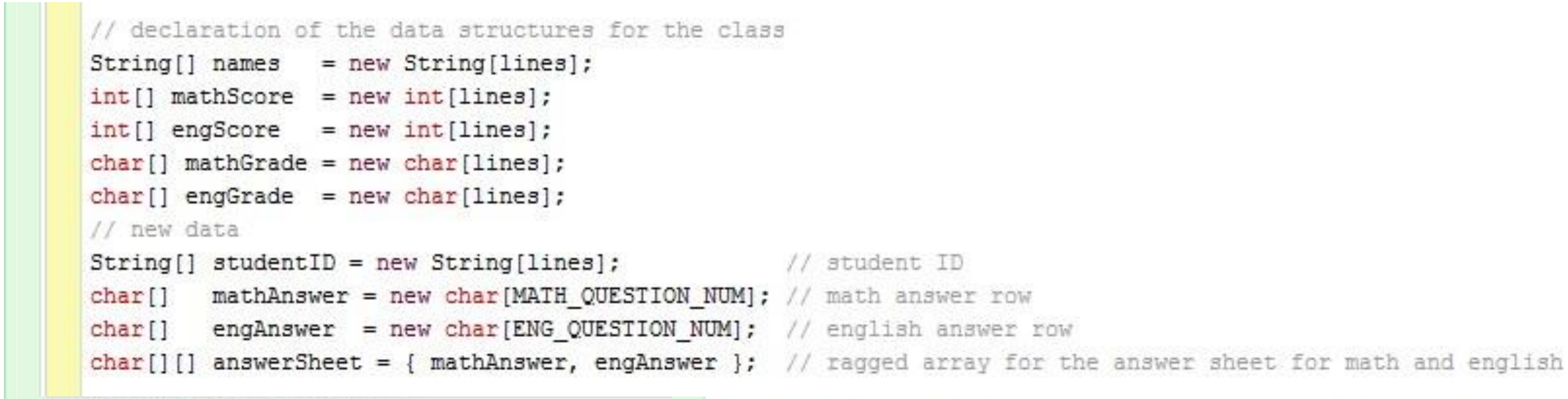
WH000 - Notepad

File Edit Format View Help

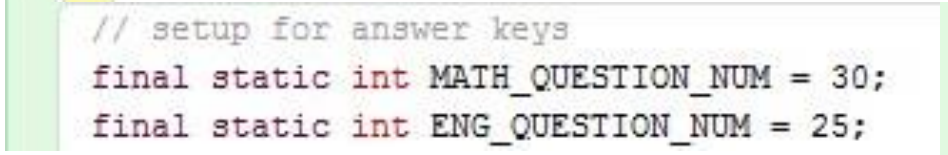
```
Jackson WH000
A C A C D E B A B C C C E B D E D C A C B D E D C C C A B E
D E C B D A D E E E A B E C D B D A E D B E C C D
```

Answer Sheet WH000.txt

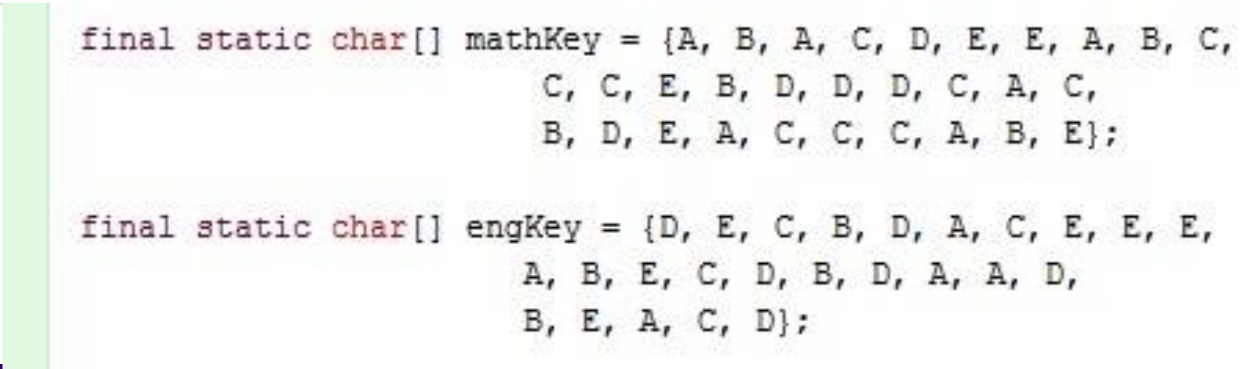
Data Structure for Student Records



```
// declaration of the data structures for the class
String[] names    = new String[lines];
int[] mathScore   = new int[lines];
int[] engScore    = new int[lines];
char[] mathGrade = new char[lines];
char[] engGrade   = new char[lines];
// new data
String[] studentID = new String[lines];             // student ID
char[]   mathAnswer = new char[MATH_QUESTION_NUM]; // math answer row
char[]   engAnswer  = new char[ENG_QUESTION_NUM];  // english answer row
char[][] answerSheet = { mathAnswer, engAnswer };  // ragged array for the answer sheet for math and english
```



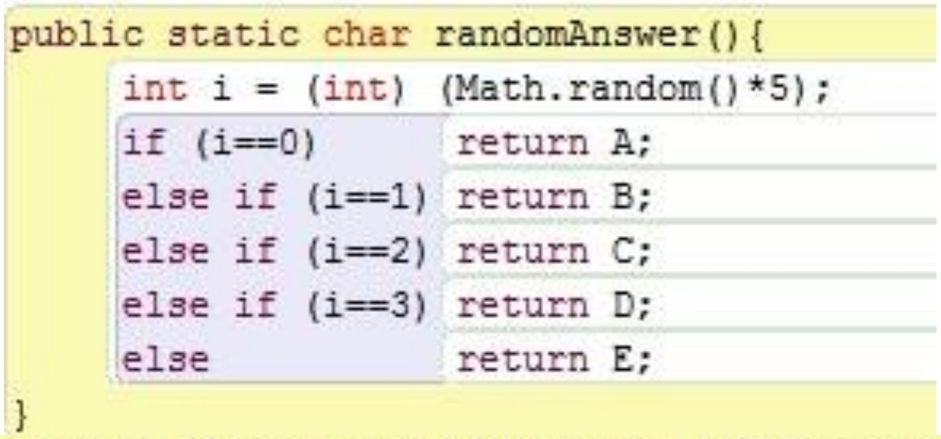
```
// setup for answer keys
final static int MATH_QUESTION_NUM = 30;
final static int ENG_QUESTION_NUM = 25;
```



