**CS1010J Finals Tips**

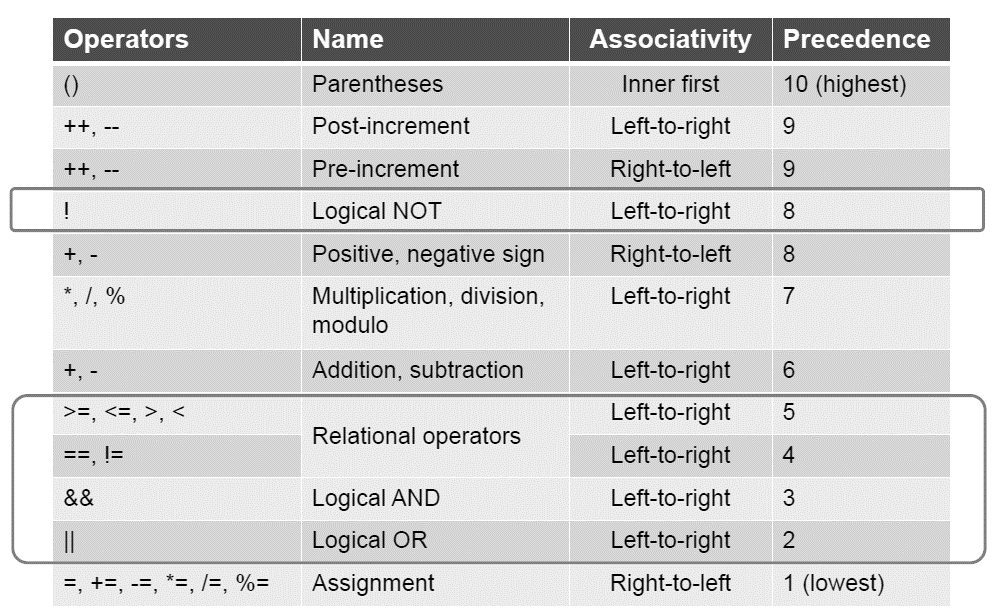
**1.1 Printing of String values**

* System.out.println(1 + 2) = 3
* System.out.println(“Java” + 1 + 2 +3) = Java123
* System.out.println(1 + 2 + ”Java” +3) = 3Java3
* System.out.println(“Java” + (1 + 2 + 3)) = Java6

**1.3 Logical Operators Truth Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **y** | **x && y** | **x || y** | **!x** |
| *True* | *True* | *True* | *True* | *False* |
| *True* | *False* | *False* | *True* | *False* |
| *False* | *True* | *False* | *True* | *True* |
| *False* | *False* | *False* | *False* | *True* |

**1.4 Precedence Table**



**1.5 Types of Errors**

* Syntax Errors – *Program does not obey Java grammar e.g incorrect expression, missing semi-colon, uninitialized variable*
* Run-Time Error – *Program terminates due to illegal operation such as divide by zero*
* Logic Error – *Program executed but produce wrong result*

**1.6 Calculate Actual Modulo Value with Calculator**

*Number / Divisor = Int Value*

*Modulo = Number – (Int Value \* Divisor)*

**1.7 Printing Output**

* System.out.println(“Bryan”);

System.out.println(“Tan”);

Output: Bryan

Tan

* System.out.print(“Bryan”);

System.out.println(“Tan”); Output: Bryan Tan

**1.8 Import Java Statements**

import Java.util.\*;

import Java.text.DecimalFormat;

