Rajalakshmi Engineering College

Rajalakshmi Nagar, Thandalam, Chennai - 602 105

Department of Computer Science and Engineering

Java

Sample Programs

Java 2 - The Complete Reference

Herbert Schildt

Compiled by:

B.BHUVANESWARAN

Assistant Professor (SS) / CSE / REC

bhuvaneswaran@rajalakshmi.edu.in

Program-01-Example.java

```
/*
This is a simple Java program.
Call this file "Example.java".
*/

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

Output:

```
This is a simple Java program.
```

Program-02-Example2.java

```
/*
Here is another short example.
Call this file "Example2.java".
*/

class Example2 {
     public static void main(String args[]) {
          int num; // this declares a variable called num

          num = 100; // this assigns num the value 100

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

          num = num * 2;

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

Output:

```
This is num: 100
The value of num * 2 is 200
```

Program-03-IfSample.java

```
/*
Demonstrate the if.
Call this file "IfSample.java".
*/

class IfSample {
     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:

```
x is less than y
x now equal to y
x now greater than y
```

Program-04-ForTest.java

```
/*
Demonstrate the for loop.
Call this file "ForTest.java".
*/

class ForTest {
     public static void main(String args[]) {
          int x;

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

Output:

```
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
```

Program-05-BlockTest.java

```
/*
Demonstrate a block of code.
Call this file "BlockTest.java"
*/

class BlockTest {
     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:

```
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
```

Program-01-Light.java

```
// Compute distance light travels using long variables.
class Light {
     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:

```
In 1000 days light will travel about 16070400000000 miles.
```

Program-02-Area.java

```
// Compute the area of a circle.
class Area {
     public static void main(String args[]) {
          double pi, r, a;

          r = 10.8; // radius of circle
          pi = 3.1416; // pi, approximately
          a = pi * r * r; // compute area

          System.out.println("Area of circle is " + a);
     }
}
```

Program-03-CharDemo.java

```
// Demonstrate char data type.
class CharDemo {
     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:

```
ch1 and ch2: X Y
```

Program-04-CharDemo2.java

```
// char variables behave like integers.
class CharDemo2 {
     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:

```
ch1 contains X
ch1 is now Y
```

Program-05-BoolTest.java

```
// Demonstrate boolean values.
class BoolTest {
     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:

```
b is false
b is true
This is executed.
10 > 9 is true
```

Program-06-DynInit.java

```
// Demonstrate dynamic initialization.
class DynInit {
    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);
    }
}
```

Program-07-Scope.java

```
// 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);
```

```
    }
}
```

Program-08-LifeTime.java

```
// 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 is: " + y); // this always prints -1
y = 100;
System.out.println("y is now: " + y);
}
}
}
```

Output:

```
y is: -1
y is now: 100
y is: -1
y is now: 100
y is: -1
y is now: 100
```

Program-09-ScopeErr.java

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

Program-10-Conversion.java

```
// Demonstrate casts.
class Conversion {
     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:

```
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
```

Program-11-Promote.java

```
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);
}
}
```

Program-12-Array.java

```
// 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.");
}
}
```

Program-13-AutoArray.java

```
// An improved version of the previous program.
class AutoArray {
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.");
}
}
```

Program-14-Average.java

```
// 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);
     }
}
```

Program-15-TwoDArray.java

```
// 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:

```
0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
```

Program-16-TwoDAgain.java

```
// 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:

```
0
1 2
3 4 5
6 7 8 9
```

Program-17-Matrix.java

```
// 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:

```
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
```

Program-18-ThreeDMatrix.java

```
// Demonstrate a three-dimensional array.
class threeDMatrix {
     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:

```
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
```

Program-01-BasicMath.java

```
// Demonstrate the basic arithmetic operators.
class BasicMath {
     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:

```
Integer Arithmetic
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
```

Program-02-Modulus.java

```
// Demonstrate the % operator.
class Modulus {
     public static void main(String args[]) {
          int x = 42;
          double y = 42.25;
          System.out.println("x mod 10 = " + x % 10);
          System.out.println("y mod 10 = " + y % 10);
     }
}
```

Output:

```
x mod 10 = 2
y mod 10 = 2.25
```

Program-03-OpEquals.java

```
// Demonstrate several assignment operators.
class OpEquals {
     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:

```
a = 6
b = 8
c = 3
```

Program-04-IncDec.java

```
// Demonstrate ++.
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:

```
a = 2
b = 3
c = 4
d = 1
```

Program-05-BitLogic.java

```
// 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:

```
        a = 0011
        b = 0110
      a|b = 0111
      a&b = 0010
      a^b = 0101
~a&b|a&~b = 0101
       ~a = 1100
```

Program-06-ByteShift.java

```
// Left shifting a byte value.
class ByteShift {
     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:

```
Original value of a: 64
i and b: 256 0
```

Program-07-MultByTwo.java

```
// Left shifting as a quick way to multiply by 2.
class MultByTwo {
     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:

```
536870908
1073741816
2147483632
-32
```

Program-08-HexByte.java

```
// 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:

```
b = 0xf1
```

Program-09-ByteUShift.java

```
// 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:

```
              b = 0xf1
         b >> 4 = 0xff
        b >>> 4 = 0xff
(b & 0xff) >> 4 = 0x0f
```

Program-10-OpBitEquals.java

```
class OpBitEquals {
     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:

```
a = 3
b = 1
c = 6
```

Program-11-BoolLogic.java

```
// Demonstrate the boolean logical operators.
class BoolLogic {
     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:

```
        a = true
        b = false
      a|b = true
      a&b = false
      a^b = true
a&b|a&!b = true
       !a = false
```

