Example:

Lab Program3: To print all prime numbers between a given range

```
import javax.swing.JOptionPane;
class lab3
{
        public static void main(String[] args)
        {
                int n,j,x,count;
                //n=Integer.parseInt(args[0]);
n=Integer.parseInt(JOptionPane.showInputDialog("Enter range"));
                for(int i=1;i<=n;i++)
                {
                x=i;
                count=0;
                j=2;
                while(j<x)
                {
                if(x%i==0)
                {
                if(x%j==0)
                {
                count=1;
                break;
                }
                j++;
                }
                if(count==0)
                System.out.println(x+"is prime");
                }
        }
}
```

3. **Jump Control Statements:** Java provides the following Jump Control statements.
   a) break statement
   b) continue statement
   c) return statement
   d) exit statement

a) ***break statement: break*** statement executed in a while, do-while, for aod switch statements causes immediate exit from that structure. It can be used in two ways as unlabeled break and labeled break statement.

Unlabeled break statement:

Syntax: break;

Example:

```
for(int k=1;k<=10;k++)
{
if(k= =5)
break;
System.out.println(k);
}
```

O/P: 1 2 3 4

Labeled break statement: To break out of nested structures, we can use the labeled break statements. This statement, when executed in a while, do-while and for statements causes immediate exit from that structure and any number of enclosing repetition structures. Program execution resumes with the first statement after the enclosing labeled compound statement.

Example:

```
class ex
{
    public static void main(String[] args)
    {
        stop:
        {
        for(int row=1;row<=10;row++)
        {
        for(int col=1;col<=5;col++)
        {
        if(row==3)
        break stop;
        System.out.print("* ");
```

```
            }
            system.out.println("\n");
            }
            }
        }
}
```

O/P: * * * * *
* * * * *

b) ***continue statement:*** continue statement executed in a while, do-while or for statements skip the remaining statements in the body of that structure and proceeds with the next iteration of the loop. It can be used in two ways as unlabeled continue and labeled continue statement.

Unlabeled continue statement:

Syntax:continue;

Example:
```
for(int k=1;k<=10;k++)
{
if(k= =5)
continue;
System.out.println(k);
}
```

O/P: 1 2 3 4 6 7 8 9 10

Labeled continue statement: The labeled continue statement when executed in a repetition structure skips the remaining statements in that structure body and any number of enclosing repetition on structure and proceeds with the next iteration of an enclosed labeled repetition.

Example:

```
class ex
{
public static void main(String[] args)
{
```

```
        stop:
        for(int row=1;row<=4;row++)
        {
        for(int col=1;col<=3;col++)
        {
        if(col==2)
        continue stop;
        System.out.print(col);
        }
        System.out.println("\n");
        }
    }
    }
```

O/P: 1111

c) ***return statement:*** The return statement is used to explicitly return from a method. This causes program control to transfer back to the caller of the method. The return statement immediately terminates the method in which it executes.

Syntax: **return;**

d) ***exit statement*:** The method exit( ) is defined in System package. The purpose of exit is to terminate the program with a specific exit code.

Syntax: **System.exit(exit_code);**

The program finish normally by placing exit_code as '0' other value means error.

Example: **System.exit(0);**

2.22 Arrays

An *Array* is a group of contiguous and related data items that share a common name. A particular value is indicated by writing a number called 'index number' or 'subscript' in square bracket after the array name. Arrays are classified into different types. They are

1. One-Dimensional Array

2. Two-Dimensional Array
3. Multi-Dimensional Array

One-Dimensional Array: ***One-Dimensional*** Array is a list of homogenious items can be given one variable name using only one subscript. Individual value of an array is called an element. Like an ordinary variable, arrays must be declared and created in the computer memory before they are used. Creation of an array contains two sections. They are 1. Declaration of the array 2. Creating memory locations.

```
Syntax:  type arrayname[ ];                  // declaration
         arrayname = new type[size];         // allocation of memory
                        (or)
         type arrayname[ ]=new type[size];
```

Where,

type defines the datatype of the variables stored in the array

arrayname defines the name of the array

new operator is used to allocate dynamic memory in the computer for the array

size defines the number of memory location of the array

```
Example:        int number[];
                number = new int[5];
```

It can also be initialize arrays automatically when they are declared. The syntax is as

```
Syntax:  type arrayname[]={list of values};
```

Where, list of values contains number of values separated by commas and surrounded by curly braces

```
Example:        type number[]={35,40,56,23};
```

In Java, all arrays store the allocated size in a variable named length. We can access the length of the array using the syntax

```
Syntax:  arrayname.length;
```

Example 1: Write a program to read 5 numbers and print the sum of it.

```
class arr1
{
        public static void main(String[] args)
        {
                int sum=0;
                int no[]=new int[10];
```

```
        for(int i=0;i<5;i++)
        {
        no[i]=Integer.parseInt(args[i]);
        sum=sum+no[i];
        }
        System.out.println("Total="+sum);
    }
}
```

Example 2: Write a program to sort the list of values in ascending order

```
class arr1
{
    public static void main(String[] args)
    {
        int temp;
        int no[]={34,78,23,11,99};
        for(int i=0;i<no.length-1;i++)
        {
        for(int j=i+1;j<no.length;j++)
        {
        if(no[i]>no[j])
        {
        temp=no[i];
        no[i]=no[j];
        no[j]=temp;
        }
        }
        }
        for(int i=0;i<no.length;i++)
        System.out.println(no[i]);
    }
}
```

Two-Dimensional Array: A Two-Dimensional Array is a collection of homogeneous data items that share a common name using two subscripts. It stores table of data. This representation is very useful for matrix representations as first subscript represents the number of rows and second subscript represents the number of columns.

