**Program 1**

29/04/2021

**Aim:** Define a class ‘product’ with data members pcode,

pname and price. Create 3 objects of the class and find

the product having the lowest price.

**Source Code:**

**package** packoops;

**public** **class** products {

**int** pcode;

String pname;

**double** price;

**double** lowest;

**void** data(**int** c, String n, **double** p){

pcode=c;

pname=n;

price=p;

}

**void** display(){

System.***out***.println(pcode+"\t\t"+pname+"\t\t"+price);

}

**static** **void** lowest(**double** price1,**double** price2, **double** price3){

**if**(price1<=price2 && price1<=price3){

System.***out***.println("\nProduct1 is the lowest price!");

}

**else** **if**(price2<=price1 && price2<=price3){

System.***out***.println("\nProduct2 is the lowest price!");

}

**else**{

System.***out***.println("\nProduct3 is the lowest price!");

}

}

**public** **static** **void** main(String[] args){

products obj1 = **new** products();

products obj2 = **new** products();

products obj3 = **new** products();

obj1.data(1,"Product1",112);

obj2.data(2,"Product2",1200);

obj3.data(3,"Product3",324.4);

System.***out***.println("Product Detail:\nProduct Code\tProduct Name\tProduct Price");

obj1.display();

obj2.display();

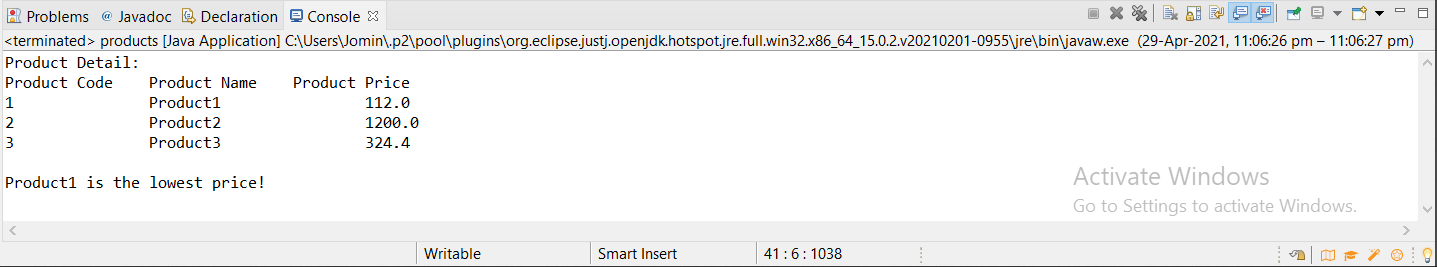
obj3.display();

*lowest*(obj1.price,obj2.price,obj3.price);

}

}

# Output :



**Program 2**

29/04/2021

**Aim:** Read 2 matrices from the console and perform matrix

addition.

**Source Code:**

**package** packoops;

**import** java.util.\*;

**public** **class** matrix {

**int** row;

**int** column;

**int**[][] array = **new** **int**[10][10];

**public** **void** get\_metrix(){

**int** rc,cc;

Scanner sc= **new** Scanner(System.***in***);

System.***out***.print("Enter the number of row : ");

**this**.row = sc.nextInt();

System.***out***.print("Enter the number of column : ");

**this**.column = sc.nextInt();

System.***out***.print("Enter matrix elements : ");

**for**(rc=0;rc<**this**.row;rc++){

**for**(cc=0;cc<**this**.column;cc++){

**this**.array[rc][cc] = sc.nextInt();

}

}

}

**public** **static** matrix sum(matrix c1, matrix c2) {

**int** rc, cc;

matrix temp = **new** matrix();

**if** (c1.row == c2.row && c1.column == c2.column) {

temp.row =c1.row;

temp.column = c1.column;

**for** (rc = 0; rc < c1.row; rc++) {

**for** (cc = 0; cc < c1.column; cc++) {

temp.array[rc][cc] = c1.array[rc][cc] + c2.array[rc][cc];