Program-12-Ternary.java

```
// 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:

```
Absolute value of 10 is 10
Absolute value of -10 is 10
```

Program-01-IfElse.java

```
// Demonstrate if-else-if statements.
class IfElse {
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:

```
April is in the Spring.
```

Program-02-SampleSwitch.java

```
// A simple example of the switch.
class SampleSwitch {
     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:

```
i is zero.
i is one.
i is two.
i is three.
i is greater than 3.
i is greater than 3.
```

Program-03-MissingBreak.java

```
// In a switch, break statements are optional.
class MissingBreak {
     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:

```
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
```

Program-04-Switch.java

```
// An improved version of the season program.
class Switch {
     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 + ".");
     }
}
```

Program-05-While.java

```
// Demonstrate the while loop.
class While {
     public static void main(String args[]) {
          int n = 10;
          while(n > 0) {
               System.out.println("tick " + n);
               n--;
          }
     }
}
```

Output:

```
tick 10
tick 9
tick 8
tick 7
tick 6
tick 5
tick 4
tick 3
tick 2
tick 1
```

Program-06-NoBody.java

```
// The target of a loop can be empty.
class NoBody {
     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:

```
Midpoint is 150
```

Program-07-DoWhile.java

```
// 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);
     }
}
```

Program-08-Menu.java

```
// Using a do-while to process a menu selection
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:

```
Help on:
1. if
2. switch
3. while
4. do-while
5. for
Choose one:
4
The do-while:
do {
statement;
} while (condition);
```

Program-09-ForTick.java

```
// Demonstrate the for loop.
class ForTick {
     public static void main(String args[]) {
          int n;

          for(n=10; n>0; n--)
               System.out.println("tick " + n);
     }
}
```

Program-10-ForTick.java

```
// Declare a loop control variable inside the for.
class ForTick {
     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);
     }
}
```

Program-11-FindPrime.java

```
// Test for primes.
class FindPrime {
     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");
        }
}
```

Program-12-Sample.java

```
class Sample {
    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--;
        }
    }
}
```

Program-14-ForVar.java

```
// 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++;
        }
    }
}
```

Program-15-Nested.java

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

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

Output:

```
..........
.........
........
.......
......
.....
....
...
..
.
```

Program-16-BreakLoop.java

```
// 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:

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

Program-17-BreakLoop2.java

```
// 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.");
     }
}
```

Program-18-BreakLoop3.java

```
// Using break with nested loops.
class BreakLoop3 {
     public static void main(String args[]) {
          for(int i=0; i<3; i++) {
          System.out.print("Pass " + i + ": ");
          for(int j=0; j<100; j++) {
          if(j == 10) break; // terminate loop if j is 10
               System.out.print(j + " ");
          }
          System.out.println();
          }
          System.out.println("Loops complete.");
     }
}
```

Output:

```
Pass 0: 0 1 2 3 4 5 6 7 8 9
Pass 1: 0 1 2 3 4 5 6 7 8 9
Pass 2: 0 1 2 3 4 5 6 7 8 9
Loops complete.
```

Program-19-Break.java

```
// Using break as a civilized form of goto.
class Break {
     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:

```
Before the break.
This is after second block.
```

Program-20-BreakLoop4.java

```
// Using break to exit from nested loops
class BreakLoop4 {
      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:

```
Pass 0: 0 1 2 3 4 5 6 7 8 9 Loops complete.
```

Program-21-BreakErr.java

```
// 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 + " ");
            }
      }
}
```

Program-22-Continue.java

```
// 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:

```
0 1
2 3
4 5
6 7
8 9
```

Program-23-ContinueLabel.java

```
// 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:

```
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
```

Program-24-Return.java

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

          System.out.println("Before the return.");

          if(t) return; // return to caller

          System.out.println("This won't execute.");
     }
}
```

Output:

```
Before the return.
```

Program-01-BoxDemo.java

```
/* A program that uses the Box class.
Call this file BoxDemo.java
*/
class Box {
     double width;
     double height;
     double depth;
}

// This class declares an object of type Box.
class BoxDemo {
     public static void main(String args[]) {
          Box mybox = new Box();
          double vol;

          // assign values to mybox's instance variables
          mybox.width = 10;
          mybox.height = 20;
          mybox.depth = 15;

          // compute volume of box
          vol = mybox.width * mybox.height * mybox.depth;

          System.out.println("Volume is " + vol);
     }
}
```

Output:

```
Volume is 3000.0
```

Program-02-BoxDemo2.java

```
// This program declares two Box objects.

class Box {
     double width;
     double height;
     double depth;
}

class BoxDemo2 {
     public static void main(String args[]) {
          Box mybox1 = new Box();
          Box mybox2 = new Box();
```

```
        double vol;

        // assign values to mybox1's instance variables
        mybox1.width = 10;
        mybox1.height = 20;
        mybox1.depth = 15;

        /* assign different values to mybox2's
        instance variables */
        mybox2.width = 3;
        mybox2.height = 6;
        mybox2.depth = 9;

        // compute volume of first box
        vol = mybox1.width * mybox1.height * mybox1.depth;
        System.out.println("Volume is " + vol);

        // compute volume of second box
        vol = mybox2.width * mybox2.height * mybox2.depth;
        System.out.println("Volume is " + vol);
    }
}
```

