//Matrix Addition

1. **import** java.util.Scanner;
2. **public** **class** Add\_Matrix
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** p, q, m, n;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter number of rows in first matrix:");
9. p = s.nextInt();
10. System.out.print("Enter number of columns in first matrix:");
11. q = s.nextInt();
12. System.out.print("Enter number of rows in second matrix:");
13. m = s.nextInt();
14. System.out.print("Enter number of columns in second matrix:");
15. n = s.nextInt();
16. **if** (p == m && q == n)
17. {
18. **int** a[][] = **new** **int**[p][q];
19. **int** b[][] = **new** **int**[m][n];
20. **int** c[][] = **new** **int**[m][n];
21. System.out.println("Enter all the elements of first matrix:");
22. **for** (**int** i = 0; i < p; i++)
23. {
24. **for** (**int** j = 0; j < q; j++)
25. {
26. a[i][j] = s.nextInt();
27. }
28. }
29. System.out.println("Enter all the elements of second matrix:");
30. **for** (**int** i = 0; i < m; i++)
31. {
32. **for** (**int** j = 0; j < n; j++)
33. {
34. b[i][j] = s.nextInt();
35. }
36. }
37. System.out.println("First Matrix:");
38. **for** (**int** i = 0; i < p; i++)
39. {
40. **for** (**int** j = 0; j < q; j++)
41. {
42. System.out.print(a[i][j]+" ");
43. }
44. System.out.println("");
45. }
46. System.out.println("Second Matrix:");
47. **for** (**int** i = 0; i < m; i++)
48. {
49. **for** (**int** j = 0; j < n; j++)
50. {
51. System.out.print(b[i][j]+" ");
52. }
53. System.out.println("");
54. }
55. **for** (**int** i = 0; i < p; i++)
56. {
57. **for** (**int** j = 0; j < n; j++)
58. {
59. **for** (**int** k = 0; k < q; k++)
60. {
61. c[i][j] = a[i][j] + b[i][j];
62. }
63. }
64. }
65. System.out.println("Matrix after addition:");
66. **for** (**int** i = 0; i < p; i++)
67. {
68. **for** (**int** j = 0; j < n; j++)
69. {
70. System.out.print(c[i][j]+" ");
71. }
72. System.out.println("");
73. }
74. }
75. **else**
76. {
77. System.out.println("Addition would not be possible");
78. }
79. }
80. }
81. **import** java.util.Scanner;
82. **public** **class** Odd\_Even
83. {
84. **public** **static** **void** main(String[] args)
85. {
86. **int** n;
87. Scanner s = **new** Scanner(System.in);
88. System.out.print("Enter the number you want to check:");
89. n = s.nextInt();
90. **if**(n % 2 == 0)
91. {
92. System.out.println("The given number "+n+" is Even ");
93. }
94. **else**
95. {
96. System.out.println("The given number "+n+" is Odd ");
97. }
98. }
99. }

//Largest of 3 numbers

1. **import** java.util.Scanner;
2. **public** **class** Biggest\_Number
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** x, y, z;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter the first number:");
9. x = s.nextInt();
10. System.out.print("Enter the second number:");
11. y = s.nextInt();
12. System.out.print("Enter the third number:");
13. z = s.nextInt();
14. **if**(x > y && x > z)
15. {
16. System.out.println("Largest number is:"+x);
17. }
18. **else** **if**(y > z)
19. {
20. System.out.println("Largest number is:"+y);
21. }
22. **else**
23. {
24. System.out.println("Largest number is:"+z);
25. }
27. }
28. }

//Equals Program in java

1. **import** java.util.Scanner;
2. **public** **class** Equal\_Integer
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** m, n;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter the first number:");
9. m = s.nextInt();
10. System.out.print("Enter the second number:");
11. n = s.nextInt();
12. **if**(m == n)
13. {
14. System.out.println(m+" and "+n+" are equal ");
15. }
16. **else**
17. {
18. System.out.println(m+" and "+n+" are not equal ");
19. }
20. }
21. }