Syntax:

```
type arrayname[][];                          // declaration
arrayname = new type[size1][size2];          // allocation of memory
                    (or)
type arrayname[][]=new type[size1][size2];
```

Where,

type defines the datatype of the variables stored in the array

arrayname defines the name of the array

new operator is used to allocate dynamic memory in the computer for the array

size1 defines the number of rows of memory location of the array

size2 defines the number of columns of memory location of the array

Example: int number[][];
number = new int[5][6];

It can also be initialize arrays automatically when they are declared. The syntax is as

Syntax:type arrayname[][]={list of values};

Or

type arrayname[][]={{row1 values},
{row2 values},
.. }

Lab Program 6: Java program to multiply two matrices

```
import javax.swing.JOptionPane;
class arr2
{
    public static void main(String[] args)
    {
    int m,n,p,q,i,j,k;
    int a[][]=new int [10][10];
    int b[][]=new int [10][10];
    int c[][]=new int [10][10];
    m=Integer.parseInt(JOptionPane.showInputDialog("Enter rows of
    Matrix A"));
    n=Integer.parseInt(JOptionPane.showInputDialog("Enter columns of
    Matrix A"));
```

```
p=Integer.parseInt(JOptionPane.showInputDialog("Enter rows of
Matrix B"));
q=Integer.parseInt(JOptionPane.showInputDialog("Enter columns of
Matrix B"));
if(n==p)
{
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
a[i][j]=Integer.parseInt(JOptionPane.showInputDialog("Enter
a["+i+"]["+j+"]:"));
}
for(i=0;i<p;i++)
{
for(j=0;j<q;j++)
b[i][j]=Integer.parseInt(JOptionPane.showInputDialog("Enter
b["+i+"]["+j+"]:"));
}
for(i=0;i<m;i++)
{
for(j=0;j<q;j++)
{
c[i][j]=0;
for(k=0;k<n;k++)
c[i][j]=c[i][j]+a[i][k]*b[k][j];
}
}
for(i=0;i<m;i++)
{
System.out.println();
for(j=0;j<q;j++)
System.out.print(" "+c[i][j]);
}
```

```
}
else
{
JOptionPane.showMessageDialog(null,"Multiplication not possible");
}
}
}
```

Multi-Dimensional Array: Multi-Dimensional Array is a list of values shared by a common name using Multiple subscripts. If the array contains three subscripts then the array is called as Three-Dimensional array and so on.

Syntax: type arrayname[][]…[]=new int[size1][size2]…[sizen];

Example: type number[][][]=new int[4][5][6];

2.23 Type conversion and Casting:

Java permits mixing of constants and variables of different types, but during execution it follows type conversions. In Java, type casting can be done in two ways. They are

1. Implicit Type Casting 2. Explicit Type Casting

1. ***Implicit Type Casting:*** In an expression, if the operands are of different types the lower type is automatically converted to the higher type before the operation preceeds by the Java Interpreter. The result is of the higher type. Such type of casting is called Implicit Type Casting. In Implicit Type Casting there is no loss of data. The following are some of implicit type castings with no loss of data

from	**to**
byte	short, char, int, long, float, double
short	char, int, long, float, double
char	int, long, float, double
int	long, float, double
long	float, double
float	double

2. ***Explicit Type Casting:*** If we want to force to convert the operands of an expression from one type to another type depending on user choice, such type of casting is called Explicit Type Casting. In Explicit Type Casting there is a chance of data

loss when we convert the operands from higher type to lower type. The process of converting the value is known as Casting a value. The general form of casting is as

Syntax: (datatype) expression;

Example: x = (int) 7.5;

Note: The final result of an expression is converted to the type of the variable before assigning to the left hand side variable. In such case following chances are introduced.

- float to int causes truncation of the fractional part
- double to float causes rounding of digits
- long to int causes dropping of the excess of higher order bits

2.24 Example programs

EXAMPLE :: 1

```
/*
 Demonstrate a block of code.

 Call this file "Test.java"
*/
class Test {
 public static void main(String args[]) {
  int x, y;

  y = 20;

  // the target of this loop is a block
  for(x = 0; x<10; x++) {
   System.out.println("This is x: " + x);
   System.out.println("This is y: " + y);
   y = y - 2;
  }
 }
}
```

OUTPUT:

```
C:\ >javac Test.java
```

```
C:\ >java Test
This is x=0
This is y=20
This is x=1
This is y=18
This is x=2
This is y=16
This is x=3
This is y=14
This is x=4
This is y=12
This is x=5
This is y=10
This is x=6
This is y=8
This is x=7
This is y=6
This is x=8
This is y=4
This is x=9
This is y=2
```

EXAMPLE :: 2

```
/*
  This is a simple Java program.
  Call this file "demo1.java".
*/
class Demo1 {
 // Your program begins with a call to main().
 public static void main(String args[]) {
  System.out.println("This is a simple Java program.");
 }
}
```

OUTPUT:

```
C:\ >javac Demo1.java
```

```
C:\ >java Demo1
This is a simple java program
```

EXAMPLE :: 3

```
/*
  Here is another short example.
  Call this file "demo2.java".
*/class Demo2 {
public static void main(String args[]) {
   int num; // this declares a variable called num

   num = 10; // this assigns num the value 10

   System.out.println("This is num: " + num);

   num = num * 2;

   System.out.print("The value of num * 2 is ");
   System.out.println(num);
  }
}
```