Output:

```
Volume is 3000.0
Volume is 162.0
```

Program-03-BoxDemo3.java

```
// This program includes a method inside the box class.

class Box {
    double width;
    double height;
    double depth;

    // display volume of a box
    void volume() {
        System.out.print("Volume is ");
        System.out.println(width * height * depth);
    }
}

class BoxDemo3 {
    public static void main(String args[]) {
        Box mybox1 = new Box();
        Box mybox2 = new Box();
```

```
            // assign values to mybox1's instance variables
            mybox1.width = 10;
            mybox1.height = 20;
            mybox1.depth = 15;

            /* assign different values to mybox2's
            instance variables */
            mybox2.width = 3;
            mybox2.height = 6;
            mybox2.depth = 9;

            // display volume of first box
            mybox1.volume();

            // display volume of second box
            mybox2.volume();
      }
}
```

Output:

```
Volume is 3000.0
Volume is 162.0
```

Program-04-BoxDemo4.java

```
// Now, volume() returns the volume of a box.

class Box {
      double width;
      double height;
      double depth;

      // compute and return volume
      double volume() {
            return width * height * depth;
      }
}

class BoxDemo4 {
      public static void main(String args[]) {
            Box mybox1 = new Box();
            Box mybox2 = new Box();
            double vol;

            // assign values to mybox1's instance variables
            mybox1.width = 10;
            mybox1.height = 20;
            mybox1.depth = 15;
```

```
            /* assign different values to mybox2's
            instance variables */
            mybox2.width = 3;
            mybox2.height = 6;
            mybox2.depth = 9;

            // get volume of first box
            vol = mybox1.volume();
            System.out.println("Volume is " + vol);

            // get volume of second box
            vol = mybox2.volume();
            System.out.println("Volume is " + vol);
      }
}
```

Program-05-BoxDemo5.java

```
// This program uses a parameterized method.

class Box {
      double width;
      double height;
      double depth;

      // compute and return volume
      double volume() {
            return width * height * depth;
      }

      // sets dimensions of box
      void setDim(double w, double h, double d) {
            width = w;
            height = h;
            depth = d;
      }
}

class BoxDemo5 {
      public static void main(String args[]) {
            Box mybox1 = new Box();
            Box mybox2 = new Box();
            double vol;

            // initialize each box
            mybox1.setDim(10, 20, 15);
            mybox2.setDim(3, 6, 9);

            // get volume of first box
            vol = mybox1.volume();
            System.out.println("Volume is " + vol);
```

```
            // get volume of second box
            vol = mybox2.volume();
            System.out.println("Volume is " + vol);
      }
}
```

Program-06-BoxDemo6.java

```
/* Here, Box uses a constructor to initialize the
dimensions of a box.
*/
class Box {
      double width;
      double height;
      double depth;

      // This is the constructor for Box.
      Box() {
            System.out.println("Constructing Box");
            width = 10;
            height = 10;
            depth = 10;
      }

      // compute and return volume
      double volume() {
            return width * height * depth;
      }
}

class BoxDemo6 {
      public static void main(String args[]) {
            // declare, allocate, and initialize Box objects
            Box mybox1 = new Box();
            Box mybox2 = new Box();

            double vol;

            // get volume of first box
            vol = mybox1.volume();
            System.out.println("Volume is " + vol);

            // get volume of second box
            vol = mybox2.volume();
            System.out.println("Volume is " + vol);
      }
}
```

Output:

```
Constructing Box
Constructing Box
Volume is 1000.0
Volume is 1000.0
```

Program-07-BoxDemo7.java

```
/* Here, Box uses a parameterized constructor to
initialize the dimensions of a box.
*/
class Box {
     double width;
     double height;
     double depth;

     // This is the constructor for Box.
     Box(double w, double h, double d) {
          width = w;
          height = h;
          depth = d;
     }

     // compute and return volume
     double volume() {
          return width * height * depth;
     }
}

class BoxDemo7 {
     public static void main(String args[]) {
          // declare, allocate, and initialize Box objects
          Box mybox1 = new Box(10, 20, 15);
          Box mybox2 = new Box(3, 6, 9);
          double vol;

          // get volume of first box
          vol = mybox1.volume();
          System.out.println("Volume is " + vol);

          // get volume of second box
          vol = mybox2.volume();
          System.out.println("Volume is " + vol);
     }
}
```

Output:

```
Volume is 3000.0
Volume is 162.0
```

Program-08-TestStack.java

```
// This class defines an integer stack that can hold 10 values.
class Stack {
     int stck[] = new int[10];
     int tos;

     // Initialize top-of-stack
     Stack() {
          tos = -1;
     }

     // Push an item onto the stack
     void push(int item) {
          if(tos==9)
          System.out.println("Stack is full.");
          else
          stck[++tos] = item;
     }

     // Pop an item from the stack
     int pop() {
          if(tos < 0) {
               System.out.println("Stack underflow.");
               return 0;
          }
          else
               return stck[tos--];
     }
}

class TestStack {
     public static void main(String args[]) {
          Stack mystack1 = new Stack();
          Stack mystack2 = new Stack();

          // push some numbers onto the stack
          for(int i=0; i<10; i++) mystack1.push(i);
          for(int i=10; i<20; i++) mystack2.push(i);

          // pop those numbers off the stack
          System.out.println("Stack in mystack1:");
          for(int i=0; i<10; i++)
               System.out.println(mystack1.pop());

          System.out.println("Stack in mystack2:");
          for(int i=0; i<10; i++)
               System.out.println(mystack2.pop());
     }
}
```

Output:

```
Stack in mystack1:
9
8
7
6
5
4
3
2
1
0
Stack in mystack2:
19
18
17
16
15
14
13
12
11
10
```