//Digits extraction in java

1. **import** java.util.Scanner;
2. **public** **class** Extract\_Digits
3. {
4. **public** **static** **void** main(String args[])
5. {
6. **int** n, m, a, i = 1, counter = 0;
7. Scanner s=**new** Scanner(System.in);
8. System.out.print("Enter any number:");
9. n = s.nextInt();
10. m = n;
11. **while**(n > 0)
12. {
13. n = n / 10;
14. counter++;
15. }
16. **while**(m > 0)
17. {
18. a = m % 10;
19. System.out.println("Digits at position "+counter+":"+a);
20. m = m / 10;
21. counter--;
22. }
23. }
24. }

//Integer Palindrome in java

**import** java.util.Scanner;

**public** **class** Palindrome

{

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

{

**int** n, m, rev = 0, x;

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

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

n = s.nextInt();

m = n;

**while**(n > 0)

{

x = n % 10;

rev = rev \* 10 + x;

n = n / 10;

}

**if**(rev == m)

{

System.out.println(" "+m+" is a palindrome number");

}

**else**

{

System.out.println(" "+m+" is not a palindrome number");

}

}

}

//Linear Search in java

1. **import** java.util.Scanner;
3. **class** LinearSearchExample2
4. {
5. **public** **static** **void** main(String args[])
6. {
7. **int** c, n, search, array[];
9. Scanner in = **new** Scanner(System.in);
10. System.out.println("Enter number of elements");
11. n = in.nextInt();
12. array = **new** **int**[n];
14. System.out.println("Enter those " + n + " elements");
16. **for** (c = 0; c < n; c++)
17. array[c] = in.nextInt();
19. System.out.println("Enter value to find");
20. search = in.nextInt();
22. **for** (c = 0; c < n; c++)
23. {
24. **if** (array[c] == search)     /\* Searching element is present \*/
25. {
26. System.out.println(search + " is present at location " + (c + 1) + ".");
27. **break**;
28. }
29. }
30. **if** (c == n)  /\* Element to search isn't present \*/
31. System.out.println(search + " isn't present in array.");
32. }
33. }

//Binary Search in java

1. **class** BinarySearchExample{
2. **public** **static** **void** binarySearch(**int** arr[], **int** first, **int** last, **int** key){
3. **int** mid = (first + last)/2;
4. **while**( first <= last ){
5. **if** ( arr[mid] < key ){
6. first = mid + 1;
7. }**else** **if** ( arr[mid] == key ){
8. System.out.println("Element is found at index: " + mid);
9. **break**;
10. }**else**{
11. last = mid - 1;
12. }
13. mid = (first + last)/2;
14. }
15. **if** ( first > last ){
16. System.out.println("Element is not found!");
17. }
18. }
19. **public** **static** **void** main(String args[]){
20. **int** arr[] = {10,20,30,40,50};
21. **int** key = 30;
22. **int** last=arr.length-1;
23. binarySearch(arr,0,last,key);
24. }
25. }

//Binary search in java Using recursion

1. **class** BinarySearchExample1{
2. **public** **static** **int** binarySearch(**int** arr[], **int** first, **int** last, **int** key){
3. **if** (last>=first){
4. **int** mid = first + (last - first)/2;
5. **if** (arr[mid] == key){
6. **return** mid;
7. }
8. **if** (arr[mid] > key){
9. **return** binarySearch(arr, first, mid-1, key);//search in left subarray
10. }**else**{
11. **return** binarySearch(arr, mid+1, last, key);//search in right subarray
12. }
13. }
14. **return** -1;
15. }
16. **public** **static** **void** main(String args[]){
17. **int** arr[] = {10,20,30,40,50};
18. **int** key = 30;
19. **int** last=arr.length-1;
20. **int** result = binarySearch(arr,0,last,key);
21. **if** (result == -1)
22. System.out.println("Element is not found!");
23. **else**
24. System.out.println("Element is found at index: "+result);
25. }
26. }

//Print Fibonacci numbers

import java.util.Scanner;

public class Main

{

public static void main(String[] args)

{

int n, a = 0, b = 0, c = 1;

Scanner s = new Scanner(System.in);

System.out.print("Enter value of n:");

n = s.nextInt();

System.out.print("Fibonacci Series:");

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

{

a = b;

b = c;

c = a + b;

