Ques.1 Write a program to find the greatest number among three given numbers using conditional operator.

Ans.

import java.util.Scanner;

public class JavaExample

{

public static void main(String[] args)

{

int num1, num2, num3, result, temp;

/\* Scanner is used for getting user input.

\* The nextInt() method of scanner reads the

\* integer entered by user.

\*/

Scanner scanner = new Scanner(System.in);

System.out.println("Enter First Number:");

num1 = scanner.nextInt();

System.out.println("Enter Second Number:");

num2 = scanner.nextInt();

System.out.println("Enter Third Number:");

num3 = scanner.nextInt();

scanner.close();

/\* In first step we are comparing only num1 and

\* num2 and storing the largest number into the

\* temp variable and then comparing the temp and

\* num3 to get final result.

\*/

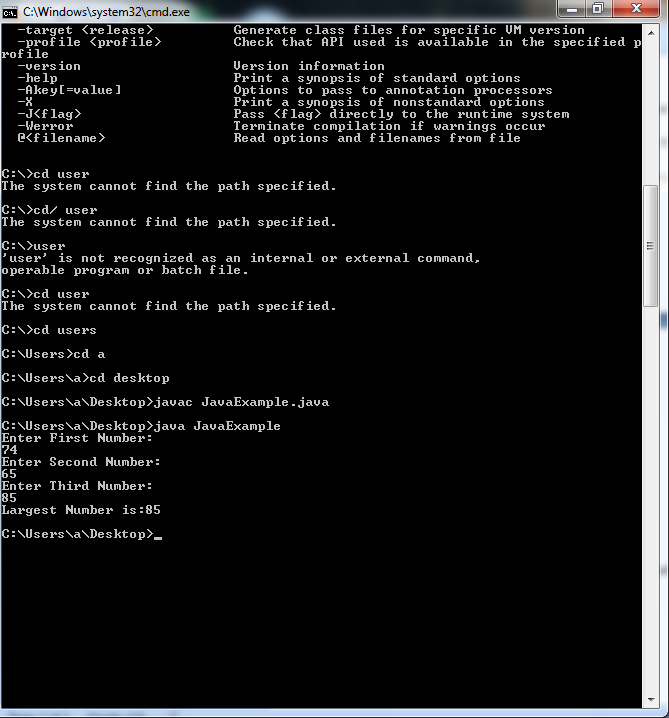
temp = num1>num2 ? num1:num2;

result = num3>temp ? num3:temp;

System.out.println("Largest Number is:"+result);

}

}

Output: 

Ques2. Write a program to display: 1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

Ans. public class Pattern {

public static void main(String[] args) {

char last = 'E', alphabet = 'A';

for(int i = 1; i <= (last-'A'+1); ++i) {

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

System.out.print(alphabet + " ");

}

++alphabet;

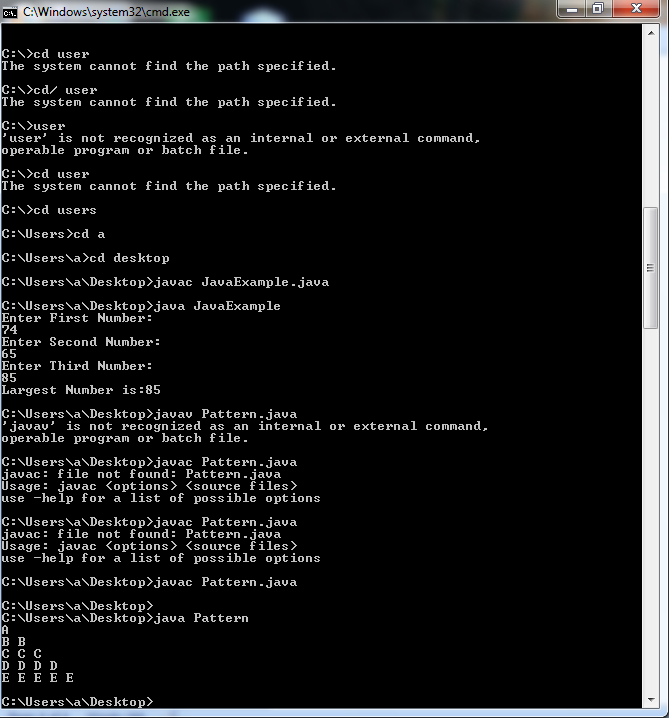
System.out.println();

}

}

}

Output:



Ques3. Write a program to demonstrate a two dimensional array with row sum.