Program-01-Overload.java

```
// Demonstrate method overloading.
class OverloadDemo {
     void test() {
          System.out.println("No parameters");
     }

     // Overload test for one integer parameter.
     void test(int a) {
          System.out.println("a: " + a);
     }

     // Overload test for two integer parameters.
     void test(int a, int b) {
          System.out.println("a and b: " + a + " " + b);
     }

     // overload test for a double parameter
     double test(double a) {
          System.out.println("double a: " + a);
          return a*a;
     }
}

class Overload {
     public static void main(String args[]) {
          OverloadDemo ob = new OverloadDemo();
          double result;

          // call all versions of test()
          ob.test();
          ob.test(10);
          ob.test(10, 20);
          result = ob.test(123.25);
          System.out.println("Result of ob.test(123.25): " +
          result);
     }
}
```

Output:

```
No parameters
a: 10
a and b: 10 20
double a: 123.25
Result of ob.test(123.25): 15190.5625
```

Program-02-Overload.java

```
// Automatic type conversions apply to overloading.
class OverloadDemo {
     void test() {
          System.out.println("No parameters");
     }

     // Overload test for two integer parameters.
     void test(int a, int b) {
          System.out.println("a and b: " + a + " " + b);
     }

     // overload test for a double parameter
     void test(double a) {
          System.out.println("Inside test(double) a: " + a);
     }
}

class Overload {
     public static void main(String args[]) {
          OverloadDemo ob = new OverloadDemo();
          int i = 88;

          ob.test();
          ob.test(10, 20);

          ob.test(i); // this will invoke test(double)
          ob.test(123.2); // this will invoke test(double)
     }
}
```

Output:

```
No parameters
a and b: 10 20
Inside test(double) a: 88
Inside test(double) a: 123.2
```

Program-03-OverloadCons.java

```
/* Here, Box defines three constructors to initialize
the dimensions of a box various ways.
*/
class Box {
     double width;
     double height;
     double depth;
```

```
    // constructor used when all dimensions specified
    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    // constructor used when no dimensions specified
    Box() {
        width = -1; // use -1 to indicate
        height = -1; // an uninitialized
        depth = -1; // box
    }

    // constructor used when cube is created
    Box(double len) {
    width = height = depth = len;
    }

    // compute and return volume
    double volume() {
        return width * height * depth;
    }
}

class OverloadCons {
    public static void main(String args[]) {
        // create boxes using the various constructors
        Box mybox1 = new Box(10, 20, 15);
        Box mybox2 = new Box();
        Box mycube = new Box(7);

        double vol;

        // get volume of first box
        vol = mybox1.volume();
        System.out.println("Volume of mybox1 is " + vol);

        // get volume of second box
        vol = mybox2.volume();
        System.out.println("Volume of mybox2 is " + vol);

        // get volume of cube
        vol = mycube.volume();
        System.out.println("Volume of mycube is " + vol);
    }
}
```

Output:

```
Volume of mybox1 is 3000.0
Volume of mybox2 is -1.0
Volume of mycube is 343.0
```

Program-04-PassOb.java

```
// Objects may be passed to methods.
class Test {
     int a, b;

     Test(int i, int j) {
          a = i;
          b = j;
     }

     // return true if o is equal to the invoking object
     boolean equals(Test o) {
          if(o.a == a && o.b == b) return true;
          else return false;
     }
}

class PassOb {
     public static void main(String args[]) {
          Test ob1 = new Test(100, 22);
          Test ob2 = new Test(100, 22);
          Test ob3 = new Test(-1, -1);

          System.out.println("ob1 == ob2: " + ob1.equals(ob2));

          System.out.println("ob1 == ob3: " + ob1.equals(ob3));
     }
}
```

Output:

```
ob1 == ob2: true
ob1 == ob3: false
```

Program-05-OverloadCons2.java

```
// Here, Box allows one object to initialize another.

class Box {
     double width;
     double height;
     double depth;
```

```
    // construct clone of an object
    Box(Box ob) { // pass object to constructor
        width = ob.width;
        height = ob.height;
        depth = ob.depth;
    }

    // constructor used when all dimensions specified
    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    // constructor used when no dimensions specified
    Box() {
        width = -1; // use -1 to indicate
        height = -1; // an uninitialized
        depth = -1; // box
    }

    // constructor used when cube is created
    Box(double len) {
        width = height = depth = len;
    }

    // compute and return volume
    double volume() {
        return width * height * depth;
    }
}

class OverloadCons2 {
    public static void main(String args[]) {
        // create boxes using the various constructors
        Box mybox1 = new Box(10, 20, 15);
        Box mybox2 = new Box();
        Box mycube = new Box(7);

        Box myclone = new Box(mybox1);

        double vol;

        // get volume of first box
        vol = mybox1.volume();
        System.out.println("Volume of mybox1 is " + vol);

        // get volume of second box
        vol = mybox2.volume();
        System.out.println("Volume of mybox2 is " + vol);
```

```
            // get volume of cube
            vol = mycube.volume();
            System.out.println("Volume of cube is " + vol);

            // get volume of clone
            vol = myclone.volume();
            System.out.println("Volume of clone is " + vol);
      }
}
```

Program-06-CallByValue.java

```
// Simple types are passed by value.
class Test {
      void meth(int i, int j) {
            i *= 2;
            j /= 2;
      }
}

class CallByValue {
      public static void main(String args[]) {
            Test ob = new Test();

            int a = 15, b = 20;

            System.out.println("a and b before call: " +
            a + " " + b);

            ob.meth(a, b);

            System.out.println("a and b after call: " +
            a + " " + b);
      }
}
```

