**PROGRAM 1: MATRIX ADDITION**

import java.util.Scanner;

public class Program

{

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

int m=3,i,j;

int a[][]=new int[m][m];

int b[][]=new int[m][m];

int c[][]=new int[m][m];

for(i=0;i<m;i++){

for(j=0;j<m;j++){

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

}

}

for(i=0;i<m;i++){

for(j=0;j<m;j++){

b[i][j]=input.nextInt();

}

}

for(i=0;i<m;i++){

for(j=0;j<m;j++){

c[i][j]=a[i][j]+b[i][j];

}

}

for(i=0;i<m;i++){

for(j=0;j<m;j++){

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

}

System.out.print("\n");

}

}

}

INPUT : 1 2 3

A = 4 5 6

7 8 9

9 8 7

B = 6 5 4

3 2 1

OUTPUT : 10 10 10

C = 10 10 10

10 10 10

**PROGRAM 2 : MATRIX MULTIPLICATION**

import java.util.Scanner;

public class Program

{

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

int m=3,i,j,k;

int a[][]=new int[m][m];

int b[][]=new int[m][m];

int c[][]=new int[m][m];

for(i=0;i<m;i++){

for(j=0;j<m;j++){

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

}

}

for(i=0;i<m;i++){

for(j=0;j<m;j++){

b[i][j]=input.nextInt();

}

}

for(i=0;i<m;i++){

for(j=0;j<m;j++){

for(k=0;k<m;k++){

c[i][j]+=a[i][k]\*b[k][j];

}

}

}

for(i=0;i<m;i++){

for(j=0;j<m;j++){

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

}

System.out.print("\n");

}

INPUT : 1 2 3

A = 4 5 6

7 8 9

9 8 7

B = 6 5 4

3 2 1

OUTPUT : 30 24 18

C = 84 69 54

138 114 90

**PROGRAM 3 : OVERLOADING METHOD**

import java.util.Scanner;

class Adder{

static int add(int a,int b){

return a+b;

}

static int add(int a,int b,int c){

return a+b+c;

}

}

class TestOverloading1{

public static void main(String arr[]){

Scanner input=new Scanner(System.in);

int a=input.nextInt();

int b=input.nextInt();

int c=input.nextInt();

System.out.println(Adder.add(a,b));

System.out.println(Adder.add(a,b,c));

}

}

INPUT : 11 22 33

OUTPUT : 33 66

**PROGRAM 4 : CREATE CLASS POINT**

class Point

{

int x,y;

void setPoint( ) {

x = 10;

y = 10;

}

}

class Demonstration\_35

{

float distance;

public static void main (String args[ ]) {

Point p = new Point( );

p.setPoint();

System.out.println ( " x = "+ p.x );

System.out.println ( " y = "+ p.y );

}

}

**OUTPUT :** x= 10

Y=10

**PROGRAM 5 : VARIABLE METHOD**

class Calculate{

static int cube(int x){

return x\*x\*x;

}

public static void main(String args[]){

int result=Calculate.cube(5);

System.out.println(result);

}

}

**OUTPUT :** 125

1. **What is conditional statement?**

A **conditional statement** is a statement that computer programming language used to decide which code has to be run when the true condition is met or which code has not to be run when the true condition is not met.

Syntax : if (condition) statement;

Or

if (condition) statement 1;

else statement 2;

1. **Write the syntax of switch case statement.**

**Switch** (expression) {

**case** value-1: Block-1; Break;

**case** value-2: Block-2; Break;

**case** value-n: Block-n; Break;

default: Block-1; Break;

}

1. **Write the difference between break and continue statement**

|  |  |
| --- | --- |
| **Break** | **continue** |
| A break can appear in both switch and loop (for, while, do) statements. | A continue can appear only in loop (for, while, do) statements. |
| A break causes the switch or loop statements to terminate the moment it is executed. Loop or switch ends abruptly when break is encountered. | A continue doesn't terminate the loop, it causes the loop to go to the next iteration. All iterations of the loop are executed even if continue is encountered. The continue statement is used to skip statements in the loop that appear after the continue. |
| The break statement can be used in both switch and loop statements. | The continue statement can appear only in loops. You will get an error if this appears in switch statement. |
| When a break statement is encountered, it terminates the block and gets the control out of the switch or loop. | When a continue statement is encountered, it gets the control to the next iteration of the loop. |
| A break causes the innermost enclosing loop or switch to be exited immediately. | A continue inside a loop nested within a switch causes the next loop iteration. |

1. **What is looping statement?**

A loop statement is a series of steps or sequence of statements executed repeatedly zero or more times satisfying the given condition is satisfied. Loop statements in programming languages, such as assembly languages or PERL make use of LABEL's to execute the statement repeatedly.