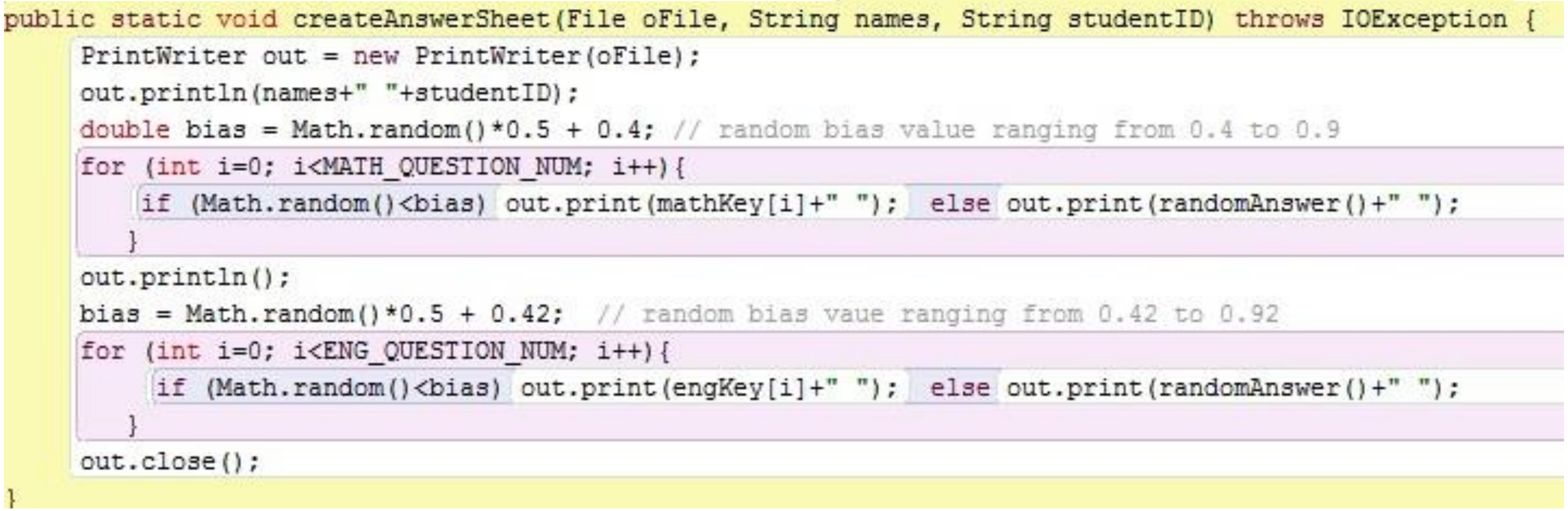
```
final static char[] mathKey = {A, B, A, C, D, E, E, A, B, C,
                               C, C, E, B, D, D, D, C, A, C,
                               B, D, E, A, C, C, C, A, B, E};

final static char[] engKey = {D, E, C, B, D, A, C, E, E, E,
                              A, B, E, C, D, B, D, A, A, D,
                              B, E, A, C, D};
```

Random Answer Sheet

(for program testing purpose)



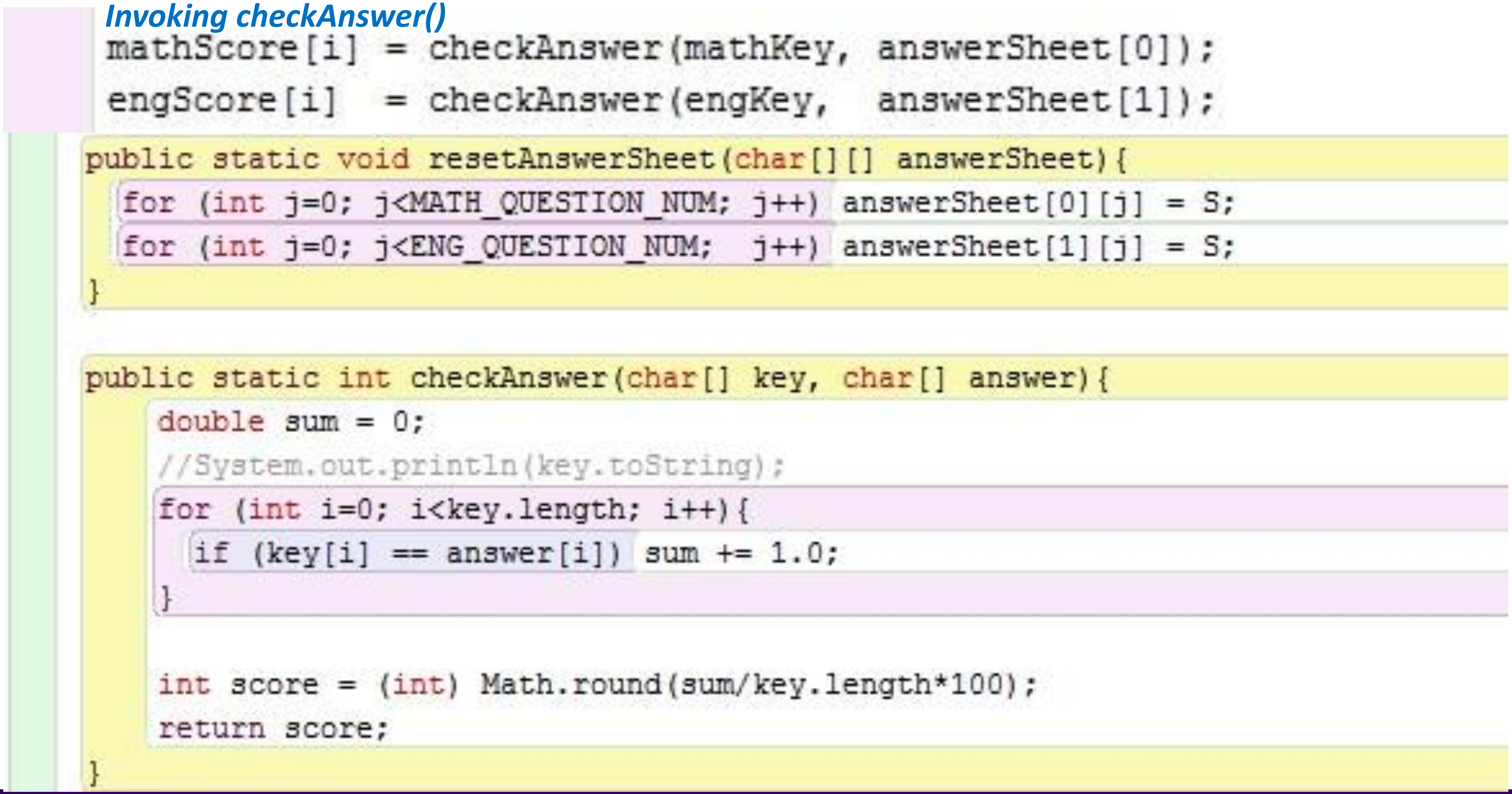
```
public static char randomAnswer(){
    int i = (int) (Math.random()*5);
    if (i==0)          return A;
    else if (i==1)     return B;
    else if (i==2)     return C;
    else if (i==3)     return D;
    else               return E;
}
```



```
public static void createAnswerSheet(File oFile, String names, String studentID) throws IOException {
    PrintWriter out = new PrintWriter(oFile);
    out.println(names+" "+studentID);
    double bias = Math.random()*0.5 + 0.4; // random bias value ranging from 0.4 to 0.9
    for (int i=0; i<MATH_QUESTION_NUM; i++){
        if (Math.random()<bias) out.print(mathKey[i]+" ");  else out.print(randomAnswer()+" ");
    }
    out.println();
    bias = Math.random()*0.5 + 0.42;  // random bias vaue ranging from 0.42 to 0.92
    for (int i=0; i<ENG_QUESTION_NUM; i++){
        if (Math.random()<bias) out.print(engKey[i]+" ");  else out.print(randomAnswer()+" ");
    }
    out.close();
}
```