OUTPUT:

```
C:\ >javac Demo2.java
C:\ >java Demo2
This is num:10
The value of num * 2 is 20
```

EXAMPLE :: 4

```
/*
 Demonstrate the for loop.

 Call this file "Demo3.java".
*/
class Demo3 {
 public static void main(String args[]) {
```

```
  int x;

  for(x = 0; x<10; x = x+1)
    System.out.println("This is x: " + x);
  }
}
```

OUTPUT:

```
C:\ >javac Demo3.java
C:\ >java Demo3
This is x=0
This is x=1
This is x=2
This is x=3
This is x=4
This is x=5
This is x=6
This is x=7
This is x=8
This is x=9
```

EXAMPLE :: 5

```
/*
 Demonstrate the if.
  Call this file "Sample.java".
*/
class Sample {
 public static void main(String args[]) {
  int x, y;

  x = 10;
  y = 20;

  if(x < y) System.out.println("x is less than y");
```

```
    x = x * 2;
    if(x == y) System.out.println("x now equal to y");

    x = x * 2;
    if(x > y) System.out.println("x now greater than y");

    // this won't display anything
    if(x == y) System.out.println("you won't see this");
  }
}
```

OUTPUT:

```
C:\ >javac Sample.java
C:\ >java Sample
X is less than y
X now equal to y
X now greater than y
```

EXAMPLE :: 7

```
// Demonstrate a one-dimensional array.
class Array {
  public static void main(String args[]) {
    int month_days[];
    month_days = new int[12];
    month_days[0] = 31;
    month_days[1] = 28;
    month_days[2] = 31;
    month_days[3] = 30;
    month_days[4] = 31;
    month_days[5] = 30;
    month_days[6] = 31;
    month_days[7] = 31;
    month_days[8] = 30;
    month_days[9] = 31;
    month_days[10] = 30;
    month_days[11] = 31;
    System.out.println("April has " + month_days[3] + " days.");
  }
}
```

OUTPUT:

```
C:\ >javac Array.java
C:\ >java Array
April has 30 days
```

EXAMPLE :: 8

```
// An improvied version of the previous program.
class Test1 {
 public static void main(String args[]) {
  int month_days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
  System.out.println("April has " + month_days[3] + " days.");
 }
}
```

OUTPUT:

```
C:\ >javac Test1.java
C:\ >java Test1
April has 30 days
```

EXAMPLE :: 9

```
// Average an array of values.
class Average {
 public static void main(String args[]) {
  double nums[] = {10.1, 11.2, 12.3, 13.4, 14.5};
  double result = 0;
  int i;

 for(i=0; i<5; i++)
   result = result + nums[i];

  System.out.println("Average is " + result / 5);
 }
}
```

OUTPUT:

```
C:\ >javac Average.java
C:\ >java Average
Average is 12.299999999999999
```

EXAMPLE :: 10

```
// Demonstrate boolean values.
class Simple {
  public static void main(String args[]) {
    boolean b;

    b = false;
    System.out.println("b is " + b);
    b = true;
    System.out.println("b is " + b);

    // a boolean value can control the if statement
    if(b) System.out.println("This is executed.");

    b = false;
    if(b) System.out.println("This is not executed.");

    // outcome of a relational operator is a boolean value
    System.out.println("10 > 9 is " + (10 > 9));
  }
}
```

OUTPUT:

```
C:\ >javac Simple.java
C:\ >java Simple
b is false
b is false
This is expected
10>9 is true
```

EXAMPLE :: 11

```
// Demonstrate char data type.
class simple2 {
 public static void main(String args[]) {
  char ch1, ch2;

  ch1 = 88;  // code for X
  ch2 = 'Y';

  System.out.print("ch1 and ch2: ");
  System.out.println(ch1 + " " + ch2);
 }
}
```

OUTPUT:

```
C:\ >javac simple2.java
C:\ >java simple2
Ch1 and ch2:x y
```

EXAMPLE :: 12

```
// char variables behave like integers.
Class charDemo {
 public static void main(String args[]) {
  char ch1;

  ch1 = 'X';
  System.out.println("ch1 contains " + ch1);

  ch1++; // increment ch1
 .System.out.println("ch1 is now " + ch1);
 }.
}
```

OUTPUT:

```
C:\ >javac charDemo.java
C:\ >java charDemo
Ch1 contains x
Ch1 is now y
```

EXAMPLE :: 13

```
// Demonstrate casts.
class Conver {
  public static void main(String args[]) {
    byte b;
    int i = 257;
    double d = 323.142;

    System.out.println("\nConversion of int to byte.");
    b = (byte) i;
    System.out.println("i and b " + i + " " + b);

    System.out.println("\nConversion of double to int.");
    i = (int) d;
    System.out.println("d and i " + d + " " + i);

    System.out.println("\nConversion of double to byte.");
    b = (byte) d;
    System.out.println("d and b " + d + " " + b);
  }
}
```

OUTPUT:

```
C:\ >javac Conver.java
C:\ >java Conver
Conversion of int to byte
i and b 257 1
Conversion of double to int
d and i 323.142 323
Conversion of double to byte
d and b 323.142 67
```

EXAMPLE :: 14

```
// Demonstrate dynamic initialization.
class Dynamic {
  public static void main(String args[]) {
    double a = 3.0, b = 4.0;
    // c is dynamically initialized
    double c = Math.sqrt(a * a + b * b);
    System.out.println("Hypotenuse is " + c);
  }
}.
```

