1: string input to primitive data types:

class Main {

public static void main(String[] args) {

// creates objects of wrapper class

Integer obj1 = Integer.valueOf(23);

Double obj2 = Double.valueOf(5.55);

Boolean obj3 = Boolean.valueOf(true);

// converts into primitive types

int var1 = obj1.intValue();

double var2 = obj2.doubleValue();

boolean var3 = obj3.booleanValue();

// print the primitive values

System.out.println("The value of int variable: " + var1);

System.out.println("The value of double variable: " + var2);

System.out.println("The value of boolean variable: " + var3);

}

}

3: arithmetic:

import java.util.Scanner;

class Main {

public static void main(String[] args) {

char operator;

Double number1, number2, result;

// create an object of Scanner class

Scanner input = new Scanner(System.in);

// ask users to enter operator

System.out.println("Choose an operator: +, -, \*, or /");

operator = input.next().charAt(0);

// ask users to enter numbers

System.out.println("Enter first number");

number1 = input.nextDouble();

System.out.println("Enter second number");

number2 = input.nextDouble();

switch (operator) {

// performs addition between numbers

case '+':

result = number1 + number2;

System.out.println(number1 + " + " + number2 + " = " + result);

break;

// performs subtraction between numbers

case '-':

result = number1 - number2;

System.out.println(number1 + " - " + number2 + " = " + result);

break;

// performs multiplication between numbers

case '\*':

result = number1 \* number2;

System.out.println(number1 + " \* " + number2 + " = " + result);

break;

// performs division between numbers

case '/':

result = number1 / number2;

System.out.println(number1 + " / " + number2 + " = " + result);

break;

default:

System.out.println("Invalid operator!");

break;

}

input.close();

}

}

21: read nd write:

/ Java program to read content from one file

// and write it into another file

// Custom paths for this program

// Reading from - gfgInput.txt

// Writing to - gfgOutput.txt

// Importing input output classes

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

// Class

class GFG {

// Main driver method

public static void main(String[] args)

{

// The file reading process may sometimes give

// IOException

// Try block to check for exceptions

try {

// Creating a FileReader object and

// file to be read is passed as in parameters

// from the local directory of computer

FileReader fr = new FileReader("gfgInput.txt");

// FileReader will open that file from that

// directory, if there is no file found it will

// through an IOException

// Creating a FileWriter object

FileWriter fw = new FileWriter("gfgOutput.txt");

// It will create a new file with name

// "gfgOutput.text", if it is already available,

// then it will open that instead

// Declaring a blank string in which

// whole content of file is to be stored

String str = "";

int i;

// read() method will read the file character by

// character and print it until it end the end

// of the file

// Condition check

// Reading the file using read() method which

// returns -1 at EOF while reading

while ((i = fr.read()) != -1) {

// Storing every character in the string

str += (char)i;

}

// Print and display the string that

// contains file data

System.out.println(str);

// Writing above string data to

// FileWriter object

fw.write(str);

// Closing the file using close() method

// of Reader class which closes the stream &

// release resources that were busy in stream

fr.close();

fw.close();

// Display message

System.out.println(

"File reading and writing both done");

}

// Catch block to handle the exception

catch (IOException e) {

// If there is no file in specified path or

// any other error occured during runtime

// then it will print IOException

// Display message

System.out.println(

"There are some IOException");

}

}

}

21: APPEND:

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.nio.file.StandardOpenOption;

public class AppendFile {

public static void main(String[] args) {

String path = System.getProperty("user.dir") + "\\src\\test.txt";

String text = "Added text";

try {

Files.write(Paths.get(path), text.getBytes(), StandardOpenOption.APPEND);

} catch (IOException e) {

}

}

}

31: LONGEST INCRESING SUBSEQUENCE

public class Demo{

static int incre\_subseq(int my\_arr[], int arr\_len){

int seq\_arr[] = new int[arr\_len];

int i, j, max = 0;

for (i = 0; i < arr\_len; i++)

seq\_arr[i] = 1;

for (i = 1; i < arr\_len; i++)

for (j = 0; j < i; j++)

if (my\_arr[i] > my\_arr[j] && seq\_arr[i] < seq\_arr[j] + 1)

seq\_arr[i] = seq\_arr[j] + 1;

for (i = 0; i < arr\_len; i++)

if (max < seq\_arr[i])

max = seq\_arr[i];

return max;

}

public static void main(String args[]){

int my\_arr[] = { 10, 22, 9, 33, 21, 50, 41, 60 };

int arr\_len = my\_arr.length;

System.out.println("The length of the longest increasing subsequence is " + incre\_subseq(my\_arr, arr\_len));

}

ANOTHER ONE

import java.util.Scanner;

/\* Class LongestIncreasingSubsequence \*/

public class LongestIncreasingSubsequence

{

/\* function lis \*/

public int[] lis(int[] X)

{

int n = X.length - 1;

int[] M = new int[n + 1];

int[] P = new int[n + 1];

int L = 0;

for (int i = 1; i < n + 1; i++)

{

int j = 0;

/\*\* Linear search applied here. Binary Search can be applied too.

binary search for the largest positive j <= L such that

X[M[j]] < X[i] (or set j = 0 if no such value exists) \*\*/

for (int pos = L ; pos >= 1; pos--)

{

if (X[M[pos]] < X[i])

{

j = pos;

break;

}

}

P[i] = M[j];

if (j == L || X[i] < X[M[j + 1]])

{

M[j + 1] = i;

L = Math.max(L,j + 1);

}

}

/\* backtrack \*/

int[] result = new int[L];

int pos = M[L];

for (int i = L - 1; i >= 0; i--)

{

result[i] = X[pos];

pos = P[pos];

}

return result;

}

/\* Main Function \*/

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

System.out.println("Longest Increasing Subsequence Algorithm Test\n");

System.out.println("Enter number of elements");

int n = scan.nextInt();

int[] arr = new int[n + 1];

System.out.println("\nEnter "+ n +" elements");

for (int i = 1; i <= n; i++)

arr[i] = scan.nextInt();

LongestIncreasingSubsequence obj = new LongestIncreasingSubsequence();

int[] result = obj.lis(arr);

/\* print result \*/

System.out.print("\nLongest Increasing Subsequence : ");

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

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

System.out.println();

}

}