Output:

```
a and b before call: 15 20
a and b after call: 15 20
```

Program-07-CallByRef.java

```
// Objects are passed by reference.
class Test {
      int a, b;

      Test(int i, int j) {
            a = i;
            b = j;
      }
```

```
     // pass an object
     void meth(Test o) {
          o.a *= 2;

          o.b /= 2;
     }
}

class CallByRef {
     public static void main(String args[]) {
          Test ob = new Test(15, 20);

          System.out.println("ob.a and ob.b before call: " +
          ob.a + " " + ob.b);

          ob.meth(ob);

          System.out.println("ob.a and ob.b after call: " +
          ob.a + " " + ob.b);
     }
}
```

Output:

```
ob.a and ob.b before call: 15 20
ob.a and ob.b after call: 30 10
```

Program-08-RetOb.java

```
// Returning an object.
class Test {
     int a;

     Test(int i) {
          a = i;
     }

     Test incrByTen() {
          Test temp = new Test(a+10);
          return temp;
     }
}

class RetOb {
     public static void main(String args[]) {
          Test ob1 = new Test(2);
          Test ob2;

          ob2 = ob1.incrByTen();
          System.out.println("ob1.a: " + ob1.a);
          System.out.println("ob2.a: " + ob2.a);
```

```
            ob2 = ob2.incrByTen();
            System.out.println("ob2.a after second increase: "
            + ob2.a);
      }
}
```

Output:

```
ob1.a: 2
ob2.a: 12
ob2.a after second increase: 22
```

Program-09-Recursion.java

```
// A simple example of recursion.
class Factorial {
      // this is a recursive function
      int fact(int n) {
            int result;

            if(n==1) return 1;
            result = fact(n-1) * n;
            return result;
      }
}

class Recursion {
      public static void main(String args[]) {
            Factorial f = new Factorial();

            System.out.println("Factorial of 3 is " + f.fact(3));
            System.out.println("Factorial of 4 is " + f.fact(4));
            System.out.println("Factorial of 5 is " + f.fact(5));
      }
}
```

Output:

```
Factorial of 3 is 6
Factorial of 4 is 24
Factorial of 5 is 120
```

Program-10-Recursion2.java

```
// Another example that uses recursion.

class RecTest {
      int values[];
      RecTest(int i) {
            values = new int[i];
      }
```

```
     // display array -- recursively
     void printArray(int i) {
          if(i==0) return;
          else printArray(i-1);
          System.out.println("[" + (i-1) + "] " + values[i-1]);
     }
}

class Recursion2 {
     public static void main(String args[]) {
          RecTest ob = new RecTest(10);
          int i;

          for(i=0; i<10; i++) ob.values[i] = i;

          ob.printArray(10);
     }
}
```

Output:

```
[0] 0
[1] 1
[2] 2
[3] 3
[4] 4
[5] 5
[6] 6
[7] 7
[8] 8
[9] 9
```

Program-11-AccessTest.java

```
/* This program demonstrates the difference between
public and private.
*/
class Test {
     int a; // default access
     public int b; // public access
     private int c; // private access

     // methods to access c
     void setc(int i) { // set c's value
          c = i;
     }

     int getc() { // get c's value
          return c;
     }
}
```

```
class AccessTest {
     public static void main(String args[]) {
          Test ob = new Test();

          // These are OK, a and b may be accessed directly
          ob.a = 10;
          ob.b = 20;

          // This is not OK and will cause an error
          // ob.c = 100; // Error!

          // You must access c through its methods
          ob.setc(100); // OK

          System.out.println("a, b, and c: " + ob.a + " " +
          ob.b + " " + ob.getc());
     }
}
```

Program-12-TestStack.java

```
// This class defines an integer stack that can hold 10 values.
class Stack {
     /* Now, both stck and tos are private. This means
     that they cannot be accidentally or maliciously
     altered in a way that would be harmful to the stack.
     */
     private int stck[] = new int[10];
     private int tos;

     // Initialize top-of-stack
     Stack() {
          tos = -1;
     }

     // Push an item onto the stack
     void push(int item) {
          if(tos==9)
               System.out.println("Stack is full.");
          else
               stck[++tos] = item;
     }

     // Pop an item from the stack
     int pop() {
          if(tos < 0) {
               System.out.println("Stack underflow.");
               return 0;
          }
          else
               return stck[tos--];
```

```
    }
}

class TestStack {
    public static void main(String args[]) {
        Stack mystack1 = new Stack();
        Stack mystack2 = new Stack();

        // push some numbers onto the stack
        for(int i=0; i<10; i++) mystack1.push(i);
        for(int i=10; i<20; i++) mystack2.push(i);

        // pop those numbers off the stack
        System.out.println("Stack in mystack1:");
        for(int i=0; i<10; i++)
            System.out.println(mystack1.pop());

        System.out.println("Stack in mystack2:");
        for(int i=0; i<10; i++)
            System.out.println(mystack2.pop());

        // these statements are not legal
        // mystack1.tos = -2;
        // mystack2.stck[3] = 100;
    }
}
```

Program-13-UseStatic.java

```
// Demonstrate static variables, methods, and blocks.
class UseStatic {
    static int a = 3;
    static int b;

    static void meth(int x) {
        System.out.println("x = " + x);
        System.out.println("a = " + a);
        System.out.println("b = " + b);
    }

    static {
        System.out.println("Static block initialized.");
        b = a * 4;
    }

    public static void main(String args[]) {
        meth(42);
    }
}
```

Output:

```
Static block initialized.
x = 42
a = 3
b = 12
```

Program-14-StaticByName.java

```
class StaticDemo {
     static int a = 42;
     static int b = 99;
     static void callme() {
          System.out.println("a = " + a);
     }
}

class StaticByName {
     public static void main(String args[]) {
          StaticDemo.callme();
          System.out.println("b = " + StaticDemo.b);
     }
}
```