OUTPUT:

```
C:\ >javac Dynamic.java
C:\ >java Dynamic
Hypotenuse is 5.0
```

EXAMPLE :: 15

```
// Demonstrate lifetime of a variable.
class LifeTime {
 public static void main(String args[]) {
  int x;

   for(x = 0; x < 3; x++) {
    int y = -1; // y is initialized each time block is entered
    System.out.println("y iz: " + y); // this always prints -1
    y = 100;
    System.out.println("y is now: " + y);
   }
  }
}
```

OUTPUT:

```
C:\ >javac LifeTime.java
C:\ >java LifeTime
Y is:-1
Y is now:100
Y is:-1
Y is now:100
Y is:-1
Y is now:100
```

EXAMPLE :: 16

```
// Compute distance light travels using long variables.
class Light Demo
{
 public static void main(String args[]) {
  int lightspeed;
  long days;
  long seconds;
  long distance;
  // approximate speed of light in miles per second
  lightspeed = 186000;
  days = 1000; // specify number of days here
  seconds = days * 24 * 60 * 60; // convert to seconds
  distance = lightspeed * seconds; // compute distance
  System.out.print("In " + days);
  System.out.print(" days light will travel about ");
  System.out.println(distance + " miles.");
 }
}
```

OUTPUT:

```
C:\ >javac LightDemo.java
C:\ >java LightDemo
In 1000 days light will travel about 16070400000000 miles
```

EXAMPLE :: 17

```
// Initialize a two-dimensional array.
class Matrix {
 public static void main(String args[]) {
  double m[][] = {
   { 0*0, 1*0, 2*0, 3*0 },
   { 0*1, 1*1, 2*1, 3*1 },
   { 0*2, 1*2, 2*2, 3*2 },
   { 0*3, 1*3, 2*3, 3*3 }
  };
  int i, j;
  for(i=0; i<4; i++) {
   for(j=0; j<4; j++)
    System.out.print(m[i][j] + " ");
   System.out.println();
  }
 }
}
```

OUTPUT:

```
C:\ >javac Matrix.java
C:\ >java matrix
0.0 0.0 0.0 0.0
```

```
0.0 1.0 2.0 3.0
0.0 2.0 4.0 6.0
0.0 3.0 6.0 9.0
```

EXAMPLE :: 18

```
class Promote {
 public static void main(String args[]) {
  byte b = 42;
  char c = 'a';
  short s = 1024;
  int i = 50000;
  float f = 5.67f;
  double d = .1234;
  double result = (f * b) + (i / c) - (d * s);
  System.out.println((f * b) + " + " + (i / c) + " - " + (d * s));
  System.out.println("result = " + result);
 }
}
```

OUTPUT:

```
C:\ >javac Promote.java
C:\ >java Promote
238.14+515-126.3616
Result=626.7784146484375
```

EXAMPLE :: 19

```
// Demonstrate block scope.
class Scope {
 public static void main(String args[]) {
```

```
  int x; // known to all code within main
  x = 10;
  if(x == 10) { // start new scope
   int y = 20; // known only to this block
   // x and y both known here.
   System.out.println("x and y: " + x + " " + y);
   x = y * 2;
  }
  // y = 100; // Error! y not known here
  // x is still known here.
  System.out.println("x is " + x);
 }
}
```

OUTPUT:

```
C:\ >javac Scope.java
C:\ >java Scope
X and y:10 20
X is 40
```

EXAMPLE :: 20

```
// This program will not compile
class ScopeErr {
  public static void main(String args[]) {
    int bar = 1;
class threeDMatrix {
    {           // creates a new scope
     int bar = 2; // Compile time error -- bar already defined!
    }
  }
}
```

OUTPUT:

```
C:\ >javac ScopeErr.java
C:\ >java ScopeErr
Compile run time—bar already defined
```

EXAMPLE :: 21

```
// Demonstrate a three-dimensional array.
Class threeDarray
 public static void main(String args[]) {
  int threeD[][][] = new int[3][4][5];
  int i, j, k;

  for(i=0; i<3; i++)
   for(j=0; j<4; j++)
    for(k=0; k<5; k++)
    threeD[i][j][k] = i * j * k;
  for(i=0; i<3; i++) {
   for(j=0; j<4; j++) {
    for(k=0; k<5; k++)
     System.out.print(threeD[i][j][k] + " ");
    System.out.println();
   }
   System.out.println();
  }
 }
}
```

OUTPUT:

```
C:\ >javac threeDarrray.java
```

```
C:\ >java threeDarray
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0

0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 12

0 0 0 0 0
0 2 4 6 8
 0 4 8 12 16
0 6 12 18 24
```

EXAMPLE :: 22

```
// Manually allocate differing size second dimensions.
class TwoDAgain {
 public static void main(String args[]) {
  int twoD[][] = new int[4][];
  twoD[0] = new int[1];
  twoD[1] = new int[2];
  twoD[2] = new int[3];
  twoD[3] = new int[4];
  int i, j, k = 0;
  for(i=0; i<4; i++)
   for(j=0; j<i+1; j++) {
    twoD[i][j] = k;
    k++;
   }
```

```
    for(i=0; i<4; i++) {
     for(j=0; j<i+1; j++)
      System.out.print(twoD[i][j] + " ");
     System.out.println();
    }
  }
}
```