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

}

}

}

//Illustrate Various Boolean operators

1. **public** **static** **void** main(String args[])
2. {
3. Scanner s = **new** Scanner(System.in);
4. System.out.print("Enter a:");
5. **boolean** a = s.nextBoolean();
6. System.out.print("Enter b:");
7. **boolean** b = s.nextBoolean();
8. **boolean** c = a | b;
9. **boolean** d = a & b;
10. **boolean** e = a ^ b;
11. **boolean** f = (!a & b) | (a & !b);
12. **boolean** g = !a;
13. System.out.println("a = " + a);
14. System.out.println("b = " + b);
15. System.out.println("a|b = " + c);
16. System.out.println("a&b = " + d);
17. System.out.println("a^b = " + e);
18. System.out.println("!a&b|a&!b = " + f);
19. System.out.println("!a = " + g);
20. }
21. }

//Illustrate Various Boolean operators

1. **import** java.util.Scanner;
2. **public** **class** Bitwise\_Operation
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** m, n, x, a;
7. Scanner s = **new** Scanner(System.in);
8. System.out.print("Enter First number:");
9. m = s.nextInt();
10. System.out.print("Enter Second number:");
11. n = s.nextInt();
12. **while**(**true**)
13. {
14. System.out.println("");
15. System.out.println("Press 1 for Right Shift by 2:");
16. System.out.println("Press 2 for Left Shift by 2:");
17. System.out.println("Press 3 for Bitwise AND:");
18. System.out.println("Press 4 for Bitwise OR by 2:");
19. System.out.println("Press 5 for Bitwise Exclusive OR:");
20. System.out.println("Press 6 for Bitwise NOT:");
21. System.out.println("Press 7 to Exit:");
22. System.out.println("");
23. System.out.print("Option:");
24. x = s.nextInt();
25. **switch**(x)
26. {
27. **case** 1:
28. a = m << 2;
29. System.out.println("Result after left shift by 2:"+a);
30. **break**;
32. **case** 2:
33. a = n >> 2;
34. System.out.println("Result after right shift by 2:"+a);
35. **break**;
37. **case** 3:
38. a = m & n;
39. System.out.println("Result after bitwise AND:"+a);
40. **break**;
42. **case** 4:
43. a = m | n;
44. System.out.println("Result after bitwise OR:"+a);
45. **break**;
47. **case** 5:
48. a = m ^ n;
49. System.out.println("Result after bitwise Exclusive OR:"+a);
50. **break**;
52. **case** 6:
53. a = ~ m;
54. System.out.println("Result after bitwise NOT:"+a);
55. **break**;
57. **case** 7:
58. System.exit(0);
59. }
60. }
61. }
62. }

//Armstrong number using While loop in java

**import** java.util.Scanner;

**public** **class** ArmStrong

{

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

{

**int** n, count = 0, a, b, c, sum = 0;

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

System.out.print("Enter a number:");

n = s.nextInt();

a = n;

c = n;

**while**(a > 0)

{

a = a / 10;

count++;

}

**while**(n > 0)

{

b = n % 10;

sum = (**int**) (sum+Math.pow(b, count));

n = n / 10;

}

**if**(sum == c)

{

System.out.println(c+ " is an Armstrong number");

}

**else**

{

System.out.println(c+ " is not an Armstrong number");

}

}

}

//Armstrong number using for loop in java

**import** java.util.Scanner;

**public** **class** ArmStrong

{

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

{

**int** n, count = 0, a, b, c, sum = 0;

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

System.out.print("Enter a number:");

n = s.nextInt();

a = n;

c = n;

*// Calculate the number of digits in 'n' using a for loop*

**for** (; a > 0; a /= 10)

{

count++;

}

*// Reset 'a' and 'sum' for the following loop*

a = n;

sum = 0;

*// Calculate the sum of cubes of digits using a for loop*

**for** (; n > 0; n /= 10)

{

b = n % 10;

sum += Math.pow(b, count);

}

**if** (sum == c)

{

System.out.println(c + " is an Armstrong number");

}

**else**

{

System.out.println(c + " is not an Armstrong number");