Output:

```
a = 42
b = 99
```

Program-15-Length.java

```
// This program demonstrates the length array member.
class Length {
     public static void main(String args[]) {
          int a1[] = new int[10];
          int a2[] = {3, 5, 7, 1, 8, 99, 44, -10};
          int a3[] = {4, 3, 2, 1};

          System.out.println("length of a1 is " + a1.length);
          System.out.println("length of a2 is " + a2.length);
          System.out.println("length of a3 is " + a3.length);
     }
}
```

Output:

```
length of a1 is 10
length of a2 is 8
length of a3 is 4
```

Program-16-TestStack2.java

```
// Improved Stack class that uses the length array member.
class Stack {
     private int stck[];
     private int tos;

     // allocate and initialize stack
     Stack(int size) {
          stck = new int[size];
          tos = -1;
     }

     // Push an item onto the stack
     void push(int item) {
          if(tos==stck.length-1) // use length member
               System.out.println("Stack is full.");
          else
               stck[++tos] = item;
     }

     // Pop an item from the stack
     int pop() {
          if(tos < 0) {
               System.out.println("Stack underflow.");
               return 0;
          }
          else
               return stck[tos--];
     }
}

class TestStack2 {
     public static void main(String args[]) {
          Stack mystack1 = new Stack(5);
          Stack mystack2 = new Stack(8);

          // push some numbers onto the stack
          for(int i=0; i<5; i++) mystack1.push(i);
          for(int i=0; i<8; i++) mystack2.push(i);

          // pop those numbers off the stack
          System.out.println("Stack in mystack1:");
          for(int i=0; i<5; i++)
               System.out.println(mystack1.pop());

          System.out.println("Stack in mystack2:");
          for(int i=0; i<8; i++)
               System.out.println(mystack2.pop());
     }
}
```

Program-17-InnerClassDemo.java

```
// Demonstrate an inner class.
class Outer {
     int outer_x = 100;

     void test() {
          Inner inner = new Inner();
          inner.display();
     }

     // this is an inner class
     class Inner {
          void display() {
               System.out.println("display: outer_x = " +
               outer_x);
          }
     }
}

class InnerClassDemo {
     public static void main(String args[]) {
          Outer outer = new Outer();
          outer.test();
     }
}
```

Output:

```
display: outer_x = 100
```

Program-18-InnerClassDemo.java

```
// This program will not compile.
class Outer {
     int outer_x = 100;

     void test() {
          Inner inner = new Inner();
          inner.display();
     }

     // this is an inner class
     class Inner {
          int y = 10; // y is local to Inner
          void display() {
               System.out.println("display: outer_x = " +
               outer_x);
          }
     }
```

```
     void showy() {
          System.out.println(y); // error, y not known here!
     }
}

class InnerClassDemo {
     public static void main(String args[]) {
          Outer outer = new Outer();
          outer.test();
     }
}
```

Program-19-InnerClassDemo.java

```
// Define an inner class within a for loop.
class Outer {
     int outer_x = 100;

     void test() {
          for(int i=0; i<10; i++) {
               class Inner {
                    void display() {
                         System.out.println("display: outer_x =
                         " + outer_x);
                    }
               }
               Inner inner = new Inner();
               inner.display();
          }
     }
}

class InnerClassDemo {
     public static void main(String args[]) {
          Outer outer = new Outer();
          outer.test();
     }
}
```

Output:

```
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
display: outer_x = 100
```

Program-20-StringDemo.java

```
// Demonstrating Strings.
class StringDemo {
     public static void main(String args[]) {
          String strOb1 = "First String";
          String strOb2 = "Second String";
          String strOb3 = strOb1 + " and " + strOb2;

          System.out.println(strOb1);
          System.out.println(strOb2);
          System.out.println(strOb3);
     }
}
```

Output:

```
First String
Second String
First String and Second String
```

Program-21-StringDemo2.java

```
// Demonstrating some String methods.
class StringDemo2 {
     public static void main(String args[]) {
          String strOb1 = "First String";
          String strOb2 = "Second String";
          String strOb3 = strOb1;
          System.out.println("Length of strOb1: " +
          strOb1.length());
          System.out.println("Char at index 3 in strOb1: " +
          strOb1.charAt(3));
          if(strOb1.equals(strOb2))
               System.out.println("strOb1 == strOb2");
          else
               System.out.println("strOb1 != strOb2");

          if(strOb1.equals(strOb3))
               System.out.println("strOb1 == strOb3");
          else
               System.out.println("strOb1 != strOb3");
     }
}
```

Output:

```
Length of strOb1: 12
Char at index 3 in strOb1: s
strOb1 != strOb2
strOb1 == strOb3
```

Program-22-StringDemo3.java

```
// Demonstrate String arrays.
class StringDemo3 {
     public static void main(String args[]) {
          String str[] = { "one", "two", "three" };

          for(int i=0; i<str.length; i++)
               System.out.println("str[" + i + "]: " + str[i]);
     }
}
```

Output:

```
str[0]: one
str[1]: two
str[2]: three
```

Program-23-CommandLine.java

```
// Display all command-line arguments.
class CommandLine {
public static void main(String args[]) {
     for(int i=0; i<args.length; i++)
          System.out.println("args[" + i + "]: " + args[i]);
}
}
```

Output:

```
java CommandLine this is a test 100 -1

args[0]: this
args[1]: is
args[2]: a
args[3]: test
args[4]: 100
args[5]: -1
```