OUTPUT:

```
C:\ >javac TwoDAgain.java
C:\ >java TwoDAgain
0
1 2
3 4 5
6 7 8 9
```

EXAMPLE :: 23

```
// Demonstrate a two-dimensional array.
class TwoDArray {
 public static void main(String args[]) {
  int twoD[][]= new int[4][5];
  int i, j, k = 0;

  for(i=0; i<4; i++)
   for(j=0; j<5; j++) {
    twoD[i][j] = k;
    k++;
   }
```

```
  for(i=0; i<4; i++) {
   for(j=0; j<5; j++)
    System.out.print(twoD[i][j] + " ");
   System.out.println();
  }
 }
}
```

OUTPUT:

```
C:\ >javac TwoDarrray.java
C:\ >java TwoDarray
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
C:\vijay>
```

EXAMPLE :: 24

```
// Demonstrate the basic arithmetic operators.
class Basic {
 public static void main(String args[]) {
  // arithmetic using integers
  System.out.println("Integer Arithmetic");
  int a = 1 + 1;
  int b = a * 3;
  int c = b / 4;
  int d = c - a;
  int e = -d;
  System.out.println("a = " + a);
  System.out.println("b = " + b);
```

```
        System.out.println("c = " + c);
        System.out.println("d = " + d);
        System.out.println("e = " + e);
        // arithmetic using doubles
        System.out.println("\nFloating Point Arithmetic");
        double da = 1 + 1;
        double db = da * 3;
        double dc = db / 4;
        double dd = dc - a;
        double de = -dd;
        System.out.println("da = " + da);
        System.out.println("db = " + db);
        System.out.println("dc = " + dc);
        System.out.println("dd = " + dd);
        System.out.println("de = " + de);
    }
}
```

OUTPUT:

```
C:\ >javac Basic.java
C:\ >java Basic
Integer Arthimetic
a=2
b=6
c=1
d=-1
e=1
Floating point Arithmetic
da=2.0
db=6.0
dc=1.5
dd=-0.5
de=0.5
```

EXAMPLE :: 25

```
// Demonstrate the bitwise logical operators.
class BitLogic {
 public static void main(String args[]) {
  String binary[] = {
   "0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111",
   "1000", "1001", "1010", "1011", "1100", "1101", "1110", "1111"
  };
  int a = 3; // 0 + 2 + 1 or 0011 in binary
  int b = 6; // 4 + 2 + 0 or 0110 in binary
  int c = a | b;
  int d = a & b;
  int e = a ^ b;
  int f = (~a & b) | (a & ~b);
  int g = ~a & 0x0f;
  System.out.println("        a = " + binary[a]);
  System.out.println("        b = " + binary[b]);
  System.out.println("      a|b = " + binary[c]);
  System.out.println("      a&b = " + binary[d]);
  System.out.println("      a^b = " + binary[e]);
  System.out.println("~a&b|a&~b = " + binary[f]);
  System.out.println("       ~a = " + binary[g]);
 }
}
```

OUTPUT:

```
C:\ >javac BitLogic.java
```

```
C:\>java BitLogic
a=0011
b=0110
a|b=0111
a&b=0010
a^b=0101
~a&b|a&~b=0101
~a=1100
```

EXAMPLE :: 26

```
// Demonstrate the boolean logical operators.
Class vera
{
 public static void main(String args[]) {
  boolean a = true;
  boolean b = false;
  boolean c = a | b;
  boolean d = a & b;
  boolean e = a ^ b;
  boolean f = (!a & b) | (a & !b);
  boolean g = !a;
  System.out.println("       a = " + a);
  System.out.println("       b = " + b);
  System.out.println("     a|b = " + c);
  System.out.println("     a&b = " + d);
  System.out.println("     a^b = " + e);
  System.out.println("!a&b|a&!b = " + f);
  System.out.println("      !a = " + g);
 }
}
```

OUTPUT:

```
C:\ >javac vera.java
C:\ >java vera
a=true
b=false
a|b=true
a&b=false
a^b=true
!a&b|a&!b=true
!a=false
```

EXAMPLE :: 27

```
// Left shifting a byte value.
class Shift {
 public static void main(String args[]) {
  byte a = 64, b;
  int i;
  i = a << 2;
  b = (byte) (a << 2);
  System.out.println("Original value of a: " + a);
  System.out.println("i and b: " + i + " " + b);
 }
}
```

OUTPUT:

```
C:\ >javac Shift.java
C:\ >java Shift
Original value of a:64
i and b:256 0
```

EXAMPLE :: 28

```
// Unsigned shifting a byte value.
class ByteUShift {
 static public void main(String args[]) {
  char hex[] = {
   '0', '1', '2', '3', '4', '5', '6', '7',
   '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'
  };
  byte b = (byte) 0xf1;
  byte c = (byte) (b >> 4);
  byte d = (byte) (b >>> 4);
  byte e = (byte) ((b & 0xff) >> 4);
  System.out.println("          b = 0x"
   + hex[(b >> 4) & 0x0f] + hex[b & 0x0f]);
  System.out.println("      b >> 4 = 0x"
   + hex[(c >> 4) & 0x0f] + hex[c & 0x0f]);
  System.out.println("      b >>> 4 = 0x"
   + hex[(d >> 4) & 0x0f] + hex[d & 0x0f]);
  System.out.println("(b & 0xff) >> 4 = 0x"
   + hex[(e >> 4) & 0x0f] + hex[e & 0x0f]);
 }
}
```

