



# AP Computer Science A

Java Programming Essentials

[Ver. 3.0]

## Unit 3: Basic Data Structures



CHAPTER 7A: ARRAYS

DR. ERIC CHOU

IEEE SENIOR MEMBER



# Objectives

---

- K-Graph Problems
- Combination
- Image Processing
- Combination Number: Binomial Theorem
- N-D Arrays



# K-Graph

---

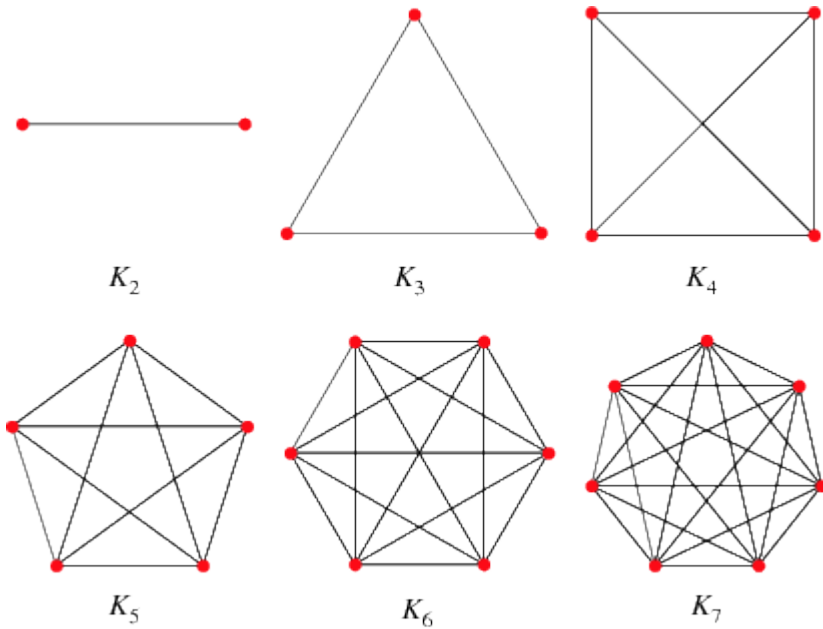
LECTURE 1



# Mutual Relationship

---

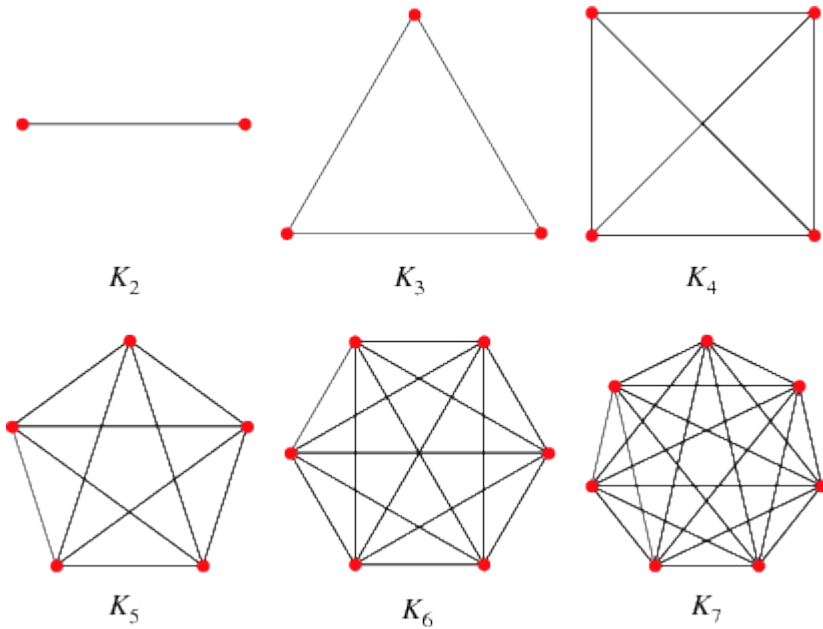
- Equality (Equality Check)
- Shortest Distance (Finding Max/Min)
- Inverse Order Count (Comparison)
- Friendships (Two-way checking)
- Matrix Transpose (Swap)



```
for (int i=1; i< arr.length; i++){
    for (int j=0; j<i; j++){
        cell[i][j] = '*';
        /* for all solution space
           (i, j) */
    }
}
```

K-graph is for  $C(n, 2)$ .

	0	1	2
0			
1	*		
2	*	*	



```
for (int i=0; i< arr.length-1; i++){
    for (int j=i+1; j<arr.length; j++){
        cell[i][j] = '*';
        /* for all solution space
           (i, j) */
    }
}
```

K-graph is for  $C(n, 2)$ .

	0	1	2
0		*	*
1			*
2			



# Diversity

Check if a group of  $n$  numbers are diverse

---

- A group of numbers are called diverse if none of its members repeats.
- Use partial 2-D traversal for all one-to-one mapping from one member to another. If there is any equality, then return false. Otherwise, if there is no identical matching members, we return true.

```
1 public class Diverse
2 {
3     static int[] a = {1, 13, 6, 3, 7, 3, 7, 9, 8, 4, 32};
4     static int[] b = {9, 8, 11, 7, 4, 3};
5
6     public static boolean isDiverse(int[] x){
7
8         for (int i=1; i<x.length; i++){
9             for (int j=0; j<i; j++){
10                 if (x[i]==x[j]) return false;
11             }
12         }
13         return true;
14     }
15
16     public static void main(String[] args){
17         System.out.println("isDiverse(a)="+isDiverse(a));
18         System.out.println("isDiverse(b)="+isDiverse(b));
19     }
20 }
```

isDiverse(a)=false  
isDiverse(b)=true





# Matrix Transpose

---

- Each pair of elements will be swapped only once.
- Use K-map traversal. (upper half or lower half)
- Swap  $\text{arr}[r][c]$  with  $\text{arr}[c][r]$

```

2 public class MatrixTranspose{
3     static int[][] m={
4         {1, 2, 3},
5         {4, 5, 6},
6         {7, 8, 9}
7     };
8     public static void transpose(int[][] m){
9         for (int i=1; i<m.length; i++){
10             for (int j=0; j<i; j++){
11                 int tmp = m[i][j];
12                 m[i][j] = m[j][i];
13                 m[j][i] = tmp;
14             }
15         }
16     }
17     public static void display(int[][] m){
18         for (int i=0; i<m.length; i++){
19             for (int j=0; j<m[i].length; j++){
20                 System.out.print(m[i][j]+" ");
21             }
22             System.out.println();
23         }
24     }
25     public static void main(String[] args){
26         System.out.println("\nBefore Transpose:");
27         display(m);
28         transpose(m);
29         System.out.println("\nAfter Transpose:");
30         display(m);
31     }
32 }

```

Before Transpose:

```

1 2 3
4 5 6
7 8 9

```

After Transpose:

```

1 4 7
2 5 8
3 6 9

```



# Inverse Order Count

---

- Given an array, see how many inverse relationship among the numbers
- Example:  $a = \{1, 4, 3, 2, 5\}$ ;
- Totally 3 inverse.

inverseCount(a)=3

```
1 public class InverseCount
2 {
3     static int[] a = {1,4, 3, 2, 5};
4     public static int inverseCount(int[] x){
5         int count = 0;
6         for (int i=1; i<x.length; i++){
7             for (int j=0; j<i; j++){
8                 if (x[i]<x[j]) count++;
9             }
10        }
11        return count;
12    }
13
14    public static void main(String[] args){
15        System.out.println("inverseCount(a)="+inverseCount(a));
16    }
17 }
```

inverseCountUpper(a)=3

```
1 public class InverseCountUpper
2 {
3     static int[] a = {1,4, 3, 2, 5};
4     public static int inverseCountUpper(int[] x){
5         int count = 0;
6         for (int i=0; i<x.length-1; i++){
7             for (int j=i+1; j<x.length; j++){
8                 if (x[i]>x[j]) count++;
9             }
10        }
11        return count;
12    }
13
14    public static void main(String[] args){
15        System.out.println("inverseCountUpper(a)="+inverseCountUpper(a));
16    }
17 }
```