Program-01-SimpleInheritance.java

```
// A simple example of inheritance.
// Create a superclass.
class A {
     int i, j;

     void showij() {
          System.out.println("i and j: " + i + " " + j);
     }
}

// Create a subclass by extending class A.
class B extends A {
     int k;

     void showk() {
          System.out.println("k: " + k);
     }

     void sum() {
          System.out.println("i+j+k: " + (i+j+k));
     }
}

class SimpleInheritance {
     public static void main(String args[]) {
          A superOb = new A();
          B subOb = new B();

          // The superclass may be used by itself.
          superOb.i = 10;
          superOb.j = 20;
          System.out.println("Contents of superOb: ");
          superOb.showij();
          System.out.println();

          /* The subclass has access to all public members of
          its superclass. */
          subOb.i = 7;
          subOb.j = 8;
          subOb.k = 9;
          System.out.println("Contents of subOb: ");
          subOb.showij();
          subOb.showk();
          System.out.println();
```

```
            System.out.println("Sum of i, j and k in subOb:");
            subOb.sum();
      }
}
```

Output:

```
Contents of superOb:
i and j: 10 20

Contents of subOb:
i and j: 7 8
k: 9

Sum of i, j and k in subOb:
i+j+k: 24
```

Program-02-Access.java

```
/* In a class hierarchy, private members remain
private to their class.

This program contains an error and will not
compile.
*/

// Create a superclass.
class A {
      int i; // public by default
      private int j; // private to A

      void setij(int x, int y) {
            i = x;
            j = y;
      }
}

// A's j is not accessible here.
class B extends A {
      int total;

      void sum() {
            total = i + j; // ERROR, j is not accessible here
      }
}

class Access {
      public static void main(String args[]) {
            B subOb = new B();

            subOb.setij(10, 12);
```

```
            subOb.sum();

            System.out.println("Total is " + subOb.total);
      }
}
```

Program-03-DemoBoxWeight.java

```
// This program uses inheritance to extend Box.
class Box {
      double width;
      double height;
      double depth;

      // construct clone of an object
      Box(Box ob) { // pass object to constructor
            width = ob.width;
            height = ob.height;
            depth = ob.depth;
      }

      // constructor used when all dimensions specified
      Box(double w, double h, double d) {
            width = w;
            height = h;
            depth = d;
      }

      // constructor used when no dimensions specified
      Box() {
            width = -1; // use -1 to indicate
            height = -1; // an uninitialized
            depth = -1; // box
      }

      // constructor used when cube is created
      Box(double len) {
            width = height = depth = len;
      }

      // compute and return volume
      double volume() {
            return width * height * depth;
      }
}

// Here, Box is extended to include weight.
class BoxWeight extends Box {
      double weight; // weight of box
```

```
     // constructor for BoxWeight
     BoxWeight(double w, double h, double d, double m) {
          width = w;
          height = h;
          depth = d;
          weight = m;
     }
}

class DemoBoxWeight {
     public static void main(String args[]) {
          BoxWeight mybox1 = new BoxWeight(10, 20, 15, 34.3);
          BoxWeight mybox2 = new BoxWeight(2, 3, 4, 0.076);
          double vol;

          vol = mybox1.volume();
          System.out.println("Volume of mybox1 is " + vol);
          System.out.println("Weight of mybox1 is " +
          mybox1.weight);
          System.out.println();

          vol = mybox2.volume();
          System.out.println("Volume of mybox2 is " + vol);
          System.out.println("Weight of mybox2 is " +
          mybox2.weight);
     }
}
```

Output:

```
Volume of mybox1 is 3000.0
Weight of mybox1 is 34.3

Volume of mybox2 is 24.0
Weight of mybox2 is 0.076
```

Program-04-RefDemo.java

```
class RefDemo {
public static void main(String args[]) {
     BoxWeight weightbox = new BoxWeight(3, 5, 7, 8.37);
     Box plainbox = new Box();
     double vol;

     vol = weightbox.volume();
     System.out.println("Volume of weightbox is " + vol);
     System.out.println("Weight of weightbox is " +
     weightbox.weight);
     System.out.println();
```

```
        // assign BoxWeight reference to Box reference
        plainbox = weightbox;
        vol = plainbox.volume(); // OK, volume() defined in Box
        System.out.println("Volume of plainbox is " + vol);

        /* The following statement is invalid because plainbox
        does not define a weight member. */
        // System.out.println("Weight of plainbox is " +
        plainbox.weight);
}
}
```