Learning Channel

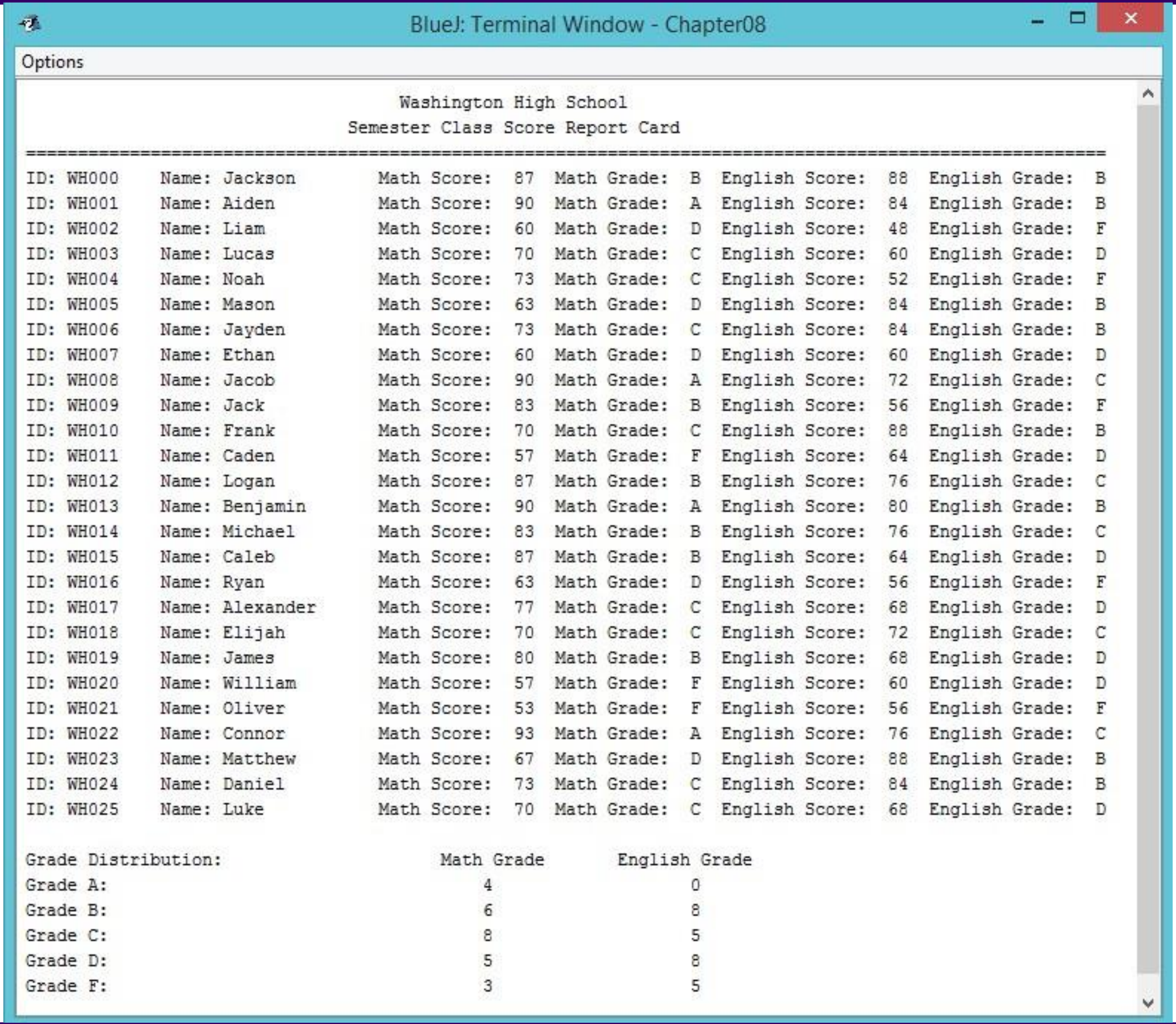
resetAnswerSheet() and checkAnswerSheet)



Invoking checkAnswer()

```
mathScore[i] = checkAnswer(mathKey, answerSheet[0]);
engScore[i]  = checkAnswer(engKey,  answerSheet[1]);
```

```
public static void resetAnswerSheet(char[][] answerSheet){
  for (int j=0; j<MATH_QUESTION_NUM; j++) answerSheet[0][j] = S;
  for (int j=0; j<ENG_QUESTION_NUM;  j++) answerSheet[1][j] = S;
}

public static int checkAnswer(char[] key, char[] answer){
    double sum = 0;
    //System.out.println(key.toString);
    for (int i=0; i<key.length; i++){
      if (key[i] == answer[i]) sum += 1.0;
    }

    int score = (int) Math.round(sum/key.length*100);
    return score;
}
```



```
                                     Washington High School
                                Semester Class Score Report Card
==========================================================================================================
ID: WH000    Name: Jackson        Math Score:  87  Math Grade:  B  English Score:  88  English Grade:  B
ID: WH001    Name: Aiden          Math Score:  90  Math Grade:  A  English Score:  84  English Grade:  B
ID: WH002    Name: Liam           Math Score:  60  Math Grade:  D  English Score:  48  English Grade:  F
ID: WH003    Name: Lucas          Math Score:  70  Math Grade:  C  English Score:  60  English Grade:  D
ID: WH004    Name: Noah           Math Score:  73  Math Grade:  C  English Score:  52  English Grade:  F
ID: WH005    Name: Mason          Math Score:  63  Math Grade:  D  English Score:  84  English Grade:  B
ID: WH006    Name: Jayden         Math Score:  73  Math Grade:  C  English Score:  84  English Grade:  B
ID: WH007    Name: Ethan          Math Score:  60  Math Grade:  D  English Score:  60  English Grade:  D
ID: WH008    Name: Jacob          Math Score:  90  Math Grade:  A  English Score:  72  English Grade:  C
ID: WH009    Name: Jack           Math Score:  83  Math Grade:  B  English Score:  56  English Grade:  F
ID: WH010    Name: Frank          Math Score:  70  Math Grade:  C  English Score:  88  English Grade:  B
ID: WH011    Name: Caden          Math Score:  57  Math Grade:  F  English Score:  64  English Grade:  D
ID: WH012    Name: Logan          Math Score:  87  Math Grade:  B  English Score:  76  English Grade:  C
ID: WH013    Name: Benjamin       Math Score:  90  Math Grade:  A  English Score:  80  English Grade:  B
ID: WH014    Name: Michael        Math Score:  83  Math Grade:  B  English Score:  76  English Grade:  C
ID: WH015    Name: Caleb          Math Score:  87  Math Grade:  B  English Score:  64  English Grade:  D
ID: WH016    Name: Ryan           Math Score:  63  Math Grade:  D  English Score:  56  English Grade:  F
ID: WH017    Name: Alexander      Math Score:  77  Math Grade:  C  English Score:  68  English Grade:  D
ID: WH018    Name: Elijah         Math Score:  70  Math Grade:  C  English Score:  72  English Grade:  C
ID: WH019    Name: James          Math Score:  80  Math Grade:  B  English Score:  68  English Grade:  D
ID: WH020    Name: William        Math Score:  57  Math Grade:  F  English Score:  60  English Grade:  D
ID: WH021    Name: Oliver         Math Score:  53  Math Grade:  F  English Score:  56  English Grade:  F
ID: WH022    Name: Connor         Math Score:  93  Math Grade:  A  English Score:  76  English Grade:  C
ID: WH023    Name: Matthew        Math Score:  67  Math Grade:  D  English Score:  88  English Grade:  B
ID: WH024    Name: Daniel         Math Score:  73  Math Grade:  C  English Score:  84  English Grade:  B
ID: WH025    Name: Luke           Math Score:  70  Math Grade:  C  English Score:  68  English Grade:  D

Grade Distribution:                     Math Grade       English Grade
Grade A:                                    4                   0
Grade B:                                    6                   8
Grade C:                                    8                   5
Grade D:                                    5                   8
Grade F:                                    3                   5
```

Demonstration Program

STUDENTSCOREMULTIPLE.JAVA

Demo Program:

Finding the Closest Pair of Points

LECTURE 7

Problem: Finding Two Points Nearest to Each Other (FindingClosestPair.java)

```
double[][] points = { {  -1, 0, 3}, { -1, -1,  -1},
                       {   4, 1, 1}, {  2,0.5,   9},
                       { 3.5, 2,-1}, {  3, 1.5,   3},
                       {-1.5, 4, 2}, {5.5,  4,-0,5}
                    };
```

- 3-D points in 2-D array. Each row is a 3-tuple 3-D points.