# Shortest Distance

---

- Find the two points in a set of 8 points. These two points have the shortest distance between them.
- Using K-map upper-half 2-D Traversal.
- Compare and identify the pair of points which have the minimum distance.
- Use  $\text{Pair}(x, y)$  as the two tuples return value.
- Use  $\text{Point}(a, b)$  as the two tuples parameters.

```

1 public class ShortestDistance
2 {
3     static Point[] alist = {
4         new Point(0.0, 0.0), new Point(1.0, 3.0),
5         new Point(-1.2, -1.0), new Point(-2.0, 0.5),
6         new Point(4.0, 0.0), new Point(3.0, -3.0),
7         new Point(-5.1, 1.0), new Point(-0.2, 3.5)
8     };
9     static public class Point{
10         double x=0;
11         double y=0;
12         Point(double a, double b){
13             x=a;
14             y=b;
15         }
16     }
17     static public class Pair{
18         int p1=0;
19         int p2=0;
20         Pair(int a, int b){
21             p1=a;
22             p2=b;
23         }
24     }

```

Minimum Distance=1.3 is between Point[1] and Point[7]

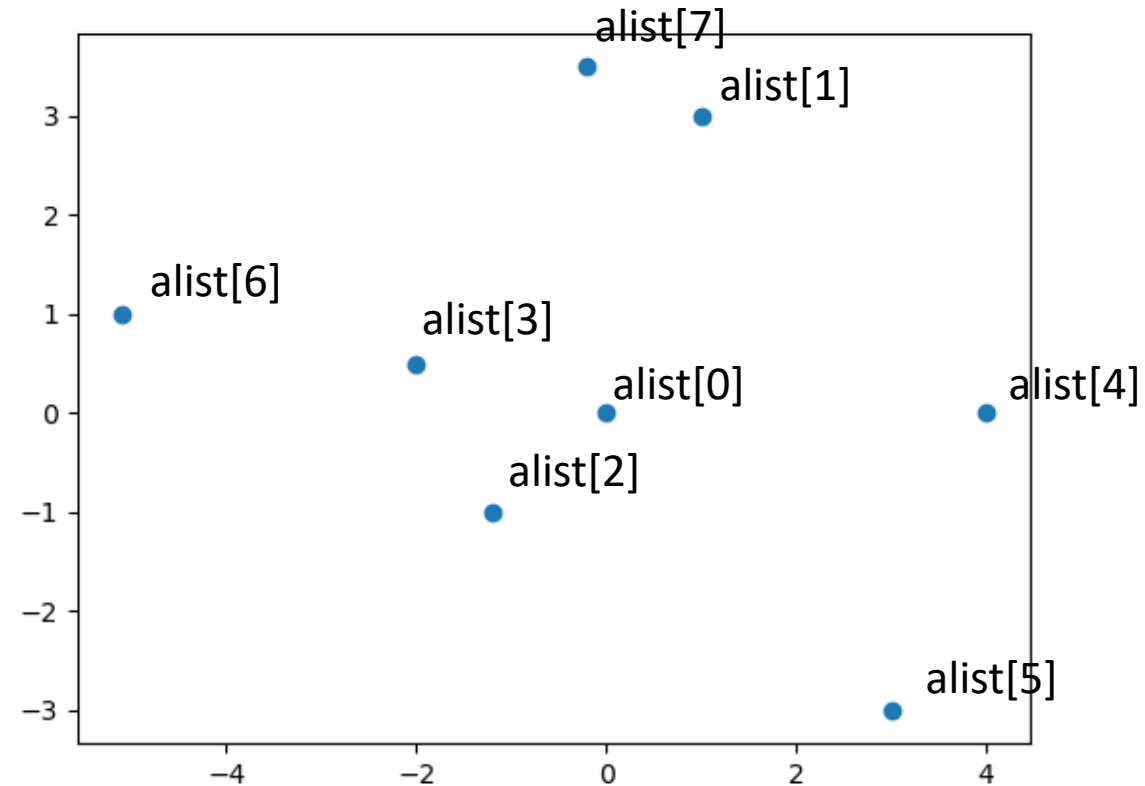
```

25 public static double distance(Point a, Point b){
26     return Math.sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
27 }
28
29 public static void main(String[] args){
30     Pair min = new Pair(0, 1);
31     double min_distance = distance(alist[0], alist[1]);
32     for (int i=0; i<alist.length-1; i++){
33         for (int j=i+1; j<alist.length; j++){
34             double dist = distance(alist[i], alist[j]);
35             if (dist < min_distance){
36                 min_distance = dist;
37                 min = new Pair(i, j);
38             }
39         }
40     }
41     System.out.println("Minimum Distance="+min_distance+
42         " is between Point["+min.p1+"] and Point["+min.p2+"]");
43 }
44 }

```

# Showing Minimum Distance of Points Using PyLab

```
1 from pylab import *
2
3 plist = [(0.0, 0.0), (1.0, 3.0), (-1.2, -1.0), (-2.0, 0.5),
4          (4.0, 0.0), (3.0, -3.0), (-5.1, 1.0), (-0.2, 3.5)]
5 x = [a[0] for a in plist]
6 y = [a[1] for a in plist]
7
8 figure()
9 scatter(x, y)
10 show()
```







# Friendship Checking

---

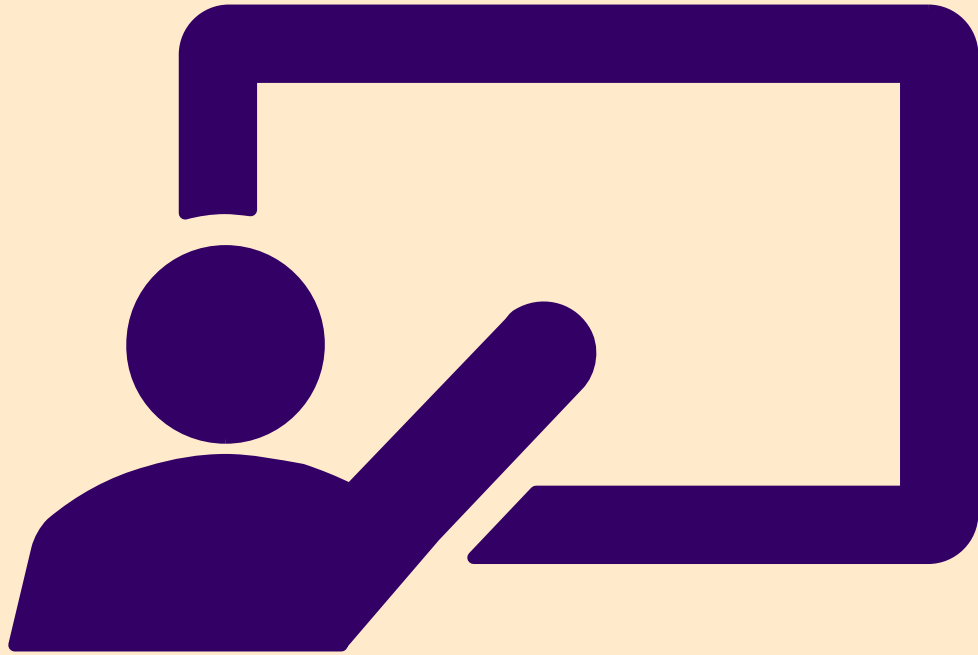
- Mutual Checking for friendship relationship.
- We have an matrix of likes[[]]. If a person who id is i likes another person j, then like[i][j] = true.
- If (likes[i][j] && likes[j][i]), then the function isFriends(i, j) will return true.
- Find out all the pair of the people who are friends. Using the same pair class from the previous example.

```
1 import java.util.ArrayList;
2 public class Friends
3 {   static boolean[][] likes = {
4         {false, true, true, false, false, true},
5         {false, false, true, true, false, true},
6         {true, true, false, true, true, false},
7         {true, false, true, false, false, false},
8         {false, false, true, false, false, true},
9         {true, false, true, true, false, false}
10    };
11    static public class Pair{
12        int p1=0;
13        int p2=0;
14        Pair(int a, int b){
15            p1=a;
16            p2=b;
17        }
18        public String toString(){ return "("+p1+", "+p2+")"; }
19    }
```

```
static boolean[][] likes = {
    {false, true, true, false, false, true},
    {false, false, true, true, false, true},
    {true, true, false, true, true, false},
    {true, false, true, false, false, false},
    {false, false, true, false, false, true},
    {true, false, true, true, false, false}
};
```

```
[(2, 0), (2, 1), (3, 2), (4, 2), (5, 0)]
```

```
20 public static void main(String[] args){
21     ArrayList<Pair> f = new ArrayList<Pair>();
22
23     for (int i=1; i<likes.length; i++){
24         for (int j=0; j<i; j++){
25             if (likes[i][j] && likes[j][i]) f.add(new Pair(i, j));
26
27         }
28     }
29
30     System.out.println(f);
31 }
32 }
```