}

}

}

**else** {

System.***out***.println("Order of matrixs is not same ");

}

**return** temp;

}

**public** **void** display\_matrix(){

**int** rc,cc;

**for**(rc=0;rc<**this**.row;rc++){

**for**(cc=0;cc<**this**.column;cc++){

System.***out***.print(**this**.array[rc][cc] + "\t" );

}

System.***out***.println("");

}

}

**public** **static** **void** main(String[] args) {

matrix first = **new** matrix();

matrix second = **new** matrix();

matrix temp = *sum*(first, second);

first.get\_metrix();

second.get\_metrix();

temp = *sum*(first,second);

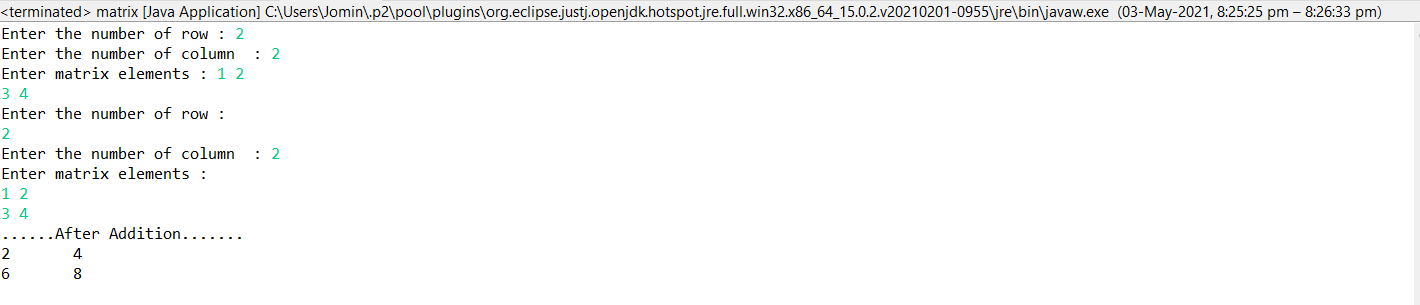
System.***out***.println("......After Addition.......");

temp.display\_matrix();

}

}

**Output:**

****

**Program 3**

03/05/2021

**Aim:** Add complex numbers.

**Source Code:**

**package** packoops;

**public** **class** complexadd {

**double** real, img;

complexadd(**double** r, **double** i){

**this**.real = r;

**this**.img = i;

}

**public** **static** complexadd sum(complexadd c1, complexadd c2)

{

complexadd temp = **new** complexadd(0, 0);

temp.real = c1.real + c2.real;

temp.img = c1.img + c2.img;

**return** temp;

}

**public** **static** **void** main(String args[]) {

complexadd c1 = **new** complexadd(7.5, 2);

complexadd c2 = **new** complexadd(1.2, 3.5);

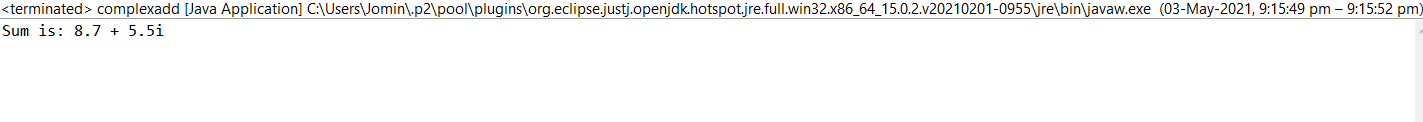
complexadd temp = *sum*(c1, c2);

System.***out***.printf("Sum is: "+ temp.real+" + "+ temp.img +"i");

}

}

**Output:**

****

**Program 4**

03/05/2021

**Aim:** Read a matrix from the console and check whether it is

symmetric or not.