Ans. class Lab4 {

public static void main(String[] args) {

int [][] scores = {{ 20, 18, 23, 20, 16 },

{ 30, 20, 18, 21, 20 },

{ 16, 19, 16, 53, 24 },

{ 25, 24, 22, 24, 25 }};

outputArray(scores);

}

public static void outputArray(int[][] array) {

int sum= 0;

int rowSize = array.length;

int columnSize = array[0].length;

System.out.println("rows=" + rowSize + "cols=" + columnSize);

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

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

sum += array[i][j];

}

System.out.println("Print the sum of rows = " + sum);

}

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

sum = 0;

sum = sum + array.length;

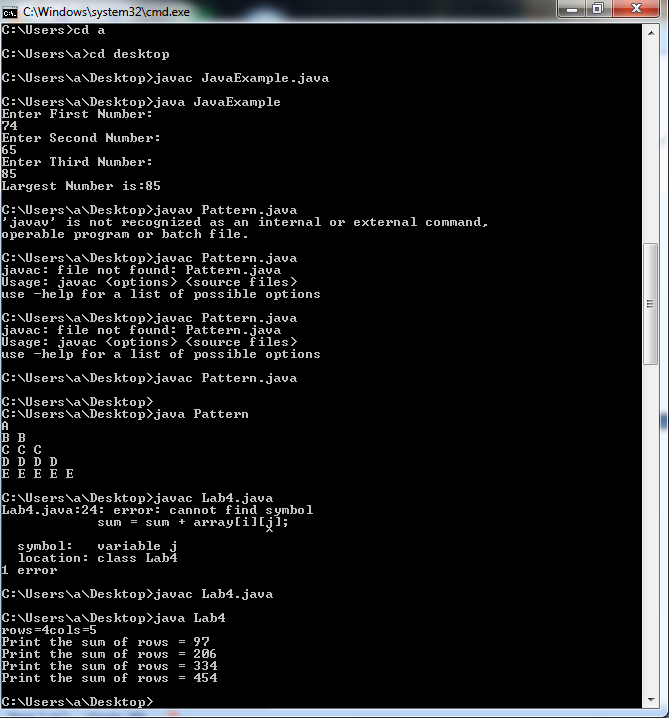
// It is telling me the j can't be resolved

}

}

}

Output:



Ques4. Write a program to demonstrate constructor overloading.

Ans4. class StudentData

{

private int stuID;

private String stuName;

private int stuAge;

StudentData()

{

//Default constructor

stuID = 100;

stuName = "New Student";

stuAge = 18;

}

StudentData(int num1, String str, int num2)

{

//Parameterized constructor

stuID = num1;

stuName = str;

stuAge = num2;

}

//Getter and setter methods

public int getStuID() {

return stuID;

}

public void setStuID(int stuID) {

this.stuID = stuID;

}

public String getStuName() {

return stuName;

}

public void setStuName(String stuName) {

this.stuName = stuName;

}

public int getStuAge() {

return stuAge;

}

public void setStuAge(int stuAge) {

this.stuAge = stuAge;

}

public static void main(String args[])

{

//This object creation would call the default constructor

StudentData myobj = new StudentData();

System.out.println("Student Name is: "+myobj.getStuName());

System.out.println("Student Age is: "+myobj.getStuAge());

System.out.println("Student ID is: "+myobj.getStuID());

/\*This object creation would call the parameterized

\* constructor StudentData(int, String, int)\*/

StudentData myobj2 = new StudentData(555, "Chaitanya", 25);

System.out.println("Student Name is: "+myobj2.getStuName());