# Combination

---

LECTURE 2



# Generation of All Combinations of Elements $C_2^n$

- This problem is a K-Graph problem.

```
public class Combinations
{
    static String s = "ABCDE";

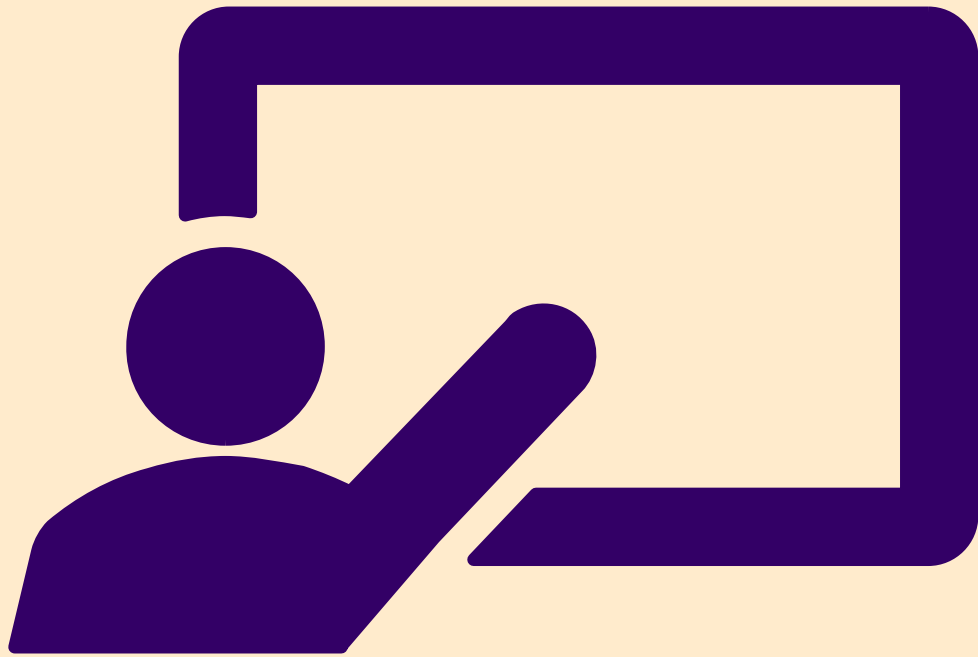
    public static void main(String[] args){
        System.out.print("\f");
        ArrayList<String> combinations = new ArrayList<String>();
        for (int i=0; i<s.length()-1; i++){
            for (int j=i+1; j<s.length(); j++){
                String x = ""+s.charAt(i)+s.charAt(j);
                combinations.add(x);
            }
        }
        System.out.println(combinations);
    }
}
```



# Generation of All Combinations of Elements $C_3^n$

```
public class Combinations3
{
    static String s = "ABCDE";

    public static void main(String[] args){
        System.out.print("\f");
        ArrayList<String> combinations3 = new ArrayList<String>();
        for (int i=0; i<s.length()-2; i++){
            for (int j=i+1; j<s.length()-1; j++){
                for (int k=j+1; k<s.length(); k++){
                    String x = ""+s.charAt(i)+s.charAt(j)+s.charAt(k);
                    combinations3.add(x);
                }
            }
        }
        System.out.println(combinations3);
    }
}
```