**2.0 Strings**

**str.toUpperCase();** // Output: "JOHNDOE123"

**str.toLowerCase();** // Output: "johndoe123"

**str.equalsIgnoreCase("JANE\_DOE123");** // Output: false

**str.length();** // Output: 10

**str.replace('1', '!');** // Output: "johnDoe!23"

**str.substring(0, enteredString.length() - 1);** // Output: "johnDoe12"

**str.substring(0, 1);** // Output: "j"

**str.substring(1, 2);** // Output: "o"

**str.substring(1);** // Output: "ohnDoe123"

**String[] parts = str.split(" ");** // For "john doe 123", Output: **["john", "doe", "123"]**

**sampleString.charAt(0)** returns 'H', which is the first character in "Hello, World!". Throws exception if cannot find

**sampleString.charAt(4)** returns 'o', which is the fifth character in "Hello, World!". Throws exception if cannot find

**sampleString.charAt(sampleString.length() - 1)** returns '!', which is the last character in "Hello, World!".

**int lastA = "Java Programming".lastIndexOf('a');** // lastA will be 13, as the last 'a' is at index 13. Return -1 if cannot find

**int firstA = "Java Programming".indexOf('a');** // firstA will be 1, as the first 'a' is at index 1. . Return -1 if cannot find

**Character.isWhitespace(space)** returns true because the character ' ' is a whitespace character.

**Character.isWhitespace(letter)** returns false because 'A' is not a whitespace character.

**originalString.trim()** removes the leading and trailing spaces from the string " Hello, World! ", resulting in "Hello, World!".

**str.equalsIgnoreCase("Java")** // Output depends on the value of **str**, ignores case in comparison.

**Character.isDigit(ch)** // Returns true if chis a digit character, or false otherwise.

**Character.isLetter(ch)** // Returns true, if ch is an alphabet letter, or false otherwise

**String[] words = {"Java", "is", "cool"};**

**String sentence = String.join(" ", words); //** join sentence together

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

// An array of integers to copy to

int[] dest = new int[5];

**System.arraycopy(src, 1, dest, 0, 3);** // Copying elements from position 1 of src to position 0 of dest, 3 elements.

Output: [2, 3, 4, 0, 0]

Note:

Remember to always add space for your strings e.g. “hellothere”, “hello there”

Always trim() before returning your strings

Always put sc.nextLine() after sc.nextInt();

**These methods will all not affect the original string, you still need to assign it if u want to make use.**

**String 2.1**

1. **// Math.abs**

int a = -5;

int absValue = Math.abs(a); // Input: -5, Output: 5

1. **// Math.ceil**

double b = 3.2;

double ceilValue = Math.ceil(b); // Input: 3.2, Output: 4.0

1. **// Math.floor**

double c = 3.8;

double floorValue = Math.floor(c); // Input: 3.8, Output: 3.0

1. **// Math.round**

float d = 3.5f;

int roundValue = Math.round(d); // Input: 3.5, Output: 4

1. **// Math.min**

int e = 5, f = 10;

int minValue = Math.min(e, f); // Input: 5, 10, Output: 5

1. **// Math.max**

int g = 5, h = 10;

int maxValue = Math.max(g, h); // Input: 5, 10, Output: 10

1. **// Math.sqrt**

double i = 9;

double sqrtValue = Math.sqrt(i); // Input: 9, Output: 3.0

1. **// Math.pow**

double j = 2, k = 3;

double powValue = Math.pow(j, k); // Input: 2, 3, Output: 8.0

1. **// Math.random (range 1 to 10)**

int min = 1;

int max = 10;

int randomInt = min + (int)(Math.random() \* ((max - min) + 1));

|  |  |
| --- | --- |
|  |  |

**ASCII Table**

A screenshot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated

**Binary Search Algo**

A screenshot of a computer game

Description automatically generated

**Bubble Sort Algo**

A white sheet with black numbers and red text

Description automatically generated

|  |  |
| --- | --- |
|  |  |

Selection sortA screenshot of a computer

Description automatically generated

Bubble sortA screenshot of a game

Description automatically generated

// Bubble sort

public static void **bubbleSort(**int[] arr) {

for (int end = arr.length - 1; end > 0; end--) {

for (int i = 0; i < end; i++) { // one pass

if (arr[i] > arr[i + 1]) { // out of order

int temp = arr[i]; // swap

arr[i] = arr[i + 1];

arr[i + 1] = temp;