}

}

}

**Examples of command-line argument**

**Example 1:**

* Java

|  |
| --- |
| // Java Program to Illustrate First Argument    // Class  **class** GFG {        // Main driver method  **public** **static** **void** main(String[] args)      {          // Printing the first argument          System.out.println(args[0]);      }  } |



Java Program to Check for Command Line Arguments

// Class

**class** GFG {

    // Main driver method

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

    {

        // Checking if length of args array is

        // greater than 0

**if** (args.length > 0) {

            // Print statements

            System.out.println("The command line"

                               + " arguments are:");

            // Iterating the args array

            // using for each loop

**for** (String val : args)

                // Printing command line arguments

                System.out.println(val);

        }

**else**

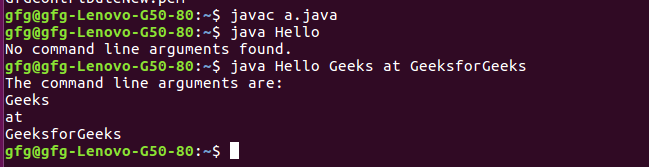
            // Print statements

            System.out.println("No command line "

                               + "arguments found.");

    }

}



## Local Variable Type Inference

Local Variable Type Inference is one of the most evident change to language available from Java 10 onwards. It allows to define a variable using var and without specifying the type of it. The compiler infers the type of the variable using the value provided. This type inference is restricted to local variables.

## Old way of declaring local variable.

String name = "Welcome to tutorialspoint.com";

## New Way of declaring local variable.

var name = "Welcome to tutorialspoint.com";

Now compiler infers the type of name variable as String by inspecting the value provided.

## Noteworthy points

* No type inference in case of member variable, method parameters, return values.
* Local variable should be initialized at time of declaration otherwise compiler will not be infer and will throw error.
* Local variable inference is available inside initialization block of loop statements.
* No runtime overhead. As compiler infers the type based on value provided, there is no performance loss.
* No dynamic type change. Once type of local variable is inferred it cannot be changed.
* Complex boilerplate code can be reduced using local variable type inference.

Map<Integer, String> mapNames = new HashMap<>();

var mapNames1 = new HashMap<Integer, String>();

### Example

Following Program shows the use of Local Variable Type Inference in JAVA 10.

import java.util.List;

public class Tester {

public static void main(String[] args) {

var names = List.of("Julie", "Robert", "Chris", "Joseph");

for (var name : names) {

System.out.println(name);

}

System.out.println("");

for (var i = 0; i < names.size(); i++) {

System.out.println(names.get(i));

}

}

}

### Output

Julie

Robert

Chris

Joseph

Julie

Robert

Chris

Joseph

# Jump Statements in Java

Jumping statements are control statements that transfer execution control from one point to another point in the program. There are three Jump statements that are provided in the Java programming language:

1. Break statement.
2. Continue statement.
3. Return Statement

### Break statement

**1. Using Break Statement to exit a loop:**

In java, the [break](https://www.geeksforgeeks.org/break-statement-in-java/) statement is used to terminate the execution of the nearest looping statement or switch statement. The break statement is widely used with the switch statement, **for**loop, **while**loop, **do-while**loop.

#### Syntax:

**break;**

// Java program to illustrate the

// break keyword in Java

**import** java.io.\*;

**class** GFG {

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

    {

**int** n = 10;

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

**if** (i == 6)

**break**;

            System.out.println(i);

        }

    }

}

**Output**

0

1

2

3

4

5

**Note :**In a switch statement, if the **break** statement is missing, every **case**label is executed till the end of the switch.

**2. Use Break as a form of goto**

Java does not have a goto statement because it produces an unstructured way to alter the flow of program execution. Java illustrates an extended form of the break statement. This form of **break** works with**the label.**The label is the name of a label that identifies a statement or a block of code.

#### **Syntax:**

**break label;**

When this form of **break**executes, control jumps out of the labeled statement or block.

**import** java.io.\*;

**class** GFG {

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

    {

**for** (**int** i = 0; i < 3; i++) {

        one : { // label one

        two : { // label two

        three : { // label three

            System.out.println("i=" + i);

**if** (i == 0)

**break** one; // break to label one

**if** (i == 1)

**break** two; // break to label two

**if** (i == 2)

**break** three; // break to label three

        }

            System.out.println("after label three");

        }

            System.out.println("after label two");

        }

            System.out.println("after label one");

        }

    }

}

### Continue Statement

The [**continue**](https://www.geeksforgeeks.org/continue-statement-in-java/)statement pushes the next repetition of the loop to take place, hopping any code between itself and the conditional expression that controls the loop.