# Passing 2D Arrays to Methods

---

LECTURE 3



# 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).

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

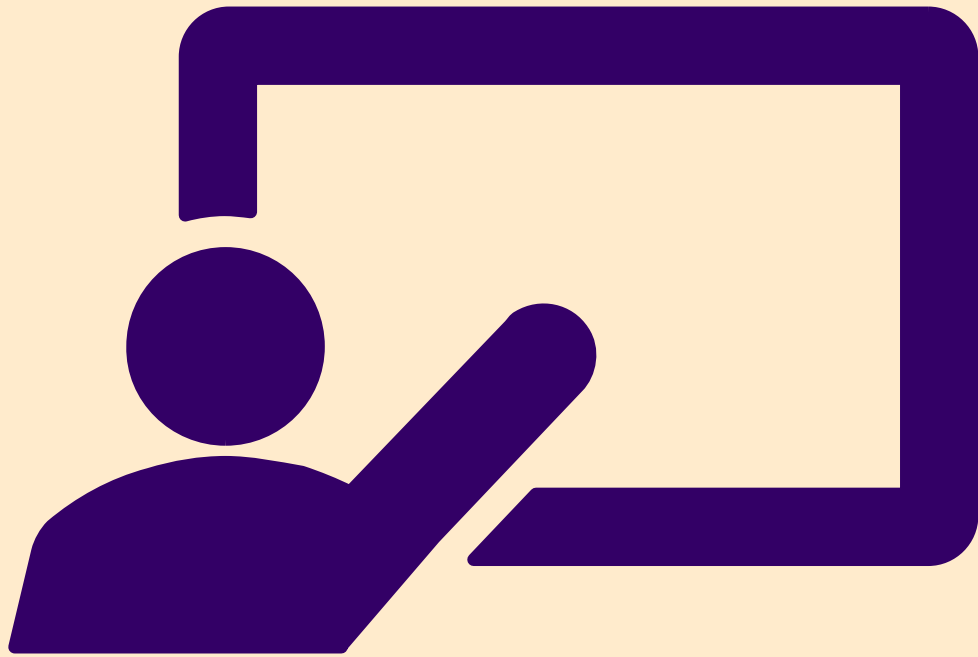
Sum of all elements is 64
```



# Demonstration Program

---

PASS2DARRAY.JAVA



# 2-D Array Image Processing

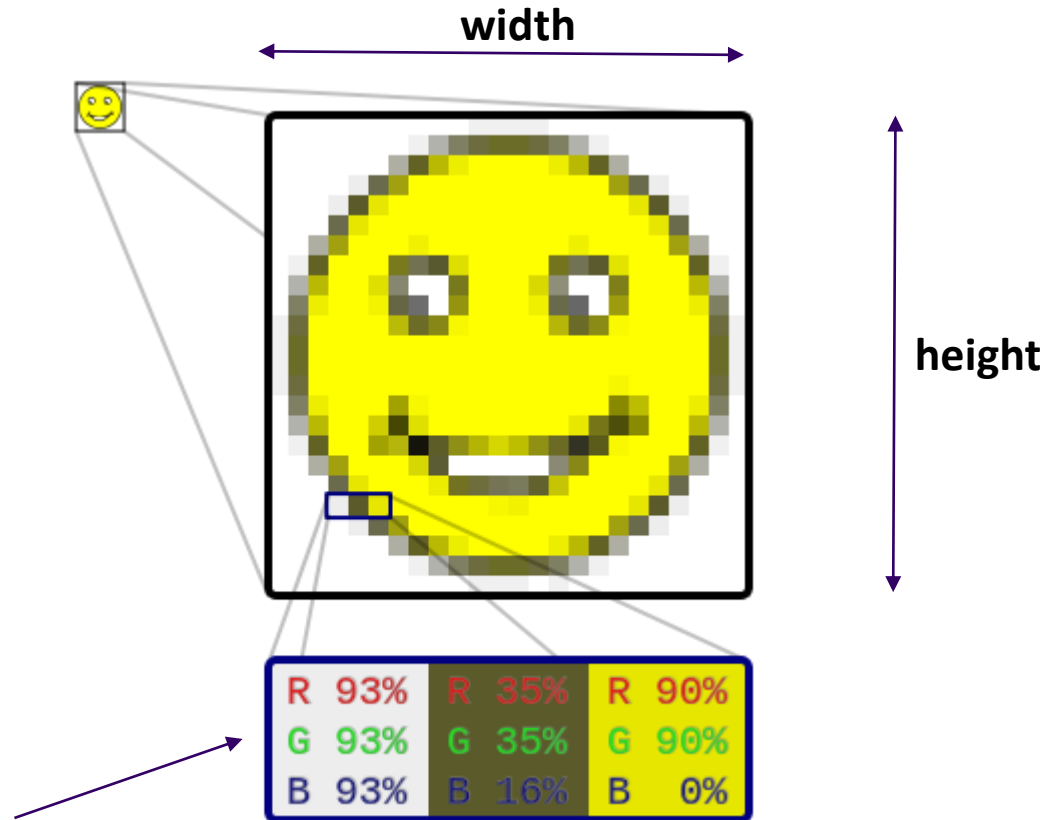
---

LECTURE 4



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

[https://en.wikipedia.org/wiki/Raster\\_graphics](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	:	<code>java.awt.Color[r=255, g=0, b=0]</code>
GREEN	:	<code>java.awt.Color[r=0, g=255, b=0]</code>
BLUE	:	<code>java.awt.Color[r=0, g=0, b=255]</code>
YELLOW	:	<code>java.awt.Color[r=255, g=255, b=0]</code>
MAGENTA	:	<code>java.awt.Color[r=255, g=0, b=255]</code>
CYAN	:	<code>java.awt.Color[r=0, g=255, b=255]</code>
WHITE	:	<code>java.awt.Color[r=255, g=255, b=255]</code>
BLACK	:	<code>java.awt.Color[r=0, g=0, b=0]</code>
GRAY	:	<code>java.awt.Color[r=128, g=128, b=128]</code>
<b>LIGHT_GRAY</b>	:	<code>java.awt.Color[r=192, g=192, b=192]</code>
<b>DARK_GRAY</b>	:	<code>java.awt.Color[r=64, g=64, b=64]</code>
PINK	:	<code>java.awt.Color[r=255, g=175, b=175]</code>
ORANGE	:	<code>java.awt.Color[r=255, g=200, b=0]</code>