}

} // end inner for loop

} // end outer for loop

}

// Sort array 'arr' in increasing order

public static void **selectionSort(**int[] arr) {

// each iteration of the for loop is one pass

for (int startIdx = 0; startIdx < arr.length-1; startIdx++) {

// find the index of the minimum element

int minIndex = startIdx;

for (int i = startIdx+1; i < arr.length; i++) {

if (arr[i] < arr[minIndex]) {

minIndex = i;

}

}

// swap minimum element with element at startIdx

int temp = arr[startIdx];

arr[startIdx] = arr[minIndex];

arr[minIndex] = temp;

} // end for loop

}

Binary search (Success)A screenshot of a computer

Description automatically generated

Binary search unsuccessful

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Sample Class**

class Lift {

private int currLevel, distance; // current level and travel distance

// SPEED is defined as a constant and can be used in Lift class directly

private final int SPEED = 2; // two seconds per level

// Constructor

public Lift() {

currLevel = 1;

distance = 0;

}

// Move lift to fromLevel to pick up passenger and then

// send passenger to toLevel

public void move(int fromLevel, int toLevel) {

//Find out the total distance the lift moves from current level to the input level

distance = distance + Math.abs(fromLevel - currLevel) + Math.abs(toLevel - fromLevel);

//Update the current level to the level it reaches

currLevel = toLevel;

}

// Return current level

public int getLevel() {

return currLevel;

}

// Return the time a lift has travelled

public int getTime() {

return distance \* SPEED;

}

}

**OOP**

cir1 == cir2 compares the addresses of the objects they refer to. Not their values.

A diagram of a circle

Description automatically generatedA diagram of a circle

Description automatically generated

**toString() can work without invocation compareObjects() sample**

A screen shot of a computer code

Description automatically generated A screenshot of a computer code

Description automatically generated

2D array read by column

// Column Major Order

for (int i = 0; i < mat[0].length; i++) {

for (int j = 0; j < mat.length; j++) {

System.out.println(mat[j][i]);

}

}

2D array read by row

// Row Major Order

for (int i = 0; i < mat.length; i++) {

for (int j = 0; j < mat[i].length; j++) {

System.out.println(mat[i][j]);

}

}

**2D ARRAY CODE**

**Flatten matrix in MISC**

**Sum of matrix in MISC**

// swap matrix positions e.g. mtx[2][4]=mtx[1][10]

public static void swap(int[][] employees, int row1, int row2) {

int[] elementsInRow1 = new int[3];

for (int i = 0; i < 3; i++) {

elementsInRow1[i] = employees[row1][i];

}

for (int i = 0; i < 3; i++) {

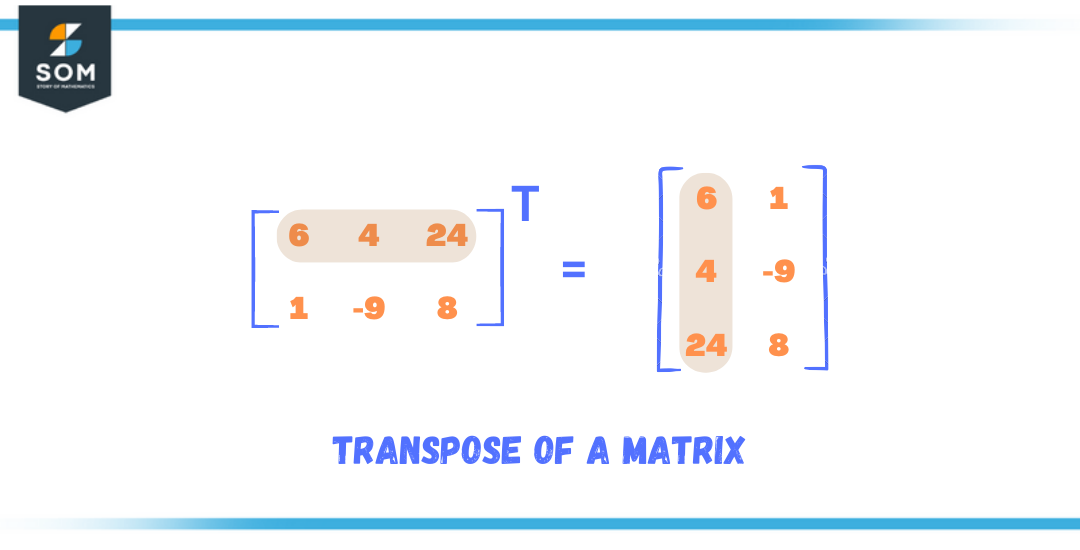
employees[row1][i] = employees[row2][i];