OUTPUT:

```
C:\ >javac ByteUShift.java
C:\ >java ByteUshift
                    b=0*f1
                 b>>4=0*ff
               b>>>4=0*ff
            <b & 0*ff> >>4=0*0f
```

EXAMPLE :: 29

```
// Masking sign extension.
class HexByte {
 static public void main(String args[]) {
  char hex[] = {
   '0', '1', '2', '3', '4', '5', '6', '7',
   '8', '9', 'a', 'b', 'c', 'd', 'e', 'f'
  };
  byte b = (byte) 0xf1;
  System.out.println("b = 0x" + hex[(b >> 4) & 0x0f] + hex[b & 0x0f]);
 }}
```

OUTPUT:

```
C:\ >javac HexByte.java
C:\ >java HexByte
b=0*f1
```

EXAMPLE :: 30

```
// Demonstrate ++ and --.
class IncDec {
 public static void main(String args[]) {
  int a = 1;
  int b = 2;
  int c;
  int d;
  c = ++b;
  d = a++;
  c++;
  System.out.println("a = " + a);
```

```
        System.out.println("b = " + b);
        System.out.println("c = " + c);
        System.out.println("d = " + d);
    }
}
```

OUTPUT:

```
C:\ >javac IncDec.java
C:\ >java IncDec
a=2
b=3
c=4
d=1
```

EXAMPLE :: 31

```
// Demonstrate the % operator.
class Modulus {
 public static void main(String args[]) {
  int x = 42;
  double y = 42.3;
  System.out.println("x mod 10 = " + x % 10);
  System.out.println("y mod 10 = " + y % 10);
 }
}
```

OUTPUT

```
C:\ >javac Modulus.java
C:\ >java Modulus
X mod 10 = 2
Y mod 10 = 2.999999999999997
```

EXAMPLE :: 32

```
// Left shifting as a quick way to multiply by 2.
class Multi{
 public static void main(String args[]) {
  int i;
  int num = 0xFFFFFFE;

  for(i=0; i<4; i++) {
   num = num << 1;
   System.out.println(num);
  } } }
```

OUTPUT

```
C:\ >javac Multi.java
C:\ >java Multi
5367870908
1073741816
2147483632
-32
```

EXAMPLE :: 33

```
class Equals {
 public static void main(String args[]) {
  int a = 1;
  int b = 2;
  int c = 3;
  a |= 4;
  b >>= 1;
  c <<= 1;
```

```
        a ^= c;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
    }}
```

OUTPUT

```
C:\ >javac Equals.java
C:\ >java Equals
  a = 3
  b = 1
  c = 6
```

EXAMPLE :: 34

```
// Demonstrate several assignment operators.
class op{
 public static void main(String args[]) {
  int a = 1;
  int b = 2;
  int c = 3;
  a += 5;
  b *= 4;
  c += a * b;
  c %= 6;
  System.out.println("a = " + a);
  System.out.println("b = " + b);
  System.out.println("c = " + c);
 }
}
```

OUTPUT

```
C:\ >javac op.java
C:\ >java op
  a = 6
  b = 8
  c = 3
```

EXAMPLE :: 35

```
// Demonstrate ?.
class Ternary {
 public static void main(String args[]) {
  int i, k;
  i = 10;
  k = i < 0 ? -i : i; // get absolute value of i
  System.out.print("Absolute value of ");
  System.out.println(i + " is " + k);
  i = -10;
  k = i < 0 ? -i : i; // get absolute value of i
  System.out.print("Absolute value of ");
  System.out.println(i + " is " + k);
 }
}
```

OUTPUT

```
C:\ >javac Ternary.java
C:\ >java Ternary
Absolute value of 10 is 10
Absolute value of -10 is -10
```

EXAMPLE :: 36

```
// Using break as a civilized form of goto.
Class br{
 public static void main(String args[]) {
  boolean t = true;
  first: {
   second: {
    third: {
     System.out.println("Before the break.");
     if(t) break second; // break out of second block
     System.out.println("This won't execute");
    }
    System.out.println("This won't execute");
   }
   System.out.println("This is after second block.");
  }
 }
}
```

OUTPUT

```
C:\ >javac br.java
C:\ >java br
Before the break
This is after second block.
C:\ >
```

EXAMPLE :: 37

```
// This program contains an error.
class BreakErr {
 public static void main(String args[]) {
```

```
.one: for(int i=0; i<3; i++) {
  System.out.print("Pass " + i + ": ");
 }
 for(int j=0; j<100; j++) {
  if(j == 10) //break one; // WRONG
  System.out.print(j + " ");
 }
}
```

OUTPUT

```
C:\ >javac BreakErr.java
C:\ >java BreakErr
Pass  0: Pass 1: Pass 2: 10
C:\ >
```

EXAMPLE :: 38

```
// Using break to exit a loop.
class BreakLoop {
 public static void main(String args[]) {
  for(int i=0; i<100; i++) {
   if(i == 10) break; // terminate loop if i is 10
   System.out.println("i: " + i);
  }
  System.out.println("Loop complete.");
 }
}
```

OUTPUT

```
C:\ >javac BreakLoop.java
C:\ >java BreakLoop
```

```
i: 0
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
i: 9
Loop complete.
```

EXAMPLE :: 39

```
// Using break to exit a while loop.
class BreakLoop2 {
 public static void main(String args[]) {
  int i = 0;
   while(i < 100) {
    if(i == 10) break; // terminate loop if i is 10
    System.out.println("i: " + i);
    i++;
   }
  System.out.println("Loop complete.");
 }
}
```