# 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

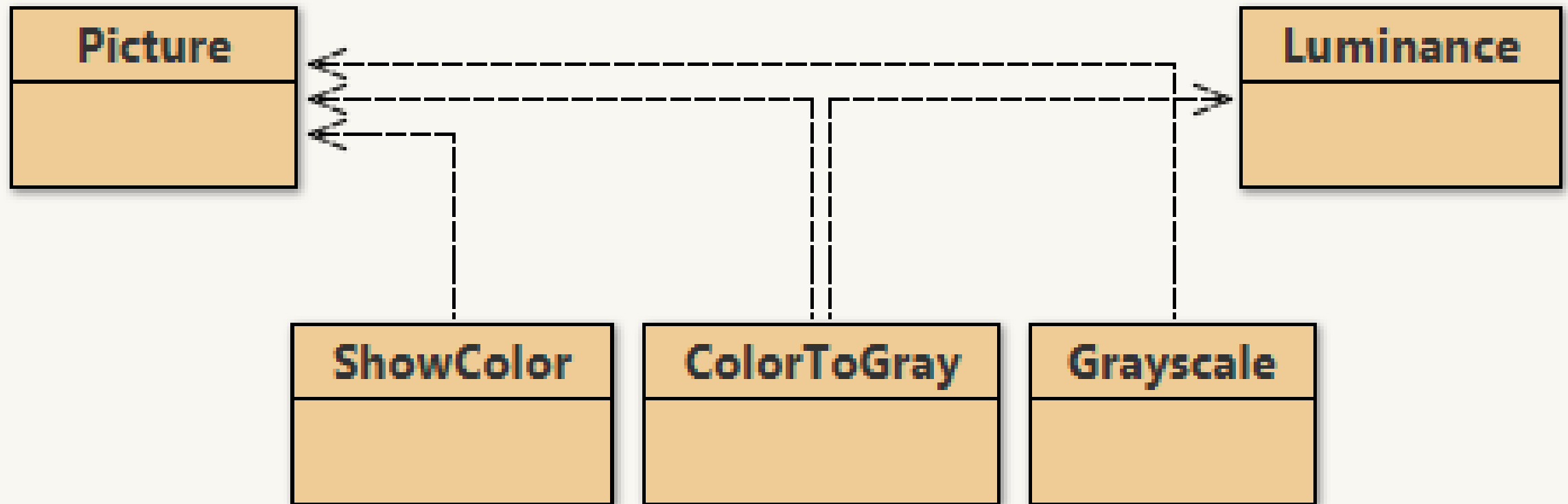
**ColorToGray.java:** convert the color image to gray level image.

**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.







# Demonstration Program

---

PICTURE.JAVA + LUMINANCE.JAVA

SHOWCOLOR.JAVA

COLORTOGRAY.JAVA

GRAYSCALE.JAVA



# Application

## Grading Multiple-Choice Tests

---

LECTURE 5



# Problem: Grading Multiple-Choice Test

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

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

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

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

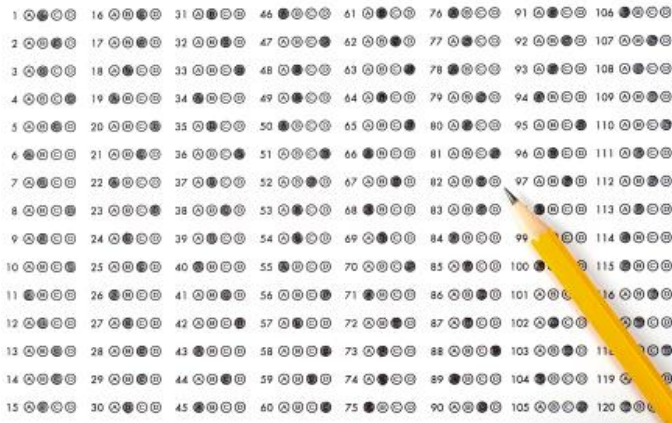
---

STUDENTANSWER.JAVA

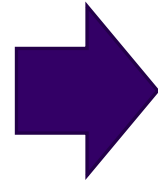


# Ragged Array for Student's Score for Multiple Subjects

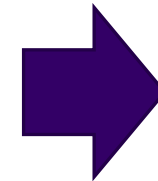
StudentScoreMultiple.java



Student Answer Sheet



Answer Sheet Scanning Reader

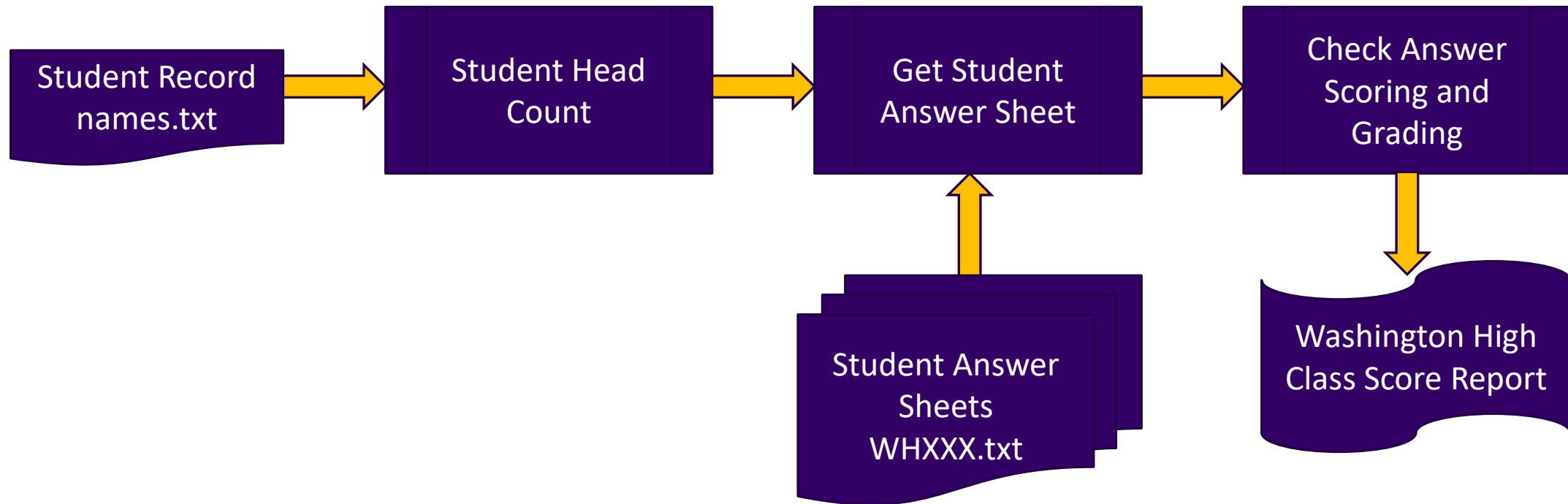


Student Answer Sheet Text File  
WH000.txt



# Data Flow Diagram

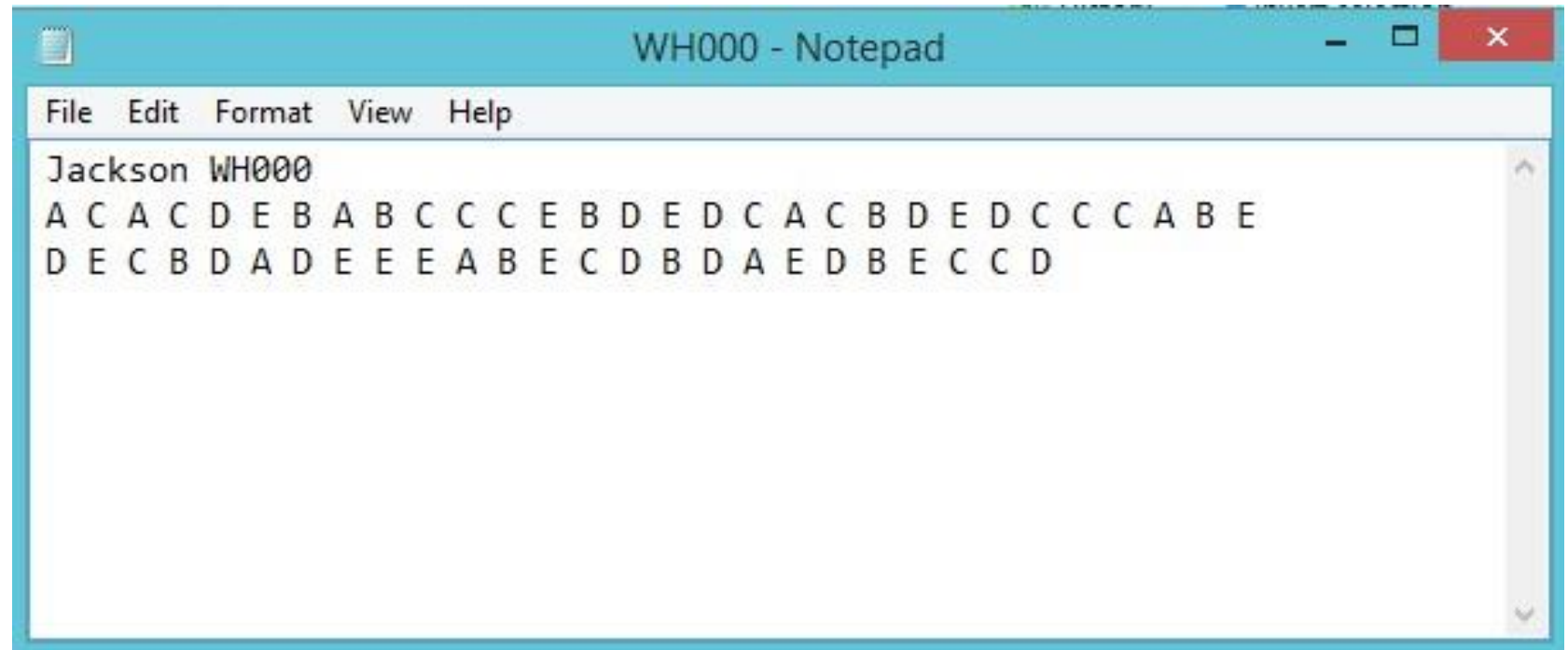
StudentScoreMultiple.java



# Student Record and Answer Sheet



Student Record names.txt



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();  
}
```