Distance of 2 3D points

- Row array points[i] is actually a point of 3 tuple. points[0]= { -1, 0, 3};

which represents a point (x, y, z) in 3D space.

- So, the distance of Two 3D points can be calculated by:

```
distance=Math.pow((p1[0]-p2[0])*(p1[0]-p2[0])
                 +(p1[1]-p2[1])*(p1[1]-p2[1])
                 +(p1[2]-p2[2])*(p1[2]-p2[2]), 0.5);
```

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

8 points has totally 28 line segments to compare for the closest pair.

	p[1]	p[2]	p[3]	p[4]	p[5]	p[6]	p[7]
p[0]	d	d	d	d	d	d	d
p[1]		d	d	d	d	d	d
p[2]			d	d	d	d	d
p[3]				d	d	d	d
p[4]					d	d	d
p[5]						d	d
p[6]							d

Find the minimum in these pair of points. Using nested loop of 2D array:

```
 for (int i=0; i<7; i++){ // pseudo code
        for (int j=i+1; j<8; j++){
             if (distance < min) min = distance;
}} // pseudo code.
```

Demonstration Program

FINDINGCLOSESTPAIR.JAVA

Lab:

CombinationNumber.java

LECTURE 8

Purpose of this project

(1) Use 2-D array for a real-world calculation application.

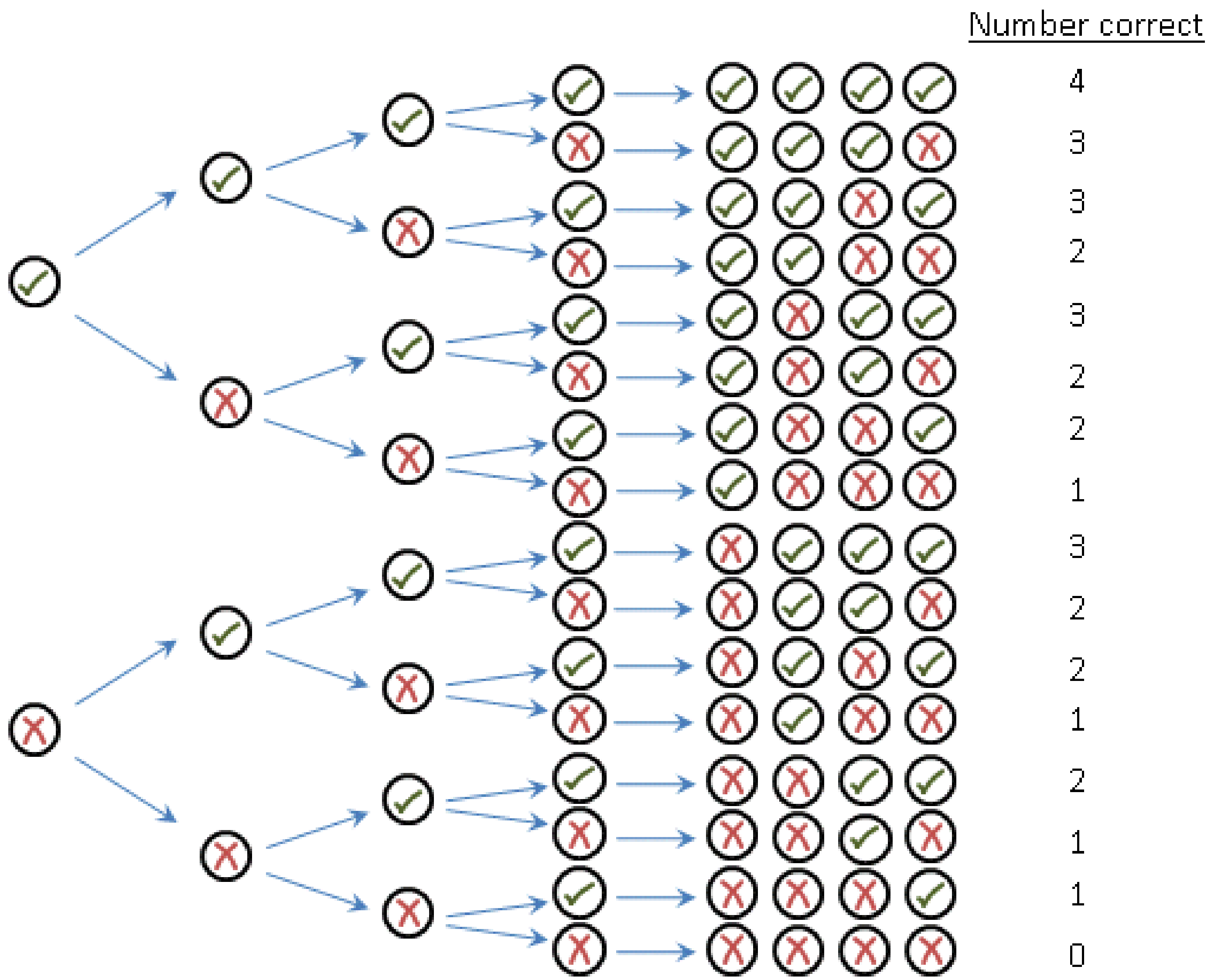
(2) Exercise on the 2D **index calculation** using **mapping**.

(3) 2-D Array can also work like a method (function)

(4) The term **Wrapper Function**.

Wrap a function with easier interface or user-friendly interface. Or, providing extra information each time wrapper method is called.

Binomial Experiments
(Coin Tossing/Yes No Question Guessing)



Lab Project: Generation of a Combination Number

- In mathematics, C_m^n denotes the number of different ways that m things can be selected from n different choices. For example, if you are choosing among six desserts and are allowed to take two, the number of different combinations you could choose is C_2^6 . Here's one formula to compute this value: $$C_m^n = \frac{n!}{m!(n-m)!}$$
- This value also gives rise to an interesting recursion:

$$C_m^n = C_{m-1}^{n-1} + C_m^{n-1}$$

- **Write an iterative function to compute combinations.** Hints: when $m=1$, C_m^n = n and when $n < m$, C_m^n = 0;

Binomial Theorem

$$\begin{aligned}
(a+b)^0 &= 1\\
(a+b)^1 &= a+b\\
(a+b)^2 &= a^2+2ab+b^2\\
(a+b)^3 &= a^3+3a^2b+3ab^2+b^3\\
(a+b)^4 &= a^4+4a^3b+6a^2b^2+4ab^3+b^4\\
(a+b)^5 &= a^5+5a^4b+10a^3b^2+10a^2b^3+5ab^4+b^5
\end{aligned}$$

Pascal Triangle

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

Combination Number:

$$C_m^n = \frac{n!}{m!(n-m)!}$$

Binomial Theorem

(1) Recursive Formula

$$C_m^n = C_{m-1}^{n-1} + C_m^{n-1}$$

$$\begin{tikzcd}
& & & & & C_0^0 \arrow[dl] \arrow[dr] & & & & & \\
& & & & C_1^1 \arrow[dl] \arrow[dr] & & C_0^1 \arrow[dl] \arrow[dr] & & & & \\
& & & C_2^2 \arrow[dr] & & C_1^2 \arrow[dl] \arrow[dr] & & C_0^2 \arrow[dl] & & & \\
& & C_3^3 & & C_2^3 \arrow[dr] & & C_1^3 \arrow[dl] & & C_0^3 & & \\
& C_4^4 & & C_3^4 & & C_2^4 & & C_1^4 & & C_0^4 & \\
C_5^5 & & C_4^5 & & C_3^5 & & C_2^5 & & C_1^5 & & C_0^5
\end{tikzcd}$$