employees[row2][i] = elementsInRow1[i];

}

}

// transpose



private static int[][] transpose(int[][] twodm) {

int[][] newtwodm = new int[twodm[0].length][twodm.length];

for (int i = 0; i < twodm.length; i++) {

for (int j = 0; j < twodm[0].length; j++) {

newtwodm[j][i] = twodm[i][j];

}

}

return newtwodm;

}

//shift 2d matrix

public static void shift(int[][] image) {

for (int row = 0; row < image.length; row++) {

int temp = image[row][0];

for (int col = 1; col < image[row].length; col++) {

image[row][col - 1] = image[row][col];

}

image[row][image[row].length - 1] = temp;

}

}

// print 2D array

for (int z = 0; z < mat.length; z++) {

for (int j = 0; j < mat[z].length; j++) {

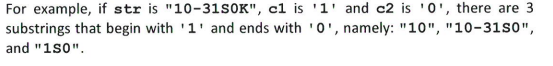
System.out.print(mat[z][j] + " ");

}

System.out.println();

}

//Count the number of substrings in a string



public static int countSubString(String str, char c1, char c2){

int count=0;

int firstIndex=0;

int nextc2=0;

if(firstIndex>0 && nextc2>0){

count++;

}else if(firstIndex==-1 || nextc2==-1){

return 0;

}

for(int i=nextc2+1;i<str.length();i++){

if(str.charAt(i)==c2 || str.charAt(i)==c1){

count++;

}

}

return count;

}

**EXAM CODES**

// Check if half the array is equals to the other half less the middle index.

A close up of a number

Description automatically generated

public static boolean equilibriumIdx(int[]arr){

int counter=0;

int sum1=0;

int sum2=0;

if (arr == null || arr.length < 3) {

return false; // An array needs at least three elements to have an equilibrium index

}

while(counter<arr.length-1){

sum1=0;

sum2=0;

for(int i=0; i< counter; i++){

sum1+=arr[i];

}

for(int j=arr.length-1; j>counter; j--){

sum2+=arr[j];

}

if(sum1==sum2){

return true;

}

counter++;

}

return false;

}

2016/2017 chatgpt version

//counts every alternate number in a 2D array e.g. A black text on a white background

Description automatically generated

public static int alternativeSum(int[][] mtx) {

int sum = 0;

boolean addElement = false; // Start with false because we skip the first element

// Iterate over each element in row-major order

for (int[] row : mtx) {

for (int elem : row) {

// Add the element if addElement is true

if (addElement) {

sum += elem;

}

// Toggle the addElement flag

addElement = !addElement;

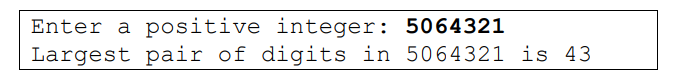
}

}

return sum;

}

//get largest digit pair in a string of numbers



public static int largestDigitPair(int num) {

// Return the num itself if less than 10

if(num<10){

return num;

}

int currentdigitpairs=num%100;

int largestDigitPair=largestDigitPair(num/100);

return Math.max(currentdigitpairs,largestDigitPair); // stub

}

// Count the number of valid north-east paths

public static int ne(int x, int y) {

if(x==0 || y==0){

return 1;