# 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;  
}
```



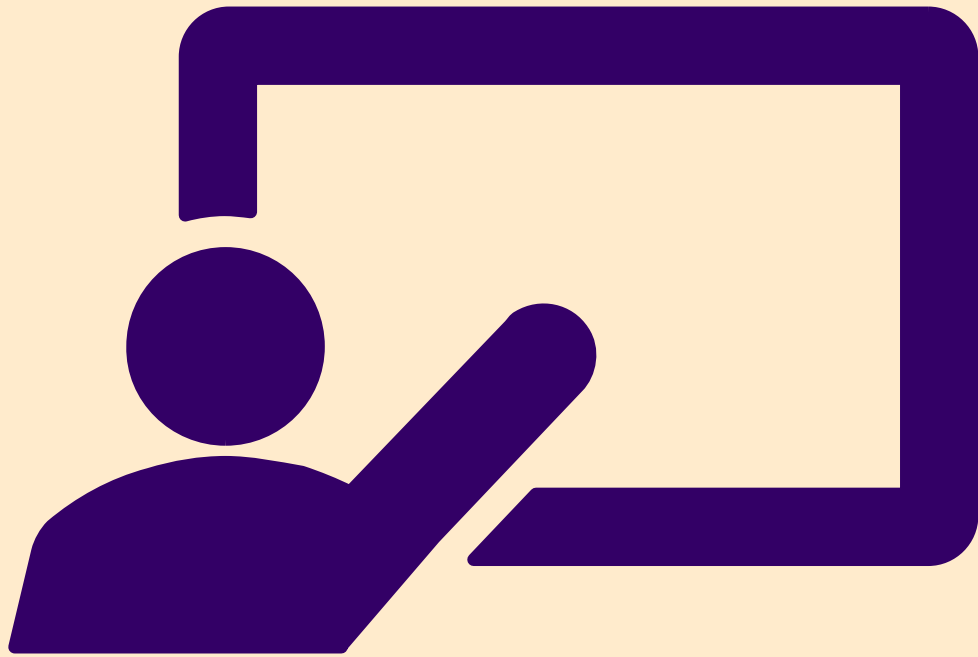
Options						
Washington High School Semester Class Score Report Card						
ID: WH000	Name: Jackson	Math Score: 87	Math Grade: B	English Score: 88	English Grade: B	
ID: WH001	Name: Aiden	Math Score: 90	Math Grade: A	English Score: 84	English Grade: B	
ID: WH002	Name: Liam	Math Score: 60	Math Grade: D	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



# Lab:

## CombinationNumber.java

---

LECTURE 6



# Purpose of this project

---

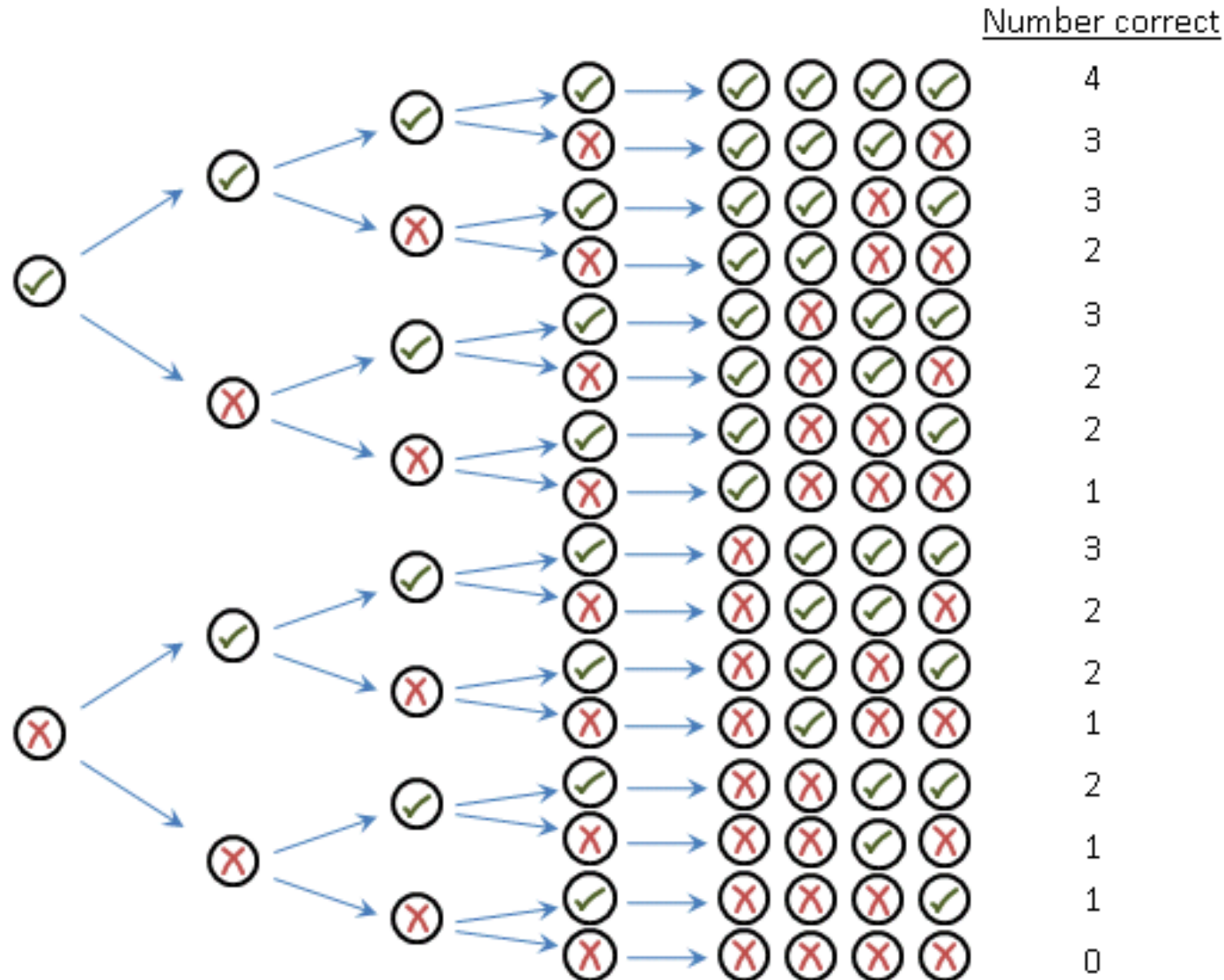
- (1) Use 2-D array for a real-world calculation application.
- (2) Exercise on the 2D **index calculation** using **mapping**.
- (3) 2-D Array can also work like a method (function)
- (4) The term **Wrapper Function**.

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

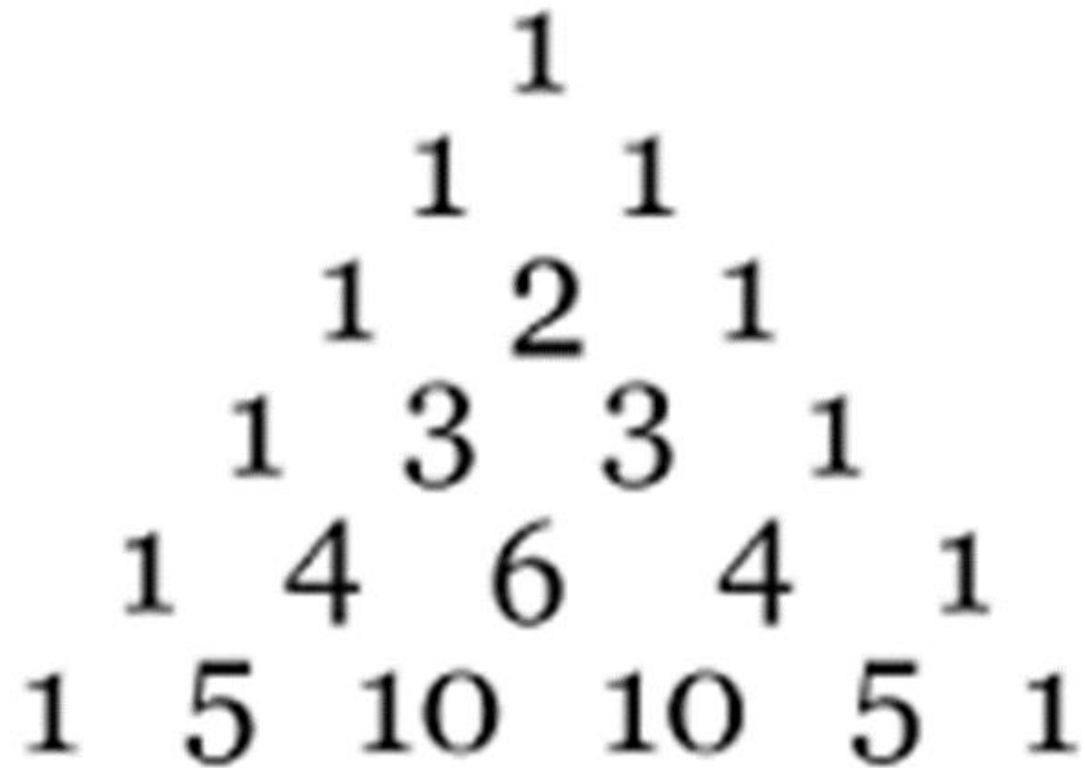
---

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



# Pascal Triangle

---





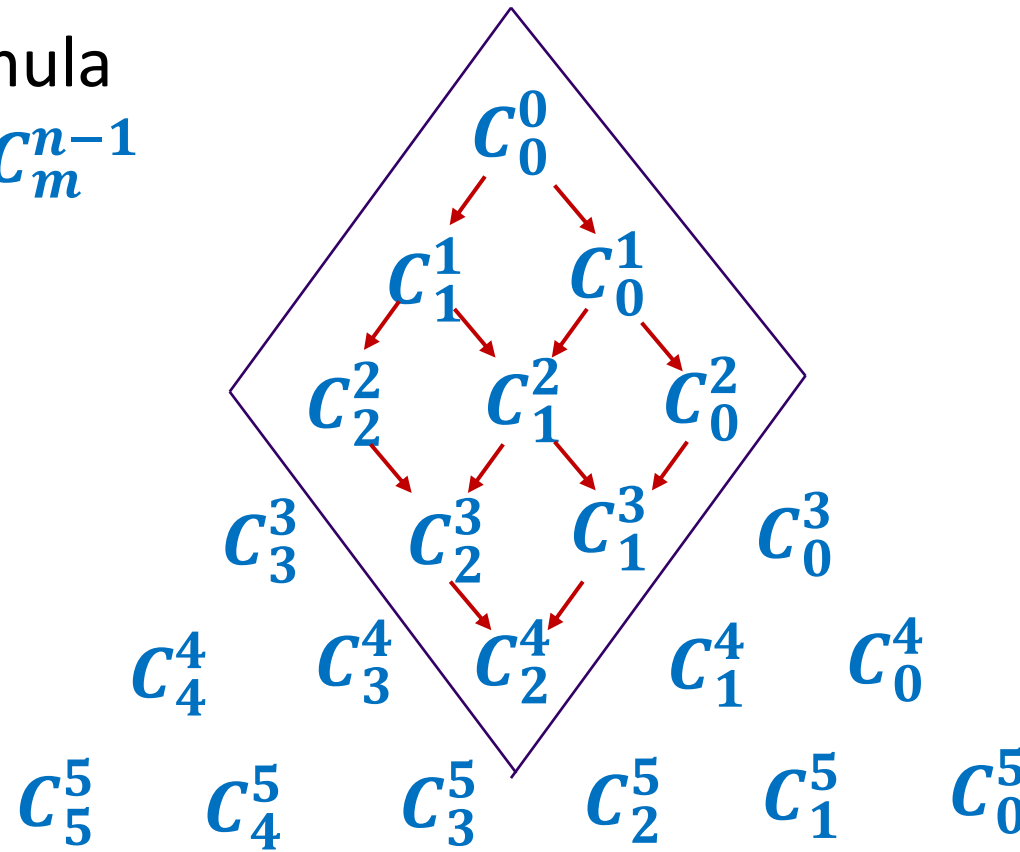
## Combination Number:

# Binomial Theorem

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

(1) Recursive Formula

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





# 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

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

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

$$a = n - m$$

$$b = m$$

$$(a+b) = n$$

$$b = m$$

To find  $C_m^n$  :

(1) We need to create 2D Array:

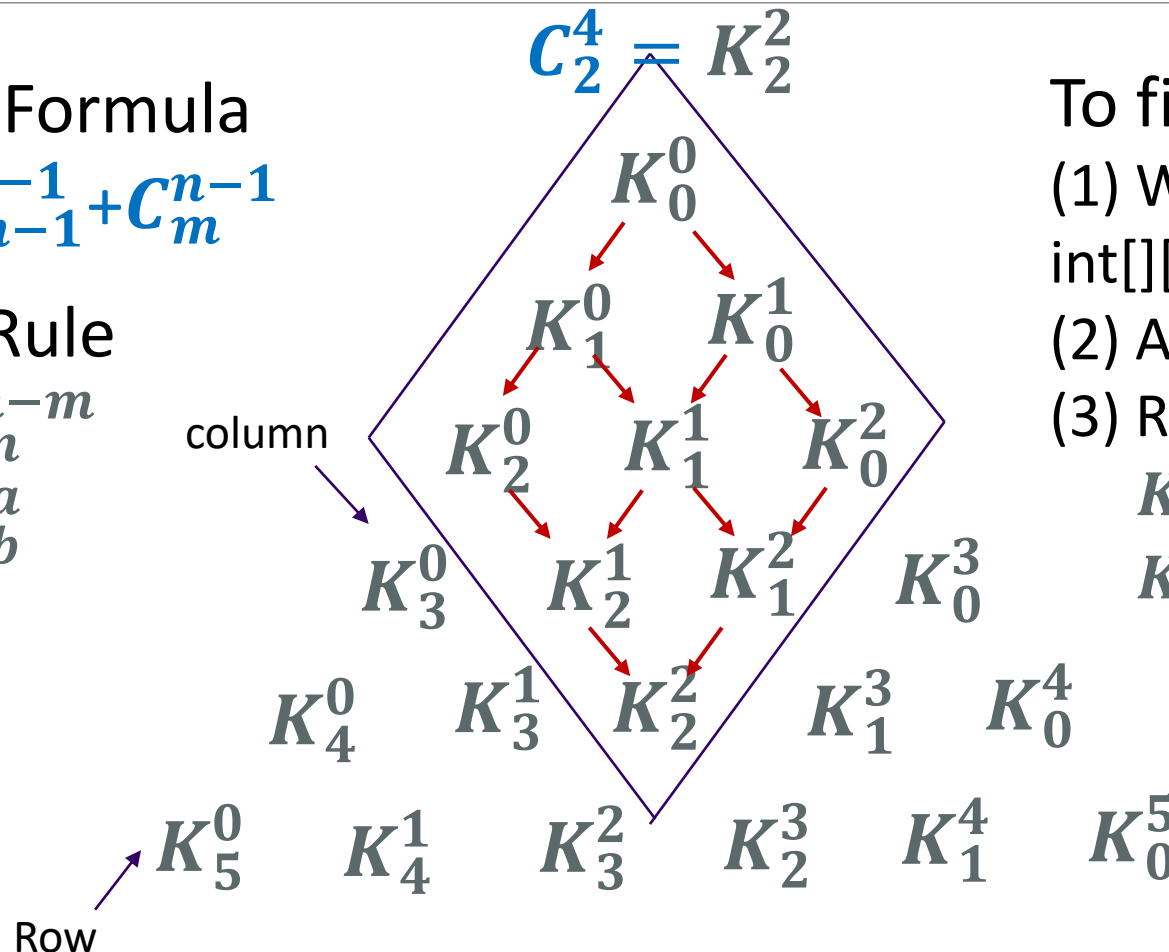
```
int[][] k = new int[n-m+1][m+1];
```

(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>
```