Java program to illustrate the

// continue keyword in Java

**import** java.io.\*;

**class** GFG {

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

    {

**for** (**int** i = 0; i < 10; i++) {

**if** (i == 6){

                  System.out.println();

                  // using continue keyword

                  // to skip the current iteration

**continue**;

            }

            System.out.println(i);

        }

    }

}

**Output**

0

1

2

3

4

5

7

8

9

In the program, when the value of i is 6, the compiler encounters the continue statement, and then 6 is skipped.

**Return Statement**

The “[**return**](https://www.geeksforgeeks.org/return-keyword-java/)” keyword can help you transfer control from one method to the method that called it. Since the control jumps from one part of the program to another, the return is also a jump statement.

* **“**return**”**is a reserved keyword means we can’t use it as an identifier.
* It is used to exit from a method, with or without a value.

|  |
| --- |
| **import** java.io.\*;    **class** ReturnExample {      // A simple method that takes two integers as input and      // returns their sum  **public** **static** **int** calculateSum(**int** num1, **int** num2)      {          // Print a message indicating the method has started          System.out.println("Calculating the sum of " + num1                             + " and " + num2);  **int** sum = num1 + num2;          System.out.println("The sum is: " + sum);            // Return the calculated sum  **return** sum;            // Note: Any code after the 'return' statement will          // not be executed. But "Final" is an exception in          // the case of try-catch-final block.          // System.out.println("end"); // error : unreachable          // statement      }  **public** **static** **void** main(String[] args)      {          // Call the calculateSum method  **int** result = calculateSum(5, 10);            // Print the result          System.out.println("Result: " + result);      }  } |

**Output**

Calculating the sum of 5 and 10

The sum is: 15

Result: 15

Output Explanation:

When we are calling a class calculateSum method that has return sum which returns the value of sum and that value gets displayed on the console.

# Write a program in Java to develop user defined exception for ‘Divide by Zero’ error.

class MyException extends Exception

{

private int ex;

MyException(int **b**)

{

ex**=**b;

}

public String toString()

{

**return** "My Exception : Number is not divided by "**+**ex;

}

}

class DivideByZeroException

{

static void divide(int **a**,int **b**) throws MyException

{

**if**(b**<=**0)

{

**throw** **new** MyException(b);

}

**else**

{

System**.**out**.**println("Division : "**+**a**/**b);

}

}

public static void main(String **arg**[])

{

**try**

{

divide(10,0);

}

**catch**(MyException me)

{

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

}

}

}

// Resizable.java

// Interface Resizable

// Declare the Resizable interface

interface Resizable {

// Declare the abstract method "resizeWidth" to resize the width

void resizeWidth(int width);

// Declare the abstract method "resizeHeight" to resize the height

void resizeHeight(int height);

}

Copy

// Rectangle.java

// Declare the Rectangle class, which implements the Resizable interface

class Rectangle implements Resizable {

// Declare private instance variables to store width and height

private int width;

private int height;

// Constructor for initializing the width and height

public Rectangle(int width, int height) {

this.width = width;

this.height = height;

}

// Implement the "resizeWidth" method to resize the width

public void resizeWidth(int width) {

this.width = width;

}

// Implement the "resizeHeight" method to resize the height

public void resizeHeight(int height) {

this.height = height;

}

// Method to print the current width and height of the rectangle

public void printSize() {

System.out.println("Width: " + width + ", Height: " + height);

}

}

Copy

// Main.java

// Declare the Main class

public class Main {

public static void main(String[] args) {

// Create an instance of the Rectangle class with an initial size

Rectangle rectangle = new Rectangle(100, 150);

// Print the initial size of the rectangle

rectangle.printSize();

// Resize the rectangle by changing its width and height

rectangle.resizeWidth(150);

rectangle.resizeHeight(200);

// Print the updated size of the rectangle

rectangle.printSize();

}

}