}

return ne(x-1,y) + ne(x,y-1); // stub

}

//Read x y points and return an array of points

public Point[] readPoints(){

int counter=0;

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of points: ");

int numPts = sc.nextInt();

Point [] points=new Point[numPts];

for(int i=0;i<numPts;i++){

System.out.println("Enter point "+(i+1)+": ");

Point newPoint=new Point(sc.nextInt(),sc.nextInt());

if(contains(points, counter, newPoint)){

continue;

}else{

points[counter]=newPoint;

counter++;

}

}

Point [] updatedPoints=new Point[counter];

System.arraycopy(points, 0, updatedPoints, 0, counter);

return updatedPoints;

}

//compute largest triangle problem methods 2018

public static double computeArea(Point[] pts, int a, int b, int c) {

// Extract the points using their indices

Point A = pts[a];

Point B = pts[b];

Point C = pts[c];

// Calculate the area using the given formula

double area = Math.abs(A.x \* (B.y - C.y) + B.x \* (C.y - A.y) + C.x \* (A.y - B.y)) / 2.0;

// Check if the points can form a triangle. If not, the area will be zero.

// This check is implicit in the area calculation since if all points are collinear,

// the area calculated will be 0.

return area;

}

public static double getLargestArea(Point [] pts){

double largestArea=0;

for(int i=0;i<pts.length;i++){

for(int j=i+1;j<pts.length;j++){

for(int k=j+1;k<pts.length;k++){

double area=computeArea(pts, i, j, k);

if(area>largestArea){

largestArea=area;

}

}

}

}

return largestArea;

}



public static int squareSum(int num) {

if(num<10){

return num\*num;

}

int x =num%10;

return squareSum(num/10) + x \* x;

}

//get pow of a number recursively e.g. x^4

public static double myPow(double x, int n) {

if(n==0){

return 1;

}

return x\*myPow(x,n-1); // stub

}

**EXAM CODES/RECURSION**

**MISC**

//returns true if n is a prime number

public static boolean isPrime(int n) {

if (n <= 1)

return false; // 0 and 1 are not prime

if (n <= 3)

return true; // 2 and 3 are prime

// Check if divisible by 2 or 3

if (n % 2 == 0 || n % 3 == 0)

return false;

// Check the rest of the numbers up to the square root of n

int i = 5;

while (i \* i <= n) {

if (n % i == 0 || n % (i + 2) == 0)

return false;

i += 6; // Jump in steps of 6 (as 2 and 3 are already tested)

}

return true;

}

//counts how many days since 1 jan 2000 including leap years

public static int dayOfWeek(int date) {

int day = date % 100;

int month = date / 100 % 100;

int year = date / 10000;

int[] daysInMonth = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

int numDays = 0; // number of days since 1 Jan 2000 (Saturday)

for (int y = 2000; y < year; y++) {

if (isLeapYear(y)) {

numDays += 366;

} else {

numDays += 365;

}

}

for (int m = 1; m < month; m++) {

if (m == 2) { // February

if (isLeapYear(year)) {

numDays += 29;

} else {

numDays += 28;

}

} else { // not February

numDays += daysInMonth[m];

}

}

numDays += day - 1;

return (numDays + 5) % 7 + 1;

}

//Convert string of integers to integers

public static int stringToInt(String num){

int[] numbers = new int[]{0,1,2,3,4,5,6,7,8,9};

int result = 0;

for(int i = 0; i < num.length(); i++){

char ch = num.charAt(i);

result = result \* 10 + numbers[ch - '0'];

}

return result;

}

Returns min/max value from array

public static int findMinMaxValue(int[] array) {

// Assume the first element is the smallest to start

int min = array[0];

for (int i = 1; i < array.length; i++) {

if (array[i] < min) { // change inequality > for max

min = array[i];

}

}

return min;

}

**MISC METHODS**

//returns min max index from array

public static int findMinMaxIndex(int[] arr) {

if (arr == null || arr.length == 0) {

throw new IllegalArgumentException("Array must not be null or empty");