Binomial Theorem

Mapping C to K array

Combination Number:

$$C_m^n = \frac{n!}{m!(n-m)!}$$

(1) Recursive Formula

$$C_m^n = C_{m-1}^{n-1} + C_m^{n-1}$$

(2) Mapping Rule

$$C_m^n = K_m^{n-m}$$

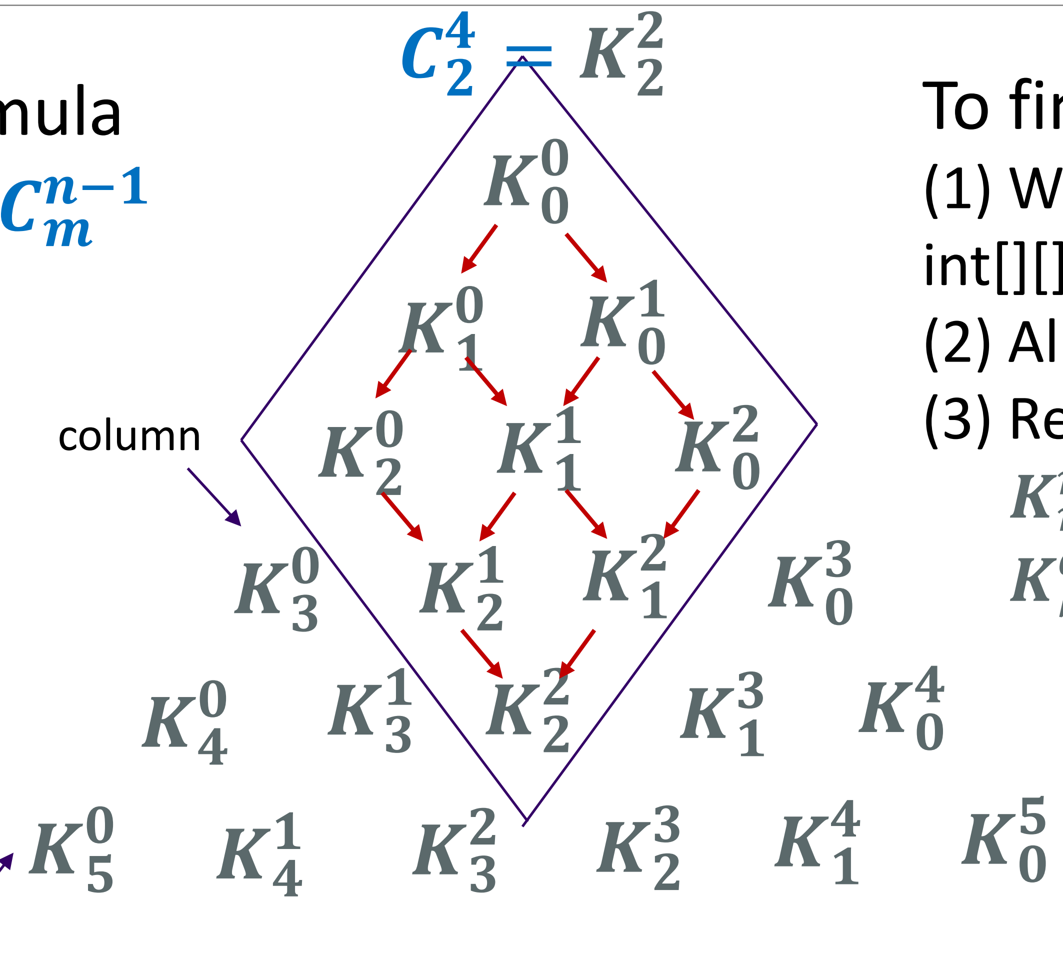
$$C_b^{a+b} = K_b^a$$

a = n-m

b = m

(a+b) = n

b = m



$C_2^4 = K_2^2$
column
Row
K_0^0
K_1^0 K_0^1
K_2^0 K_1^1 K_0^2
K_3^0 K_2^1 K_1^2 K_0^3
K_4^0 K_3^1 K_2^2 K_1^3 K_0^4
K_5^0 K_4^1 K_3^2 K_2^3 K_1^4 K_0^5

To find C_m^n :

(1) We need to create 2D Array:

int[][] k = new int[n-m+1][m+1];

(2) All row 0 and col 0 are 1.

(3) Recursive formula for K:

$$K_m^{n-m} = K_{m-1}^{n-m} + K_m^{n-m-1}$$

$$K_b^a = K_{b-1}^a + K_b^{a-1}$$

Pseudo Code

2D array is also a method (function)

```
public static c(int n, int m){ // also called as wrapper function.
    Create k array of n-m+1 row and m+1 column.
    set row 0 of array k to 1.
    set col 0 of array k to 1.
    set a = n-m; b = m;
    for (int i=0; i<= a; i++)
        for (int j=0; j<= b; j++)
                k[i][j] = k[i-1][j] + k[i][j-1];
     return k[a][b];
}
```

Other Wrapper Functions:

```
public static htmlTagWrapper(String tag, String source){
    return “<”+tag+”> ”+source+” </”+tag+”>”;
}
```