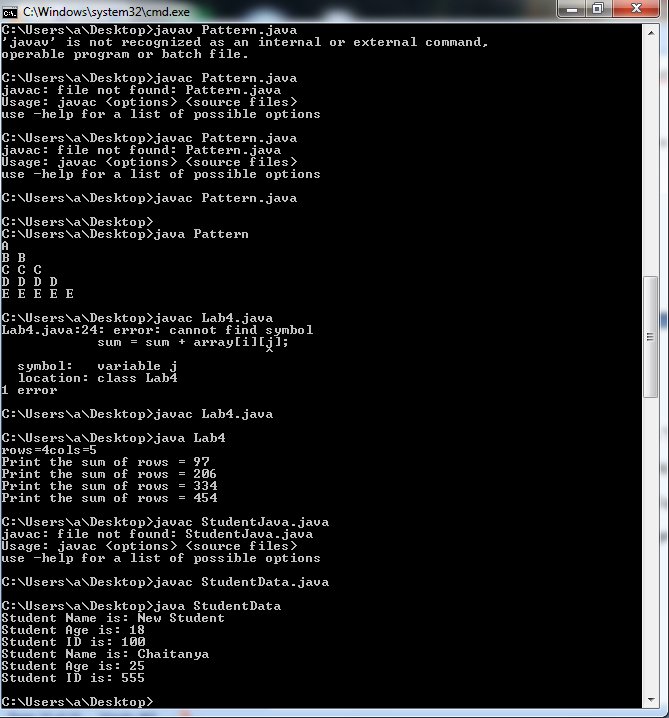
System.out.println("Student Age is: "+myobj2.getStuAge());

System.out.println("Student ID is: "+myobj2.getStuID());

}

}

Output:



Ques9. Write a program to demonstrate the try, catch and finally block in exception handling.

Ans9.

class TestExceptions {

static void myMethod(int testnum) throws Exception {

System.out.println ("start - myMethod");

if (testnum == 12)

throw new Exception();

System.out.println("end - myMethod");

return;

}

public static void main(String args[]) {

int testnum = 12;

try {

System.out.println("try - first statement");

myMethod(testnum);

System.out.println("try - last statement");

}

catch ( Exception ex) {

System.out.println("An Exception");

}

finally {

System. out. println( "finally") ;

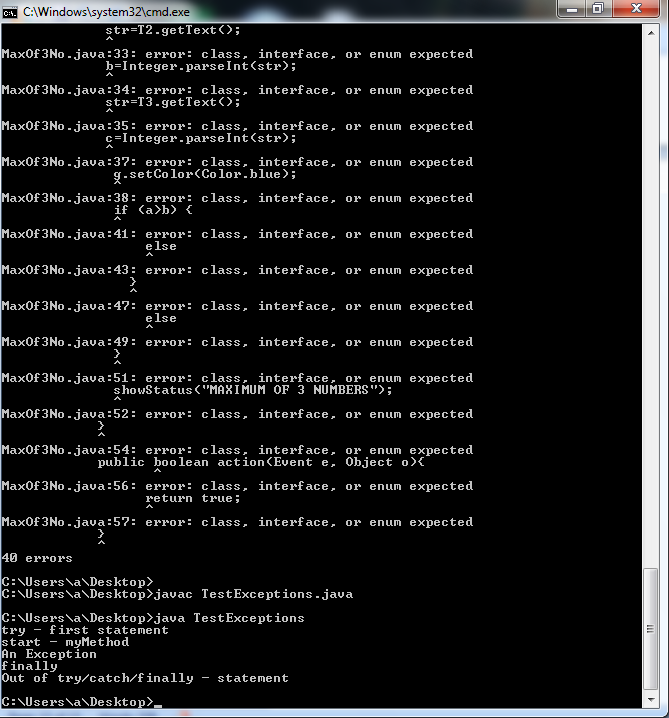
}

System.out.println("Out of try/catch/finally - statement");

}

}

Output:



Ques8. Write a program to construct multiplication table of 3 using multithreading.