}

int maxIndex = 0;

for (int i = 1; i < arr.length; i++) {

if (arr[i] > arr[maxIndex]) { // flip equality to < to find MinIndex

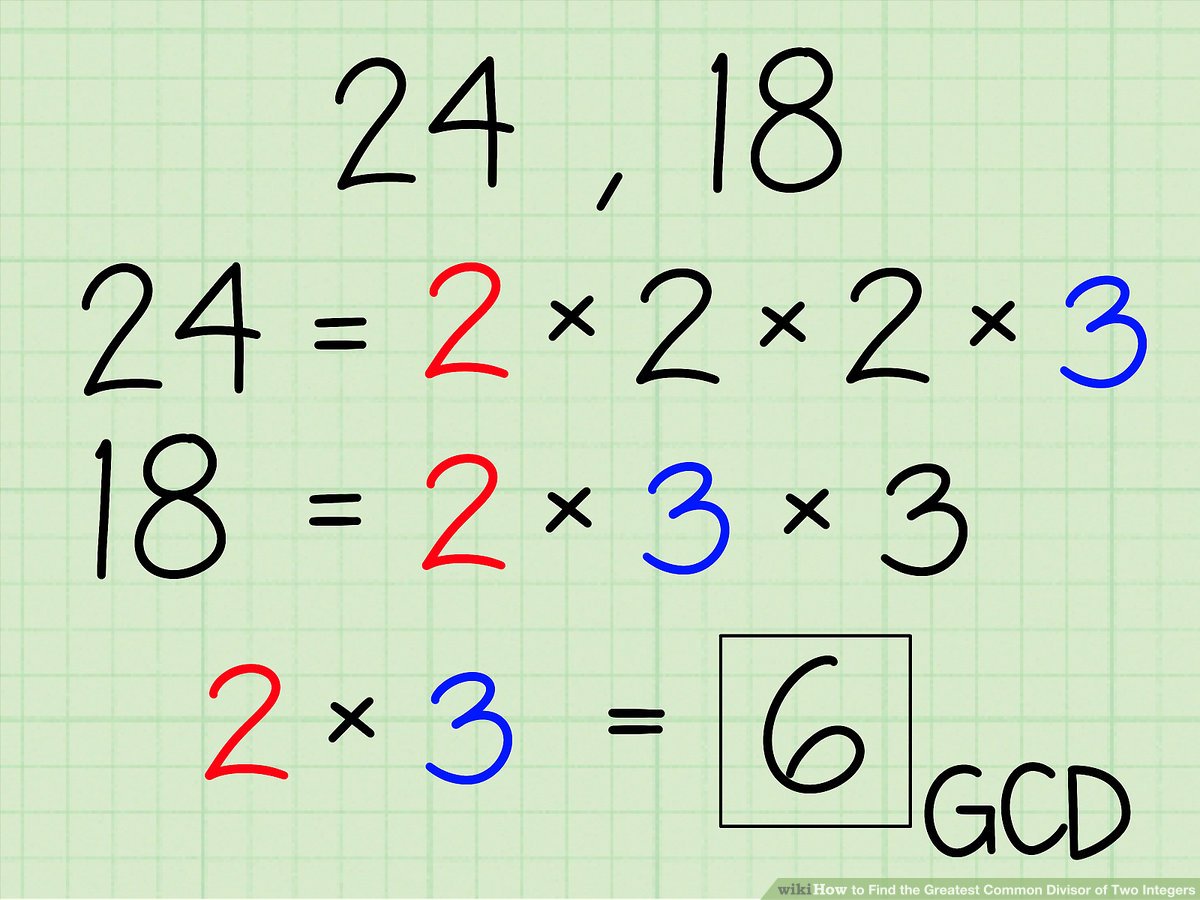
maxIndex = i;

}

}

return maxIndex;