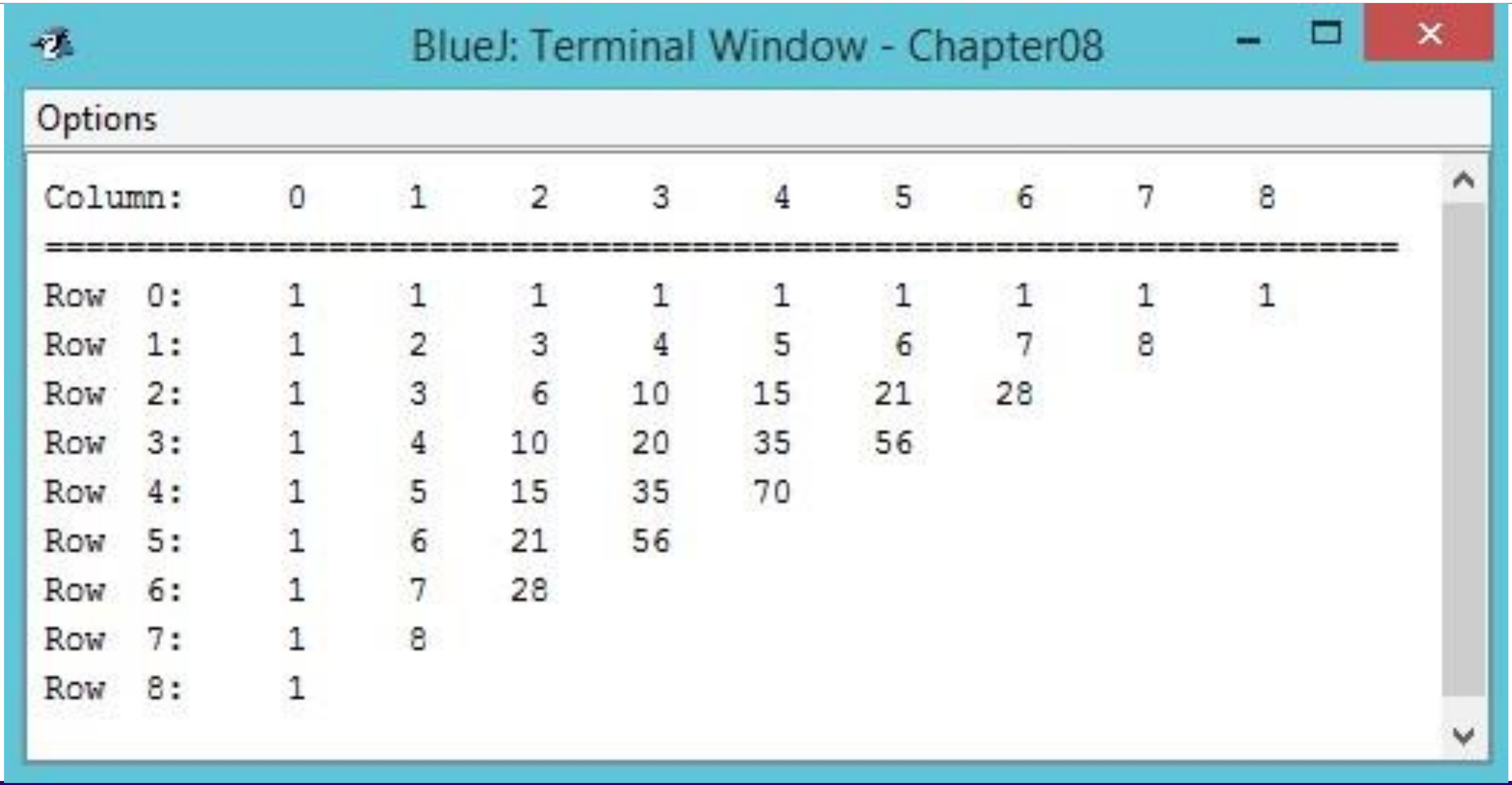
Example:

```
htmlTagWrapper(“p”, “This is a paragraph in HTML. ”);
```

Output:

```
<p> This is a paragraph in HTML. </p>
```

Expected Result:



```
Column:      0    1    2    3    4    5    6    7    8
=========================================================
Row   0:     1    1    1    1    1    1    1    1    1
Row   1:     1    2    3    4    5    6    7    8
Row   2:     1    3    6   10   15   21   28
Row   3:     1    4   10   20   35   56
Row   4:     1    5   15   35   70
Row   5:     1    6   21   56
Row   6:     1    7   28
Row   7:     1    8
Row   8:     1
```

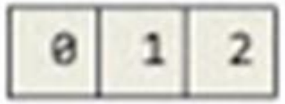
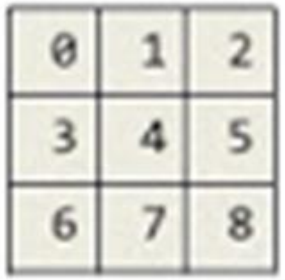
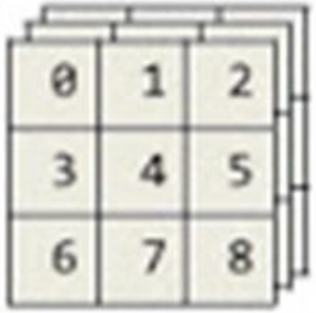
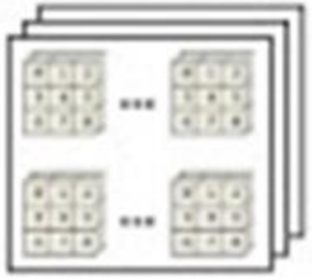
Lab

COMBINATIONNUMBER.JAVA

N-D Arrays

LECTURE 9

What is an array?

Dimensions	Example	Terminology
1		Vector
2		Matrix
3		3D Array (3rd order Tensor)
N		ND Array

Multidimensional Arrays

A two-dimensional array consists of one-dimensional arrays and a three-dimensional array consists of two-dimensional arrays

- Occasionally, you will need to represent n-dimensional data structures. In Java, you can create n-dimensional arrays for any integer `n`.
- The way to declare two-dimensional array variables and create two-dimensional arrays can be generalized to declare n-dimensional array variables and create n-dimensional arrays for `n>=3`.

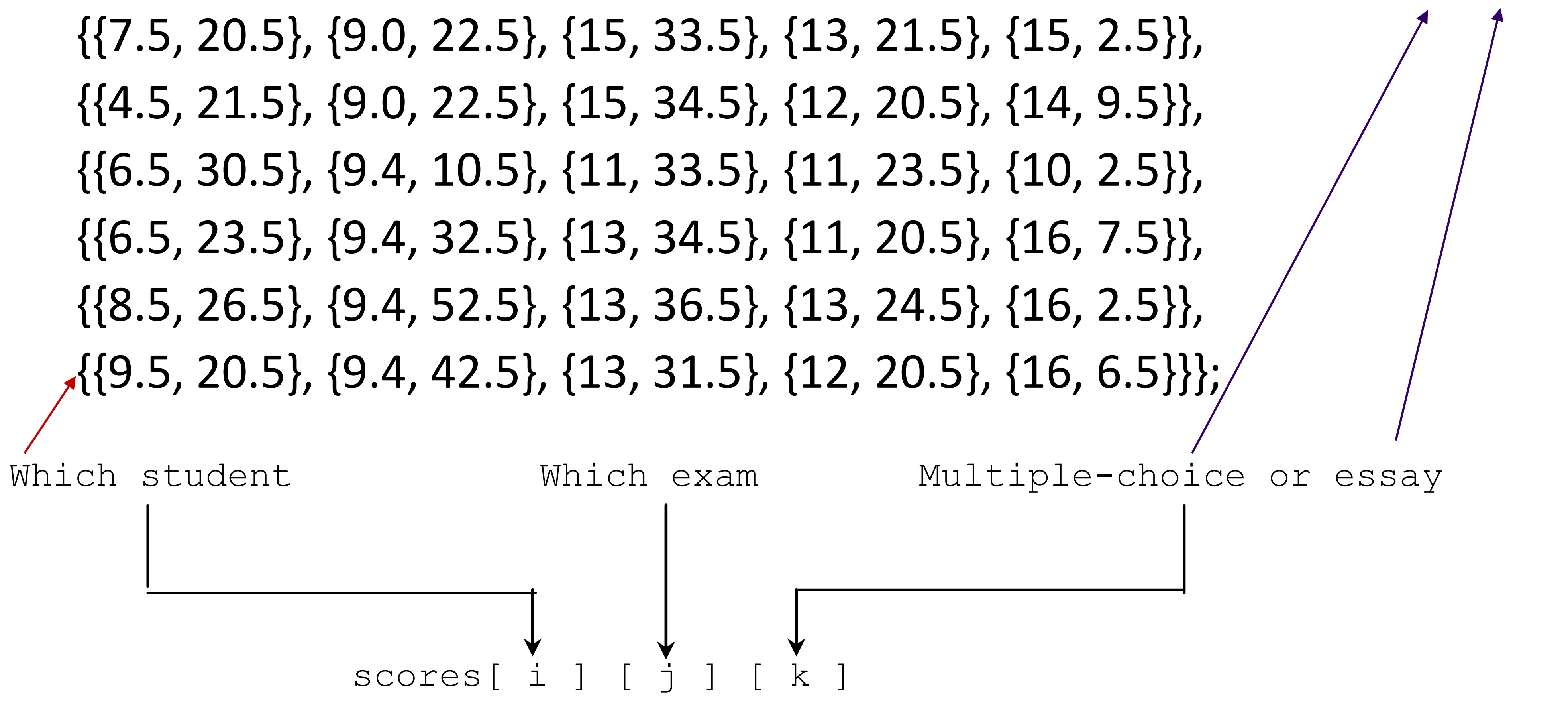
Example: Calculating Total Scores

- **Objective:** write a program that calculates the total score for students in a class. Suppose the scores are stored in a three-dimensional array named scores. The first index in scores refers to a **student**, the second refers to an **exam**, and the third refers **to the part of the exam**.
- Suppose there are 7 students, 5 exams, and each exam has two parts--the multiple-choice part and the programming part. So, scores[i][j][0] represents the score on the multiple-choice part for the i's student on the j's exam. Your program displays the total score for each student.