OUTPUT

```
C:\ >javac BreakLoop2.java
C:\ >java BreakLoop2
```

```
i: 0
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
i: 9
Loop complete.
```

EXAMPLE :: 40

```
// Using break to exit from nested loops
class Loop4 {
 public static void main(String args[]) {
  outer: for(int i=0; i<3; i++) {
   System.out.print("Pass " + i + ": ");
   for(int j=0; j<100; j++) {
    if(j == 10) break outer; // exit both loops
    System.out.print(j + " ");
   }
   System.out.println("This will not print");
  }
 System.out.println("Loops complete.");
 }
}
```

OUTPUT

```
C:\ >javac Loop4.java
C:\ >java Loop4
Pass 0: 0 1 2 3 4 5 6 7 8 9  Loops complete
```

EXAMPLE :: 41

```
// Using the comma.
class pro{
 public static void main(String args[]) {
  int a, b;

  for(a=1, b=4; a<b; a++, b--) {
   System.out.println("a = " + a);
   System.out.println("b = " + b);
  }
 }
}
```

OUTPUT

```
C:\ >javac pro.java
C:\ >java pro
a = 1
b = 4
a = 2
b = 3
```

EXAMPLE :: 43

```
// Demonstrate continue.
class Continue {
 public static void main(String args[]) {
  for(int i=0; i<10; i++) {
   System.out.print(i + " ");
   if (i%2 == 0) continue;
   System.out.println("");
  }
 }
}
```

OUTPUT

```
C:\ >javac Continue.java
C:\ >java Continue
0 1
2 3
4 5
6 7
8 9
```

EXAMPLE :: 44

```
// Using continue with a label.
class ContinueLabel {
  public static void main(String args[]) {
outer: for (int i=0; i<10; i++) {
        for(int j=0; j<10; j++) {
          if(j > i) {
            System.out.println();
            continue outer;
          }
          System.out.print(" " + (i * j));
        }
      }
      System.out.println();
  }
}
```

OUTPUT

```
C:\ >javac ContinueLabel.java
C:\ >java ContinueLabel
```

```
0
0  1
0  2  4
0  3  6  9
0  4  8  12  16
0  5  10 15  20  25
0  6  12 18  24  30  36
0  7  14 21  28  35  42  49
0  8  16 24  32  40  48  56  64
0  9  18 27  36  45  54  63  72  81
```

EXAMPLE :: 45

```
// Demonstrate the do-while loop.
class DoWhile {
  public static void main(String args[]) {
    int n = 10;

    do {
      System.out.println("tick " + n);
      n--;
    } while(n > 0);
  }
}
```

OUTPUT

```
C:\ >javac DoWhile.java
C:\ >java  DoWhile
Tick 10
Tick  9
Tick  8
```

```
Tick 7
Tick 6
Tick 5
Tick 4
Tick 3
Tick 2
Tick 1
```

EXAMPLE :: 46

```
// Test for primes.
class Prime {
 public static void main(String args[]) {
  int num;
  boolean isPrime = true;
  num = 14;
  for(int i=2; i < num/2; i++) {
   if((num % i) == 0) {
    isPrime = false;
    break;
   }
  }
  if(isPrime) System.out.println("Prime");
  else System.out.println("Not Prime");
 }
}
```

OUTPUT

```
C:\ >javac Prime.java
C:\ >java  Prime
Not  Prime
```

EXAMPLE :: 48

```
// Declare a loop control variable inside the for.
class ForTick1 {
 public static void main(String args[]) {

  // here, n is declared inside of the for loop
  for(int n=10; n>0; n--)
   System.out.println("tick " + n);
 }
}
```

OUTPUT

```
C:\>javac ForTick1.java
C:\>java  ForTick1
Tick 10
Tick 9
Tick 8
Tick 7
Tick 6
Tick 5
Tick 4
Tick 3
Tick 2
Tick 1
```

EXAMPLE :: 49

```
// Parts of the for loop can be empty.
class ForVar {
 public static void main(String args[]) {
  int i;
```

```
  boolean done = false;
i = 0;
  for( ; !done; ) {
   System.out.println("i is " + i);
   if(i == 10) done = true;
   i++;
  }
 }
}
```

OUTPUT

```
C:\ >javac ForVar.java
C:\ >java  ForVar
i  is 0
i  is 1
i  is 2
i  is 3
i  is 4
i  is 5
i  is 6
i  is 7
i  is 8
i  is 9
i  is 10
```

EXAMPLE :: 50

```
// Demonstrate if-else-if statements.
class pro2{
 public static void main(String args[]) {
  int month = 4; // April
  String season;
```

```
    if(month == 12 || month == 1 || month == 2)
      season = "Winter";
    else if(month == 3 || month == 4 || month == 5)
      season = "Spring";
    else if(month == 6 || month == 7 || month == 8)
      season = "Summer";
    else if(month == 9 || month == 10 || month == 11)
      season = "Autumn";
    else
      season = "Bogus Month";

    System.out.println("April is in the " + season + ".");
  }
}
```

OUTPUT

```
C:\ >javac pro2.java
C:\ >java  pro2
April  is  in the Spring.
C:\ >
```