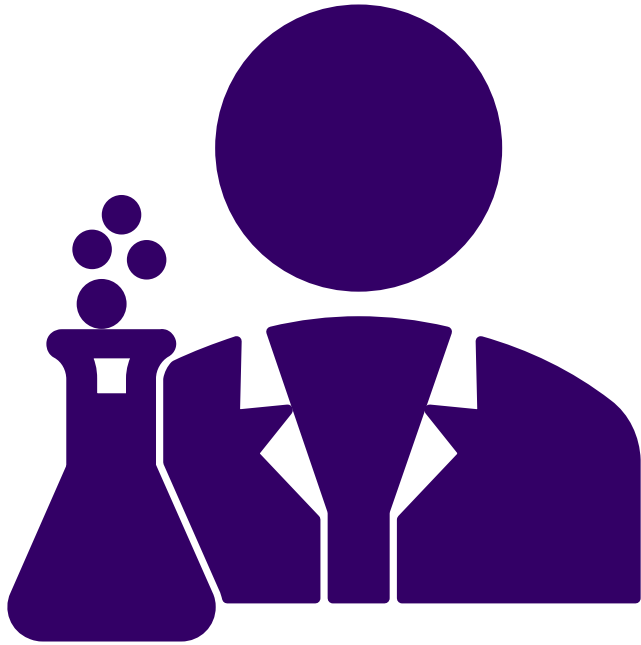


# Expected Result:

BlueJ: Terminal Window - Chapter08

Options										
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



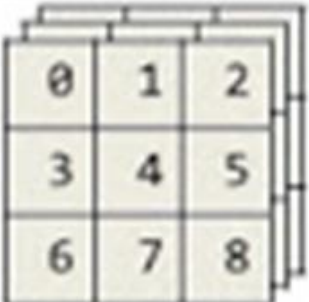
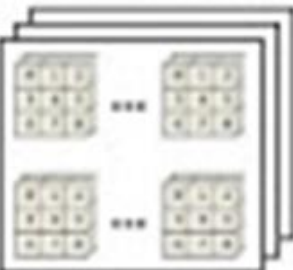


# N-D Arrays

---

## LECTURE 7

# What is an array?

Dimensions	Example	Terminology
1		Vector
2		Matrix
3		3D Array (3 <sup>rd</sup> 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  $n \geq 3$ .



# 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}

Which student

Which exam

Multiple-choice or essay

scores[ i ] [ j ] [ k ]