Ans8.

public class Multiplicatin\_Table implements Runnable{

private int number;

public Multiplicatin\_Table(int number) {

this.number=number;

}

@Override

public void run() {

// TODO Auto-generated method stub

for (int i = 1; i <= 10; i++) {

System.out.printf("%s: %d \* %d = %d\n", Thread.currentThread().getName(),

number, i, i \* number);

}

}

public static void main(String[] args) {

// TODO Auto-generated method stub

System.out.println("I will print table of 1 to 3 ");

for (int i = 1; i <= 3; i++) {

Multiplicatin\_Table mul = new Multiplicatin\_Table(i);

Thread thread = new Thread(mul);

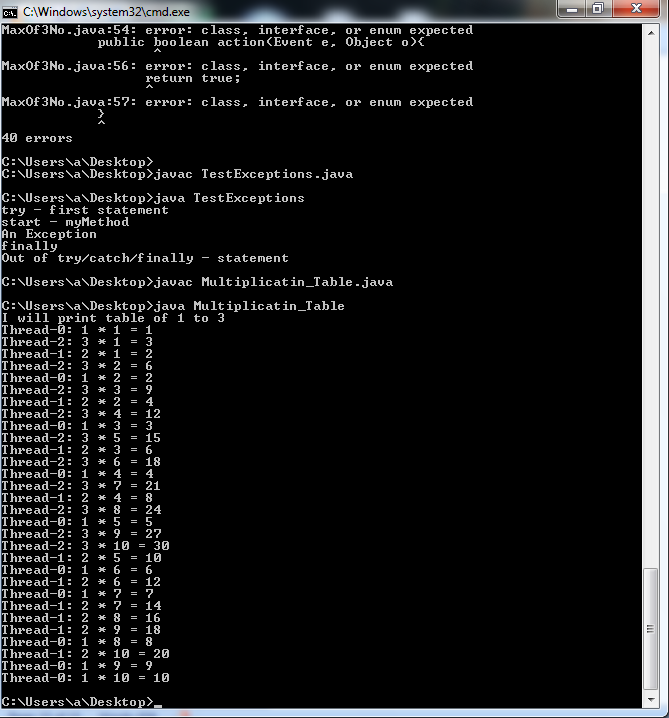
thread.start();

}

}

}

Output:



Ques6. Write a program to demonstrate multilevel inheritance.

Ans6.

class MultilevelInheritance

{

protected String str;

MultilevelInheritance() {

str = "This ";

}

}

class ChildClass1 extends MultilevelInheritance {

ChildClass1() {

str = str.concat("is ");

}

}

class ChildClass2 extends ChildClass1 {

ChildClass2() {

str = str.concat("Multilevel Inheritance ");

}

}

class ChildClass3 extends ChildClass2 {

ChildClass3() {

str = str.concat("Example.");

}

void display() {

System.out.println(str);

}

}

class MultilevelInheritanceMain {

public static void main(String[] args) {

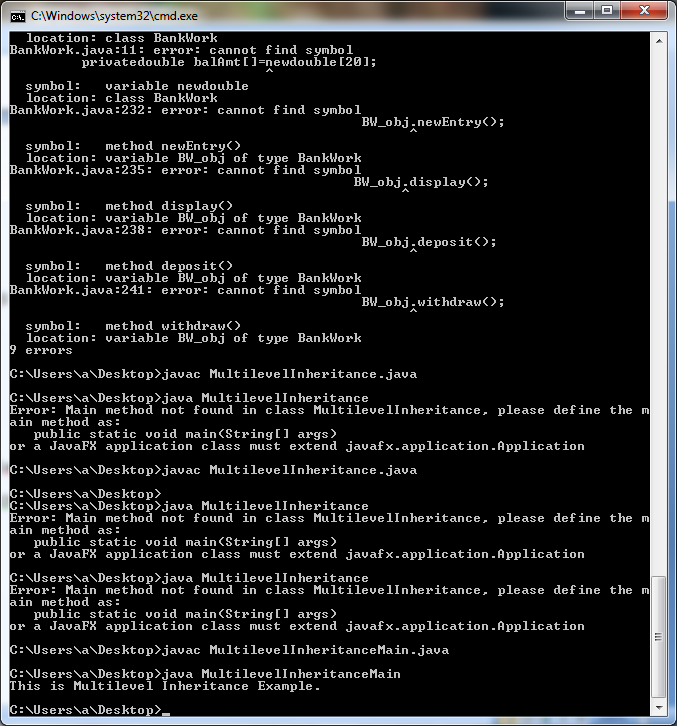
ChildClass3 obj = new ChildClass3();

obj.display();

}

}

Output:



Ques7. Give an example where interface can be used to support Multiple Inheritance.