Multidimensional Arrays

```
double[][][] scores = {
  {{7.5, 20.5}, {9.0, 22.5}, {15, 33.5}, {13, 21.5}, {15, 2.5}},
  {{4.5, 21.5}, {9.0, 22.5}, {15, 34.5}, {12, 20.5}, {14, 9.5}},
  {{6.5, 30.5}, {9.4, 10.5}, {11, 33.5}, {11, 23.5}, {10, 2.5}},
  {{6.5, 23.5}, {9.4, 32.5}, {13, 34.5}, {11, 20.5}, {16, 7.5}},
  {{8.5, 26.5}, {9.4, 52.5}, {13, 36.5}, {13, 24.5}, {16, 2.5}},
  {{9.5, 20.5}, {9.4, 42.5}, {13, 31.5}, {12, 20.5}, {16, 6.5}}};
```

An exam: {9.0, 22.5}



Demo Project: Weather Model

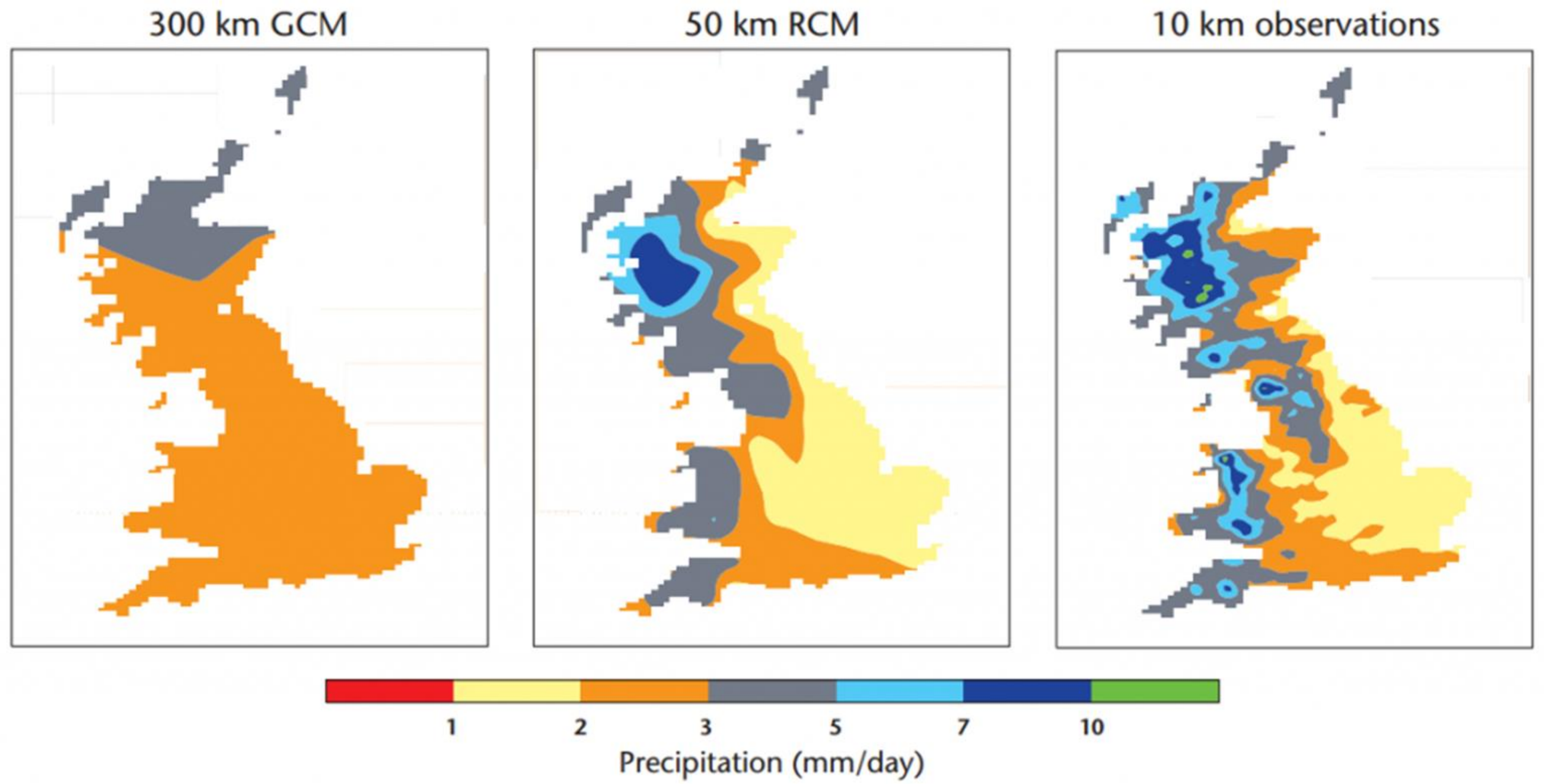
Global Climate Models:

Climate models divide the surface of the Earth into a horizontal grid, the atmosphere into vertical levels, and time into discrete timesteps.

```
GCM(x, y, h, day, hour, temperature, humidity)
```

Single_Point_CM(day, hour, temperature, humidity) // in Weather.txt

Global Climate Model (GCM) Versus Regional Climate Model (RCM)