# Demo Project: Weather Model

---

## Global Climate Models:

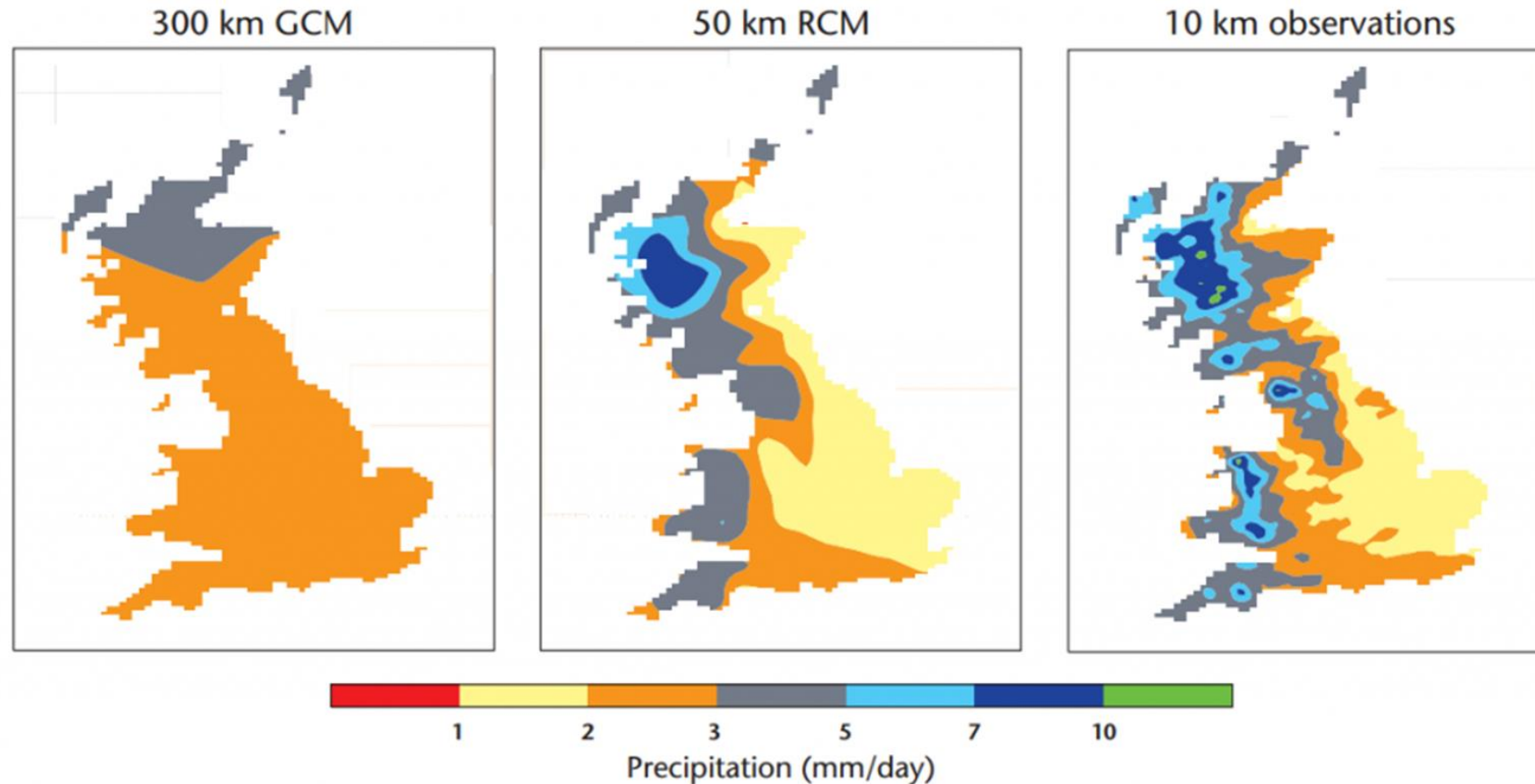
Climate models divide the surface of the Earth into a horizontal grid, the atmosphere into vertical levels, and time into discrete timesteps.

```
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)





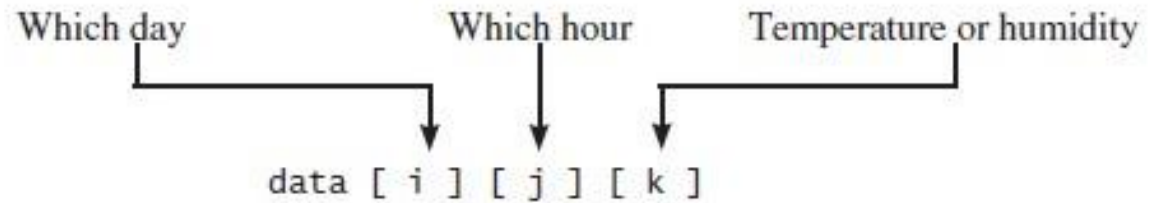


# Demo Project: Weather Information

[Weather.java](#)

---

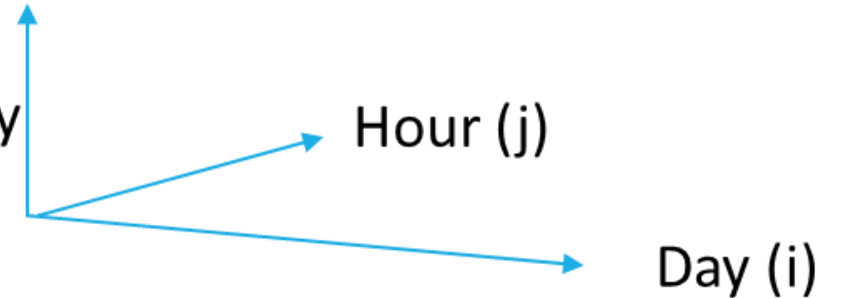
- Suppose a meteorology station records the temperature and humidity at each hour of every day and stores the data for the past ten days in a text file named **weather.txt**.
- Each line of the file consists of four numbers that indicate the day, hour, temperature, and humidity. Your task is to write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



# Data Structure

Weather.txt:

k=0 -> temp  
k=1 -> 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)

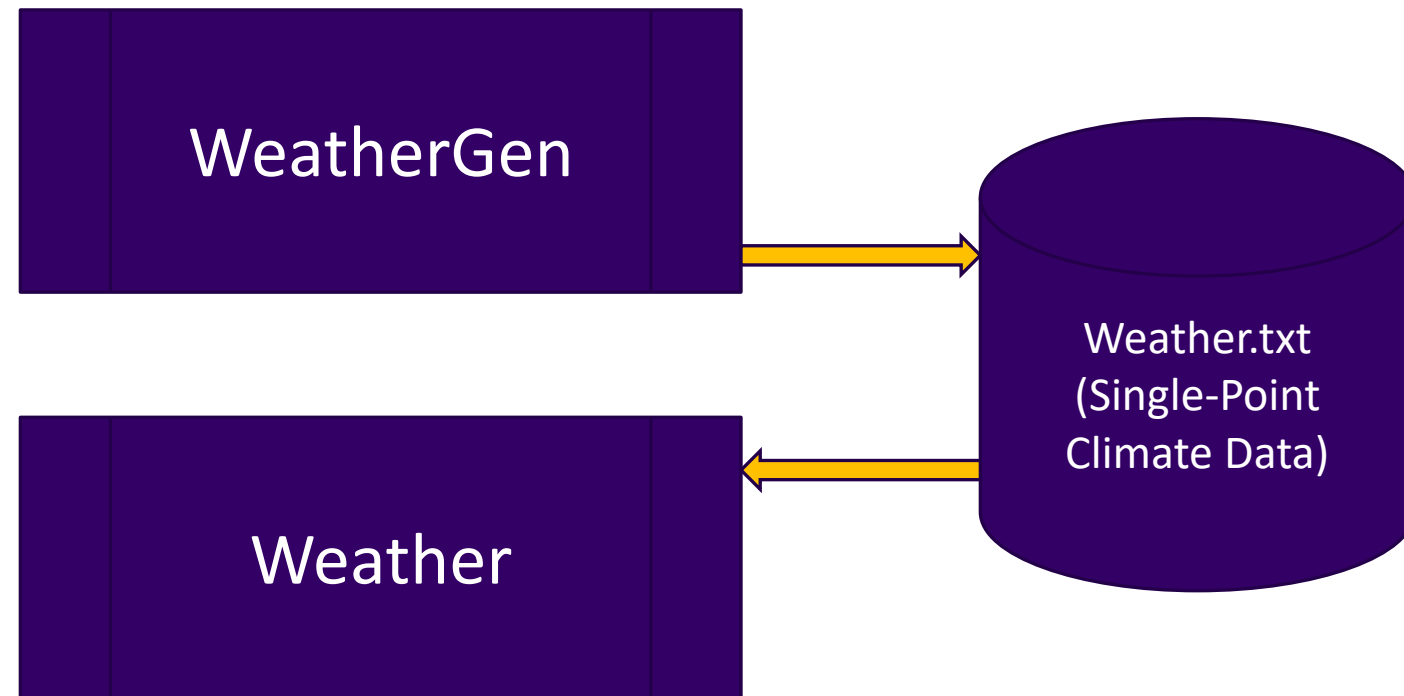
Day	Hour	Temperature	Humidity
10	24	98.7	0.74
1	2	77.7	0.93
...	...	...	...
10	23	97.7	0.71
1	1	76.4	0.92

(b)



# Data Flow Diagram

---





# Demonstration Program

---

WEATHER.JAVA

WEATHERGEN.JAVA