EXAMPLE :: 51

```
// Using a do-while to process a menu selection -- a simple help system.
class Menu {
 public static void main(String args[])
  throws java.io.IOException {
  char choice;
  do {
   System.out.println("Help on:");
   System.out.println("  1. if");
```

```
System.out.println("  2. switch");
System.out.println("  3. while");
System.out.println("  4. do-while");
System.out.println("  5. for\n");
System.out.println("Choose one:");
choice = (char) System.in.read();
} while( choice < '1' || choice > '5');
System.out.println("\n");
switch(choice) {

case '1':
  System.out.println("The if:\n");
  System.out.println("if(condition) statement;");
  System.out.println("else statement;");
  break;

case '2':
  System.out.println("The switch:\n");
  System.out.println("switch(expression) {");
  System.out.println("  case constant:");
  System.out.println("    statement sequence");
  System.out.println("  break;");
  System.out.println("  // ...");
  System.out.println("}");
  break;

case '3':
  System.out.println("The while:\n");
  System.out.println("while(condition) statement;");
  break;

case '4':
  System.out.println("The do-while:\n");
  System.out.println("do {");
```

```
        System.out.println("  statement;");
        System.out.println("} while (condition);");
        break;
      case '5':
        System.out.println("The for:\n");
        System.out.print("for(init; condition; iteration)");
        System.out.println(" statement;");
        break;
    }
  }
}
```

OUTPUT

```
C:\ >javac Menu.java
C:\ >java  Menu
Help on:
1. if
2. switch
3. while
4. do-while
5.  for

Choose one:  1
if<condition> statement;
else statement;
```

EXAMPLE :: 52

```
// In a switch, break statements are optional.
Class pro1{
 public static void main(String args[]) {
```

```
for(int i=0; i<12; i++)
 switch(i) {
  case 0:
  case 1:
  case 2:
  case 3:
  case 4:
   System.out.println("i is less than 5");
   break;
  case 5:
  case 6:
  case 7:
  case 8:
  case 9:
   System.out.println("i is less than 10");
   break;
  default:
   System.out.println("i is 10 or more.");
    }
 }
}
```

OUTPUT

```
C:\ >javac pro1.java
C:\ >java  pro1
i  is less than 5
i  is less than 5
i  is less than 5
i  is less than 5
i  is less than 5
i  is less than 10
i  is less than 10
```

```
i is less than 10
i is less than 10
i is less than 10
i is 10 or more
i is 10 or more
```

EXAMPLE :: 53

```
// Loops may be nested.
class Nested {
 public static void main(String args[]) {
  int i, k;

  for(i=0; i<10; i++) {
   for(k=i;k<10;k++)
    System.out.print(".");
   System.out.println();
  }
 }
}
```

OUTPUT

```
C:\ >javac Nested.java
C:\ >java  Nested
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . .
```

. . . .

. . .

. .

.

EXAMPLE :: 54

```
// The target of a loop can be empty.
class  target{
 public static void main(String args[]) {
   int i, j;
   i = 100;
   j = 200;
   // find midpoint between i and j
   while(++i < --j) ; // no body in this loop
   System.out.println("Midpoint is " + i);
 }
}
```

OUTPUT

```
C:\ >javac target.java
C:\ >java  target
Midpoint is 150
```

EXAMPLE :: 55

```
// Demonstrate return.
class Return {
  public static void main(String args[]) {
    boolean k= true;
```

```
    System.out.println("Before the return.");
    if(k) return // return to caller
    System.out.println("This won't execute.");
  }
}
```

OUTPUT

```
Error
```

EXAMPLE :: 56

```
class Sample3{
  public static void main(String args[]) {
  .int a, b;

    b = 4;
    for(a=1; a<b; a++) {
      System.out.println("a = " + a);
      System.out.println("b = " + b);
      b--;
    }
  }
}
```

OUTPUT

```
C:\ >javac Sample3.java
C:\ >java  Sample
a = 1
b = 4
a = 2
b = 3
```

EXAMPLE :: 57

```
// A simple example of the switch.
Class Switch3{
 public static void main(String args[]) {
  for(int i=0; i<6; i++)
   switch(i) {
    case 0:
     System.out.println("i is zero.");
     break;
    case 1:
     System.out.println("i is one.");
     break;
    case 2:
     System.out.println("i is two.");
     break;
    case 3:
     System.out.println("i is three.");
     break;
    default:
     System.out.println("i is greater than 3.");
   }
 }
}
```

OUTPUT

```
C:\ >javac Switch3.java
C:\ >java  Switch3
i is zero
i is one
i is two
i is three
i is greater than 3
i is greater than 3
```

EXAMPLE :: 58

```
// An improved version of the season program.
class Session{
   public static void main(String args[]) {
      int month = 4;
      String season;
      switch (month) {
       case 12:
       case 1:
       case 2:
        season = "Winter";
        break;
       case 3:
       case 4:
       case 5:
        season = "Spring";
        break;
       case 6:
       case 7:
       case 8:
        season = "Summer";
        break;
       case 9:
       case 10:
       case 11:
        season = "Autumn";
        break;
      default:
        season = "Bogus Month";
    }
    System.out.println("April is in the " + season + ".");
  }
}
```

OUTPUT:

```
C:\ >javac Session.java
C:\ >java Session
April is in the spring
```

EXAMPLE :: 59

```
// Demonstrate the while loop.
class Loop
 public static void main(String args[]) {
  int n = 10;

  while(n > 0) {
   System.out.println("tick " + n);
   n--;
  }
 }
}
```

OUTPUT:

```
C:\ >javac Loop.java
C:\ >java Loop
Tick 10
Tick 9
Tick 8
Tick 7
```

Tick 6

Tick 5

Tick 4

Tick 3

Tick 2

Tick 1