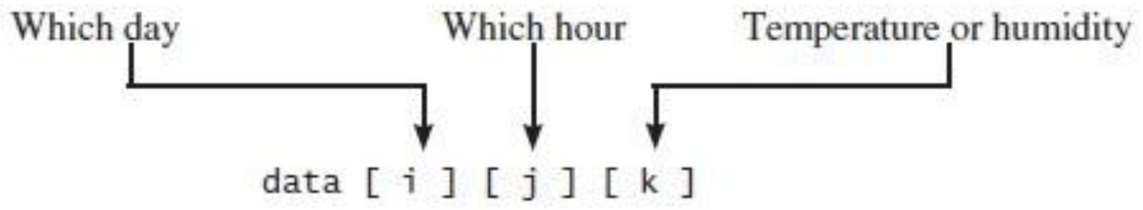


Demo Project: Weather Information

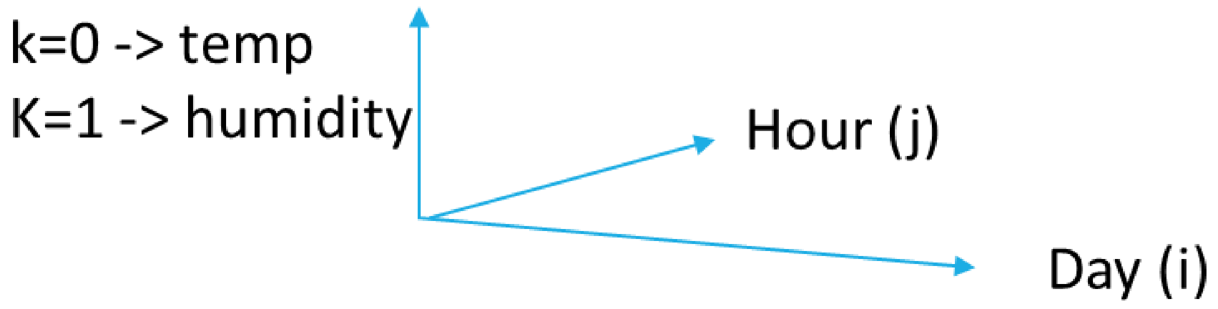
Weather.java

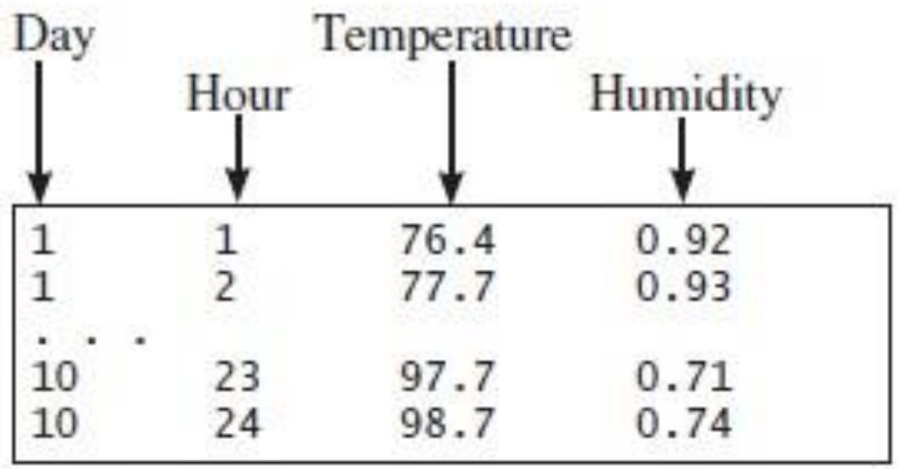
- Suppose a meteorology station records the temperature and humidity at each hour of every day and stores the data for the past ten days in a text file named **weather.txt**.
- Each line of the file consists of four numbers that indicate the day, hour, temperature, and humidity. Your task is to write a program that calculates the **average** daily temperature and humidity for the 10 days.

Data Structure



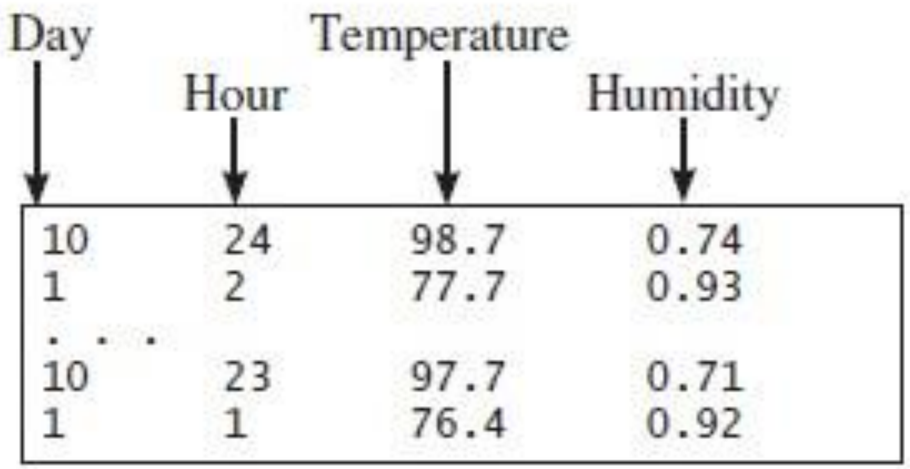
Weather.txt:




Day
Hour
Temperature
Humidity

Day	Hour	Temperature	Humidity
1	1	76.4	0.92
1	2	77.7	0.93
. . .			
10	23	97.7	0.71
10	24	98.7	0.74

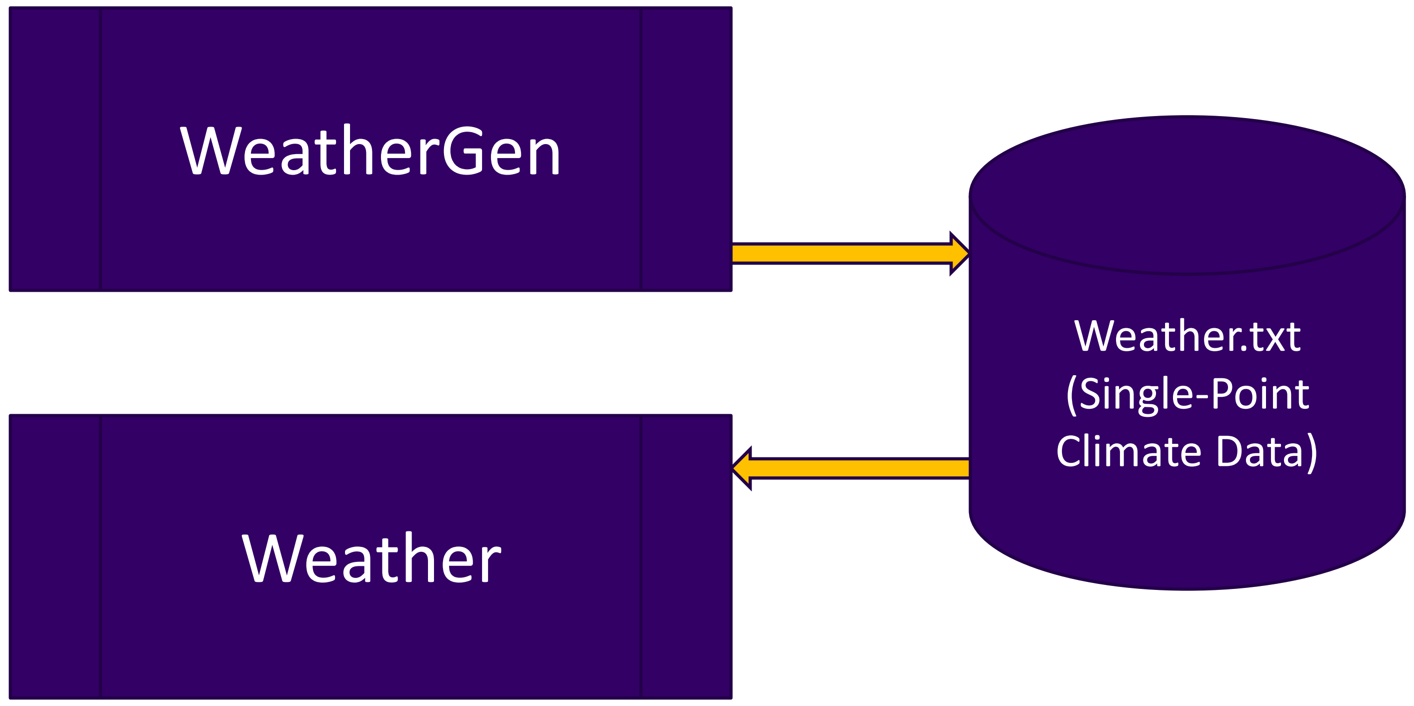
(a)


Day
Hour
Temperature
Humidity

Day	Hour	Temperature	Humidity
10	24	98.7	0.74
1	2	77.7	0.93
. . .			
10	23	97.7	0.71
1	1	76.4	0.92

(b)

Data Flow Diagram



Demonstration Program

WEATHER.JAVA

WEATHERGEN.JAVA