**Source Code:**

**package** packoops;

**import** java.util.Scanner;

**public** **class** symmetricmatrix {

**public** **static** **void** main(String[] args) {

**int** x=0;

Scanner scan=**new** Scanner(System.***in***);

System.***out***.println("Enter the number of rows in matrix");

**int** row=scan.nextInt();

System.***out***.println("Enter the number of columns in matrix");

**int** col=scan.nextInt();

**if**(row!=col) {

System.***out***.println("Cannot find Symmetry for this matrix");

}

**else** {

**int**[][] a=**new** **int**[10][10];

System.***out***.println("Enter the values of matrix");

{

**for**(**int** i=0;i<row;i++) {

**for**(**int** j=0;j<col;j++) {

a[i][j]=scan.nextInt();

}

}

}

**for**(**int** i=0;i<row;i++) {

**for**(**int** j=0;j<col;j++) {

**if**(a[i][j]!=a[j][i]) {

x=x+1;

**break**;

}

**if**(x>0) {

**break**;

}

}

}

**if**(x==0) {

System.***out***.println("It is a Symmetric Matrix");

}

**else** {

System.***out***.println("It is not a Symmetric Matrix");

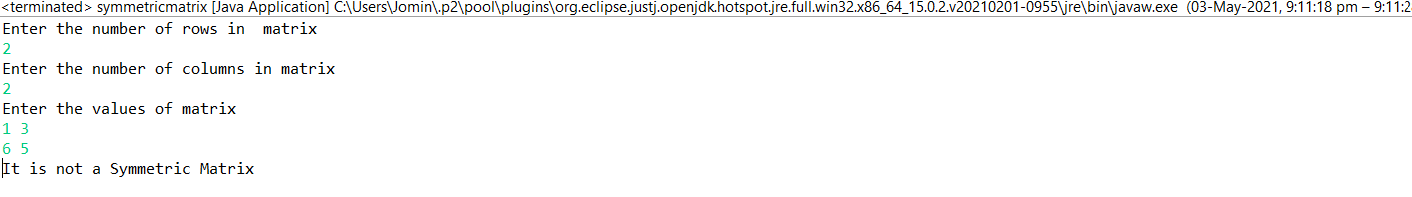
}

}

}

}

**Output:**

****

**Program 5**

06/05/2021

**Aim:** Create CPU with attribute price. Create inner class

Processor (no. of cores, manufacturer) and static nested

class RAM (memory, manufacturer). Create an object

of CPU and print information of processor and RAM.

**Source Code:**

**package** packoops;

**public** **class** cpu {

**int** price;

**public** **class** processor

{

**int** no\_of\_core;

String manufacturer;

}

**static** **class** RAM

{

**int** memory;

String manufacturer;

}

**public** **void** display(**int** x,**int** y,**int** z,String g, String f)

{ System.***out***.println("\n Processor");

System.***out***.println("\n Price-> "+x+"\n"+"Number of cores-> "+y+"\n"+"Manufacturer-> "+z);

System.***out***.println("\n RAM ");

System.***out***.println(" \n Memory: "+g+"\n"+"Manufacturer: "+z);

}

**public** **static** **void** main(String[] args) {

**int** x,y,z;

String g,f;

cpu obj=**new** cpu();

cpu.processor obj1=obj.**new** processor();

x=obj.price=17964;

y=obj1.no\_of\_core=6;

g=obj1.manufacturer="Ryzen";

cpu.RAM obj2=**new** RAM();

z=obj2.memory=16;

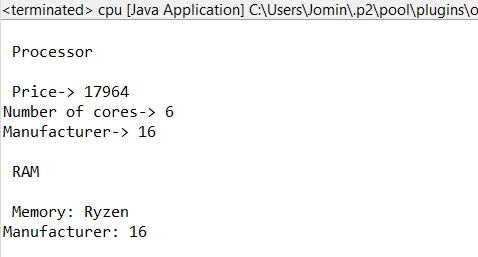
f=obj2.manufacturer="Crucial";

obj.display(x,y,z,g,f);

}

}

**Output:**

****