1**.Matrix Addition:**

Import java.util.Scanner;  
 public class Program  
 {  
    public static void main(String[] args) {  
    int m1,n1,m2,n2;     
    Scanner s=new Scanner(System.in);  
    m1=s.nextInt();  
    n1=s.nextInt();  
    m2=s.nextInt();  
    n2=s.nextInt();  
    int a[][]=new int[m1][n1];  
    int b[][]=new int[m2][n2];  
    for(int i=0;i<m1;i++){  
        for(int j=0;j<n1;j++){  
            a[i][j]=s.nextInt();  
        }  
    }  
    for(int i=0;i<m2;i++){  
        for(int j=0;j<n2;j++){  
            b[i][j]=s.nextInt();  
        }  
    }  
    if(m1==m2 && n1==n2){  
        for(int i=0;i<m1;i++){  
            for(int j=0;j<n1;j++){  
                a[i][j]=a[i][j]+b[i][j];  
                System.out.print(a[i][j]+"\t\t\t");  
            }  
           System.out.println();    
     }  
    }  
       else{  
           System.out.print("matrix addition is not possible");  
       }  
    }  
}

**Output:**

**Input:** 2(m1)

3(n1)

2(m2)

3(n2)

1 2 3 1 2 3

4 5 6 4 5 6

**Output:** 2 4 6

8 10 12

2.**Matrix Multiplication:**

import java.util.Scanner;  
public class Program  
{  
    public static void main(String[] args) {  
    int m1,n1,m2,n2;     
    Scanner s=new Scanner(System.in);  
    m1=s.nextInt();  
    n1=s.nextInt();  
    m2=s.nextInt();  
    n2=s.nextInt();  
    int a[][]=new int[m1][n1];  
    int b[][]=new int[m2][n2];  
    for(int i=0;i<m1;i++){  
        for(int j=0;j<n1;j++){  
            a[i][j]=s.nextInt();  
       }  
   }  
   for(int i=0;i<m2;i++){  
        for(int j=0;j<n2;j++){  
            b[i][j]=s.nextInt();  
        }  
    }  
    if(n1==m2){  
        for(int i=0;i<m1;i++){  
            for(int j=0;j<n2;j++){  
             int q=0;  
            for(int k=0;k<m2;k++){             
                q+=a[i][k]\*b[k][j];  
             }  
           System.out.print(q+"\t\t\t");  
       }  
           System.out.println();    
     }  
     }  
      else{  
           System.out.print("matrix multiplication is not possible");  
       }  
    }  
}

**Output:**

**Input:** 2(m1)

3(n1)

3(m2)

2(n2)

1 2 1 1 1

3 4 1 1 1

1 0

**Output**: 4 3

8 7

3.**Write a java program to demonstrate method overloading?**

import java.util.Scanner;  
class MethodOverloading{  
    public void add(){  
        System.out.println("inputs are not provided");  
    }  
    public void add(int a,float b){  
        System.out.println(a+b);  
    }  
    public void add(float a,int b){  
        System.out.println(a+b);  
    }     
}  
public class Program  
{  
    public static void main(String[] args) {  
       MethodOverloading s=new MethodOverloading();  
       s.add();  
       Scanner q=new Scanner(System.in);  
       int a=q.nextInt();  
       float b=q.nextFloat();  
       s.add(a,b);  
       s.add(b,a);  
    }  
}

**Output:- Input:** 5

2.2

**Output:** inputs are not provided

7.2

7.2

4**.Create a class point with two data members x & y. Include all Constructors and display().**

import java.util.Scanner;  
public class Con  
{  
int a;  
int b;  
Con(){  
    System.out.println("no arguments constructor");  
}  
Con(int x,int y){  
    this.a=x;  
    this.b=y;  
}  
public void dis(){  
    System.out.print(this.a+this.b);  
}  
    public static void main(String[] args) {  
    Scanner q=new Scanner(System.in);  
    int a=q.nextInt();  
    int b=q.nextInt();  
        Con s=new Con();  
        Con d=new Con(a,b);  
        d.dis();  
     }  
  }

**Output:-Input:** 3

5

**Output:** no arguments constructor

8

5.**Write a java program using static method?**

import java.util.Scanner;  
class Static{  
    static void add(int a,int b){  
        System.out.println(a+b);  
    }  
}  
public class Program  
{  
    public static void main(String[] args) {   
    int a,b;  
    Scanner s=new Scanner(System.in);  
    a=s.nextInt();  
    b=s.nextInt();  
    Static.add(a,b);  
    }  
}

**Output**:

**Input:** 3

16

**Output:**19

1**.What is conditional statements?**

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.

Java has the following conditional statements:

\*Use if  to specify a block of code to be executed, if a specified condition is true.

\*Use else to specify a block of code to be executed, if the same condition is false.

\*Use else if to specify a new condition to test, if the first condition is false.

\*Use switch to specify many alternative blocks of code to be executed.

If statement:

Syntax:

If (condition) {

// block of code to be executed if the condition is true

}

Else statement:

Syntax:

If (condition) {

// block of code to be executed if the condition is true

} else {

// block of code to be executed if the condition is false

}

Else if Statement:

Syntax:

If (condition1) {

// block of code to be executed if condition1 is true

} else if (condition2) {

// block of code to be executed if the condition1 is false and

condition2 is true

} else {

//block of code to be executed if the condition1 is false and

condition2 is false

}

Switch statement:

The switch expression is evaluated once and the value of the expression is compared with the values of each case.