}



public static int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

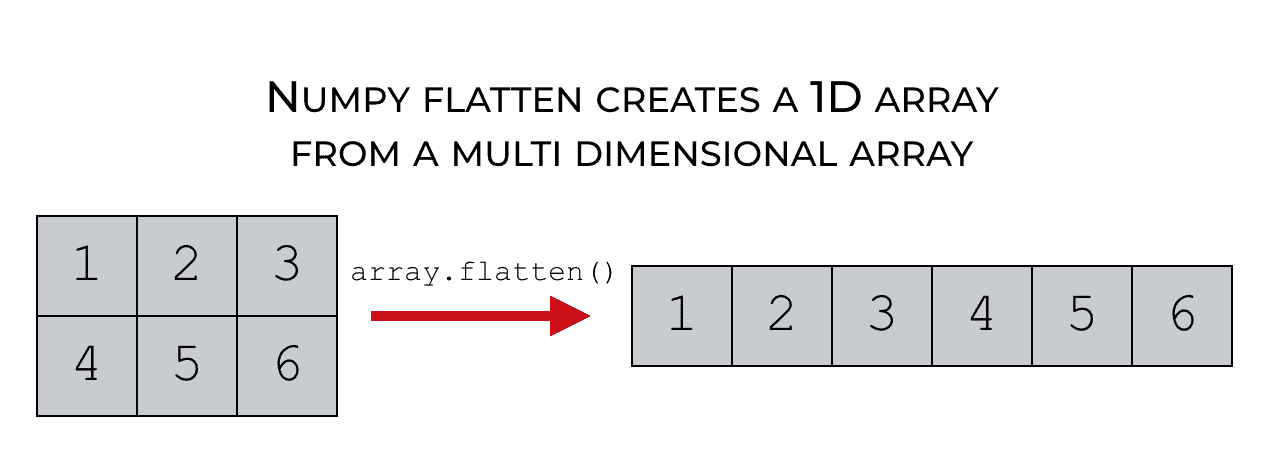
a = temp;

}

return a;

}

//Flatten matrix from 2D to one D array. E.g.



public static int[] flattenMatrix(int[][] matrix) {

int rows = matrix.length;

int cols = matrix[0].length; // assuming a rectangular matrix

int[] flatArray = new int[rows \* cols];

int index = 0;

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

flatArray[index++] = matrix[i][j];

}

}

return flatArray;

}

public static void printDigitsOneByOne(int number) {

while (number != 0) {

int digit = number % 10; // Extract the least significant digit

System.out.println(digit);

number = number / 10; // Remove the least significant digit

}

}

// sums 2 2d arrays together

public static int[][] sumMatrix(int[][] mtxA, int[][] mtxB) {

int numRows = mtxA.length;

int numCols = mtxA[0].length;

// Creating a new matrix to store the sum of mtxA and mtxB

int[][] mtxC = new int[numRows][numCols];

// Iterating through each element and adding corresponding elements from mtxA and mtxB

for (int row = 0; row < numRows; row++) {

for (int col = 0; col < numCols; col++) {

mtxC[row][col] = mtxA[row][col] + mtxB[row][col];

}

}

// Returning the resulting sum matrix

return mtxC;

}

*//Switch Statement*

switch(variable) {

case 1: variable = something;

break;

case 2: variable = something;

break;

case 3: variable = something;

break;

default: variable = “Invalid”;

break;

}

A close up of text

Description automatically generatedpublic static int enumerate(int n) {

if(n==1){

return 2;

}

if(n==2){

return 3;

}

int count\_h=enumerate(n-1);

int count\_f=enumerate(n-2);

return count\_h+count\_f;

//modify the ignore array to remove whatever you need to remove from a string

public static String cleanString(String str) {

String[] ignore = new String[] { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", ",", "!", "." };

String[] splitted = str.split(" ");

String cleaned = "";

for (String x : splitted) {

for (String y : ignore) {

x = x.replace(y, "");

}

cleaned += x;

}

return cleaned.trim();

}