Types of loop statements: 1. for loop

2. while loop

3. do-while loop

SYNTAX : for loop

for (initialization, condition, increment)

{

------------------------------------------;

------------------------------------------;

------------------------------------------;

}

SYNTAX : while loop

while(contidion) {

statements;

}

SYNTAX : do-while loop

do{  
 statements;   
 } while( Condition );

1. **Write the difference between while and do-while statement**

| **BASIS FOR COMPARISON** | **WHILE** | **DO-WHILE** |
| --- | --- | --- |
| General Form | while (condition) { statements; //body of loop } | do{ statements; // body of loop. } while( Condition ); |
| Controlling Condition | In 'while' loop the controlling condition appears at the start of the loop. | In 'do-while' loop the controlling condition appears at the end of the loop. |
| Iterations | The iterations do not occur if, the condition at the first iteration, appears false. | The iteration occurs at least once even if the condition is false at the first iteration. |
| Alternate name | Entry-controlled loop | Exit-controlled loop |
| Semi-colon | Not used | Used at the end of the loop |

1. **What is array? How it is created?**

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. To declare an array, define the variable type with **square brackets.**

**Ex : String names[]={“ramu”, “ravi”, “sitha”}**

**int numbers[]={1,2,3,4,5,6}**

1. **What is class?**

Classes and Objects are basic concepts of Object Oriented Programming which revolve around the real life entities. A class is a user defined blueprint or prototype from which objects are created.  It represents the set of properties or methods that are common to all objects of one type.

1. **What is constructor?**

A constructor in Java is a special method that is used to initialize objects. The constructor is called when an object of a class is created. Fields are variables that provides the state of the class and its objects, and methods are used to implement the behavior of the class and its objects.

1. **What is the use of copy constructor?**

A copy constructor in a Java class is a constructor that creates an object using another object of the same Java class. That's helpful when we want to copy a complex object that has several fields, or when we want to make a deep copy of an existing object.

1. **What is the use of this keyword?**

The this keyword refers to the current object in a method or constructor. The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name (because a class attribute is shadowed by a method or constructor parameter).

1. **What is method overloading?**

**Method Overloading** is a feature that allows a class to have more than one **method** having the same name, if their argument lists are different. It is similar to constructor **overloading in Java**, that allows a class to have more than one constructor having different argument lists.

1. **What is static variable?**

Static variable in Java is variable which belongs to the class and initialized only once at the start of the execution. It is a variable which belongs to the class and not to object (instance) . Static variables are initialized only once, at the start of the execution.

1. **What is access modifier?**

The access modifiers in Java specifies the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and class by applying the access modifier on it.

There are four types access modifiers:

1. Private
2. Default
3. Protected
4. Public
5. **Write the difference between instance and static methods.**

* Instance method are methods which require an object of its class to be created before it can be called. Static methods are the methods in Java that can be called without creating an object of class.
* Static method is declared with static keyword. Instance method is not with static keyword.
* Static method means which will exist as a single copy for a class. But instance methods exist as multiple copies depending on the number of instances created for that class.
* Static methods can be invoked by using class reference. Instance or non static methods are invoked by using object reference.
* Static methods can’t access instance methods and instance variables directly. Instance method can access static variables and static methods directly

1. **What is object? How it is created?**

It is a basic unit of Object Oriented Programming and represents the real life entities.  A typical Java program creates many objects, which as you know, interact by invoking methods. An object consists of:

**State**: It is represented by attributes of an object. It also reflects the properties of an object.

**Behavior**: It is represented by methods of an object. It also reflects the response of an object with other objects.

**Identity**: It gives a unique name to an object and enables one object to interact with other objects