Ans.7.

interface X

{

public void myMethod();

}

interface Y

{

public void myMethod();

}

class JavaExample implements X, Y

{

public void myMethod()

{

System.out.println("Implementing more than one interfaces");

}

public static void main(String args[]){

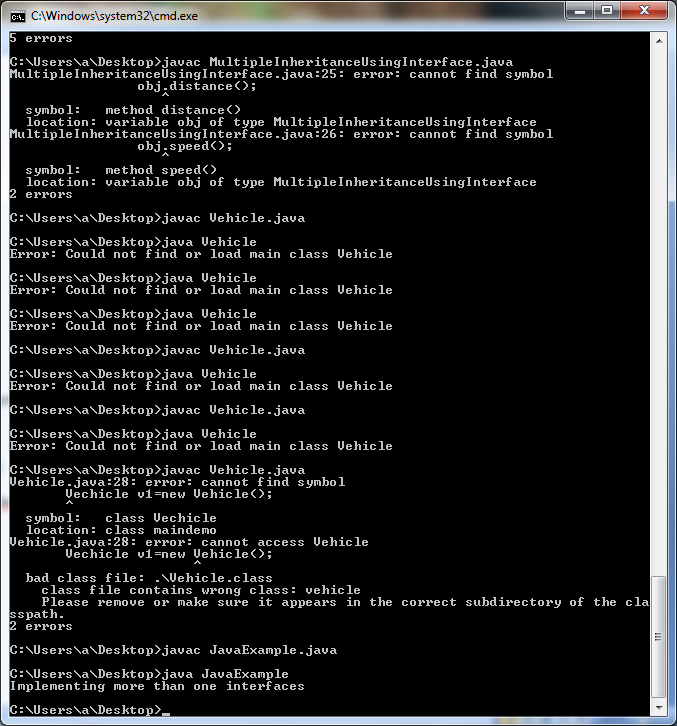
JavaExample obj = new JavaExample();

obj.myMethod();

}

}

Output:



Ques10. Develop an applet that receive three numeric values as input from the user and then displays the largest of the three on the screen. Write an HTML page and test the applet

Ans.

import java.awt.\*;

import java.applet.\*;

public class DisplayNumericalValues extends Applet

{

public void paint(Graphics g)

{

int val1 = 10;

int val2 = 20;

int sum = val1 + val2;

String str\_sum = "Sum="+String.valueOf(sum);

g.drawString(str\_sum,100,200);

}

}

This applet runs using the following HTML file

<HTML>

<HEAD>

<TITLE>Display Numerical Values</TITLE>

</HEAD>

<BODY>

<APPLET Code="DisplayNumericalValues.class" Width=400 Height=300>

</APPLET>

</BODY>

</HTML>

<HTML>

<HEAD>

<TITLE>Display Numerical Values</TITLE>

</HEAD>

<BODY>

<APPLET Code="DisplayNumericalValues.class" Width=400 Height=300>

</APPLET>

</BODY>

</HTML>

Output:

|  |
| --- |
| [https://4.bp.blogspot.com/-g1JPECdCK5Q/UtaeWh5XErI/AAAAAAAAFmY/wJArE11YVfg/s640/DisplayNumericalValues-comiple.jpg](https://4.bp.blogspot.com/-g1JPECdCK5Q/UtaeWh5XErI/AAAAAAAAFmY/wJArE11YVfg/s1600/DisplayNumericalValues-comiple.jpg) |
|  |

[](https://3.bp.blogspot.com/-BmoXMfXo1DE/UtaefzqnxzI/AAAAAAAAFmg/XUVFsWZ_HcE/s1600/DisplayNumericalValues-applet.jpg)