Syntax:

Switch(expression) {

case x:

//code block

Break;

case y:

//code block

default:

//code block

}

2**. Syntax of Switch case statement**

Switch statement:

The switch expression is evaluated once and the value of the expression is compared with the values of each case.

Syntax:

Switch(expression) {

case x:

//code block

Break;

case y:

//code block

default:

//code block

}

3**. Difference between BREAK and CONTINUE statement:**

Break leaves the loop completely and executes the statements after the loop. Whereas continue leaves the current iteration and executes with the next value in the loop.

Break completely exits the loop. continue skips the statements after the continue statement and keeps looping.

 Break statement resumes the control of the program to the end of loop and made executional flow outside that loop. Continue statement  resumes the control of the program to the next iteration of that loop enclosing continue and made executional flow inside the loop again.

4**. Looping Statement:**

Looping statement are the statements execute one or more statement repeatedly several number of times. There are three types of loops:- while, for and do-while.

**While loop:**

In while loop in Java first check the condition if condition is true then control goes inside the loop body otherwise goes outside of the body. while loop will be repeats in clock wise direction.

**For Loop:**

For loop allows code to be repeatedly executed. For loop contains 3 parts initialization, Condition and Increment or Decrements.

\* Initialization: This allows to declare and initialize any loop control variables.

\*  Condition: After initialization, if it is true, the body of the loop is executed, if it is false then the body of the loop does not execute and flow of control goes outside of the for loop.

\* Increment or Decrements: After completion of Initialization and Condition steps loop body code is executed and then Increment or Decrements steps is executed. This statement allows to update any loop control variables.

**do-while loop:**

A do-while loop is similar to a while loop, except that a do-while loop is execute at least one time.

A do while loop is a control flow statement that executes a block of code at least once, and then repeatedly executes the block, or not, depending on a given condition at the end of the block.

5**.Difference between while and do…while statement:**

In while loop, the condition is tested at the beginning of the loop, and if the condition is True, then only statements in that loop will be executed. So, the While loop executes the code block only if the condition is True.

In do while loop, the condition is tested at the end of the loop. So, the Do While executes the statements in the code block at least once even if the condition Fails.

A do-while loop is similar to a while loop, except that a do-while loop is execute at least one time.

6**.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.

String[ ] cars;

\*To create an array of integers, you could write:

int[ ] myNum = {10, 20, 30, 40};

we declared a variable that holds an array of integers by placing the values in a comma-separated list, inside curly braces.

7**.What is Class:**

Everything in Java is associated with classes and objects, along with its attributes and methods. class is a collection of objects. For Example - In real life, a car is an object. The car has attributes, such as weight and colour, and methods such as drive and brake. A Class is like an object constructor, or a "blueprint" for creating objects.

8.**What is Constructor?**

A constructor is a special method that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes.

A constructor resembles an instance method but it’s not a method as it doesn’t have a return type. It has the same name as its class and is syntactically similar to a method.

9**.Use of Copy Constructor?**

Copy constructors are widely used for creating duplicates of objects known as cloned objects.

Duplicate object in the sense the object will have the same characteristics of the original object from which duplicate object is created. But both original and duplicate objects refer to different memory locations. To initialize the values of instance variables of a class we create a parameterized constructor accepting the values for all instance variables and initialize them with the given values.

10**.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.

 \* this can also be used to:

Invoke current class constructor

Invoke current class method

Return the current class object

Pass an argument in the method call

Pass an argument in the constructor call

11**. Method overloading?**

Method overloading is a concept that allows to declare multiple methods with same name but different parameters in the same class. Java supports method overloading and always occur in the same class.

Method overloading is one of the ways through which java supports polymorphism.

Method overloading can be done by changing number of arguments or by changing the data type of arguments.

If two or more method have same name and same parameter list but differs in return type cannot be overloaded.

12**. Static variable?**

Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.

There would only be one copy of each class variable per class, regardless of how many objects are created from it.

Static variables are rarely used other than being declared as constants. Constants are variables that are declared as public/private, final, and static. Constant variables never change from their initial value.

Static variables are stored in the static memory. It is rare to use static variables other than declared final and used as either public or private constants.

Static variables are created when the program starts and destroyed when the program stops.

Static variables can be accessed by calling with the class name ClassName.VariableName.

13**. What is access modifier?**

There are two types of modifiers: access modifiers and non-access modifiers. The access modifiers 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 of access modifiers:

Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.

Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.

Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

14**.Difference between instance and static methods?**

Static methods as name states defined at the class level and could be accessed on the class name that is no need of class object creation in order to access the static methods. If we do not use the static keyword with method than it belongs or categorized as instance method which is defined at instance level and need class object for their accessibility.

Static methods exist as a single copy for a class while instance methods exist as multiple copies depending on the number of instances created for that particular class. Static methods can't access instance methods directly while instance methods can access static variables and static methods directly.

15**. What is Object? How it is created?**

An entity that has state and behaviour is known as an object ex: chair, bike, marker, pen, table, car, etc. It can be physical or logical.

An object has three characteristics:

State:  It represents the data of an object.

Behaviour: It represents the behaviour of an object such as deposit, withdraw, etc.

Identity: An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

A class provides the blueprints for objects. So, an object is created from a class. the new keyword is used to create new objects.

There are three steps when creating an object from a class −

Declaration − A variable declaration with a variable name with an object type.

Instantiation − The 'new' keyword is used to create the object.

Initialization − The 'new' keyword is followed by a call to a constructor. This call initializes the new object.