Program-05-DemoSuper.java

```
// A complete implementation of BoxWeight.
class Box {
        private double width;
        private double height;
        private double depth;

        // construct clone of an object
        Box(Box ob) { // pass object to constructor
                width = ob.width;
                height = ob.height;
                depth = ob.depth;
        }

        // constructor used when all dimensions specified
        Box(double w, double h, double d) {
                width = w;
                height = h;
                depth = d;
        }

        // constructor used when no dimensions specified
        Box() {
                width = -1; // use -1 to indicate
                height = -1; // an uninitialized
                depth = -1; // box
        }

        // constructor used when cube is created
        Box(double len) {
                width = height = depth = len;
        }

        // compute and return volume
        double volume() {
                return width * height * depth;
        }
}
```

```
// BoxWeight now fully implements all constructors.
class BoxWeight extends Box {
     double weight; // weight of box

     // construct clone of an object
     BoxWeight(BoxWeight ob) { // pass object to constructor
          super(ob);
          weight = ob.weight;
     }

     // constructor when all parameters are specified
     BoxWeight(double w, double h, double d, double m) {
          super(w, h, d); // call superclass constructor
          weight = m;
     }

     // default constructor
     BoxWeight() {
          super();
          weight = -1;
     }

     // constructor used when cube is created
     BoxWeight(double len, double m) {
          super(len);
          weight = m;
     }
}

class DemoSuper {
public static void main(String args[]) {
     BoxWeight mybox1 = new BoxWeight(10, 20, 15, 34.3);
     BoxWeight mybox2 = new BoxWeight(2, 3, 4, 0.076);
     BoxWeight mybox3 = new BoxWeight(); // default
     BoxWeight mycube = new BoxWeight(3, 2);
     BoxWeight myclone = new BoxWeight(mybox1);
     double vol;

     vol = mybox1.volume();
     System.out.println("Volume of mybox1 is " + vol);
     System.out.println("Weight of mybox1 is " +
     mybox1.weight);
     System.out.println();

     vol = mybox2.volume();
     System.out.println("Volume of mybox2 is " + vol);
     System.out.println("Weight of mybox2 is " +
     mybox2.weight);
     System.out.println();
```

```
        vol = mybox3.volume();
        System.out.println("Volume of mybox3 is " + vol);
        System.out.println("Weight of mybox3 is " +
        mybox3.weight);
        System.out.println();

        vol = myclone.volume();
        System.out.println("Volume of myclone is " + vol);
        System.out.println("Weight of myclone is " +
        myclone.weight);
        System.out.println();

        vol = mycube.volume();
        System.out.println("Volume of mycube is " + vol);
        System.out.println("Weight of mycube is " +
        mycube.weight);
        System.out.println();
        }
}
```

Output:

```
Volume of mybox1 is 3000.0
Weight of mybox1 is 34.3

Volume of mybox2 is 24.0
Weight of mybox2 is 0.076

Volume of mybox3 is -1.0
Weight of mybox3 is -1.0

Volume of myclone is 3000.0
Weight of myclone is 34.3

Volume of mycube is 27.0
Weight of mycube is 2.0
```

Program-06-UseSuper.java

```
// Using super to overcome name hiding.
class A {
     int i;
}

// Create a subclass by extending class A.
class B extends A {
     int i; // this i hides the i in A
     B(int a, int b) {
          super.i = a; // i in A
          i = b; // i in B
     }
```

```
     void show() {
          System.out.println("i in superclass: " + super.i);
          System.out.println("i in subclass: " + i);
     }
}

class UseSuper {
     public static void main(String args[]) {
          B subOb = new B(1, 2);
          subOb.show();
     }
}
```

Output:

```
i in superclass: 1
i in subclass: 2
```

Program-07-DemoShipment.java

```
// Extend BoxWeight to include shipping costs.

// Start with Box.
class Box {
     private double width;
     private double height;
     private double depth;

     // construct clone of an object
     Box(Box ob) { // pass object to constructor
          width = ob.width;
          height = ob.height;
          depth = ob.depth;
     }

     // constructor used when all dimensions specified
     Box(double w, double h, double d) {
          width = w;
          height = h;
          depth = d;
     }

     // constructor used when no dimensions specified
     Box() {
          width = -1; // use -1 to indicate
          height = -1; // an uninitialized
          depth = -1; // box
     }
```

```
        // constructor used when cube is created
        Box(double len) {
            width = height = depth = len;
        }

        // compute and return volume
        double volume() {
            return width * height * depth;
        }
}

// Add weight.
class BoxWeight extends Box {
        double weight; // weight of box

        // construct clone of an object
        BoxWeight(BoxWeight ob) { // pass object to constructor
            super(ob);
            weight = ob.weight;
        }

        // constructor when all parameters are specified
        BoxWeight(double w, double h, double d, double m) {
            super(w, h, d); // call superclass constructor
            weight = m;
        }

        // default constructor
        BoxWeight() {
            super();
            weight = -1;
        }

        // constructor used when cube is created
        BoxWeight(double len, double m) {
            super(len);
            weight = m;
        }
}

// Add shipping costs
class Shipment extends BoxWeight {
        double cost;

        // construct clone of an object
        Shipment(Shipment ob) { // pass object to constructor
            super(ob);
            cost = ob.cost;
        }
```

```
      // constructor when all parameters are specified
      Shipment(double w, double h, double d,
      double m, double c) {
            super(w, h, d, m); // call superclass constructor
            cost = c;
      }

      // default constructor
      Shipment() {
            super();
            cost = -1;
      }

      // constructor used when cube is created
      Shipment(double len, double m, double c) {
            super(len, m);
            cost = c;
      }
}

class DemoShipment {
public static void main(String args[]) {
      Shipment shipment1 =
      new Shipment(10, 20, 15, 10, 3.41);
      Shipment shipment2 =
      new Shipment(2, 3, 4, 0.76, 1.28);

      double vol;

      vol = shipment1.volume();
      System.out.println("Volume of shipment1 is " + vol);
      System.out.println("Weight of shipment1 is "
      + shipment1.weight);

      System.out.println("Shipping cost: $" + shipment1.cost);
      System.out.println();

      vol = shipment2.volume();
      System.out.println("Volume of shipment2 is " + vol);
      System.out.println("Weight of shipment2 is "
      + shipment2.weight);
      System.out.println("Shipping cost: $" + shipment2.cost);
}
}
```

Output:

```
Volume of shipment1 is 3000.0
Weight of shipment1 is 10.0
Shipping cost: $3.41
```

```
Volume of shipment2 is 24.0
Weight of shipment2 is 0.76
Shipping cost: $1.28
```

Program-08-CallingCons.java

```
// Demonstrate when constructors are called.

// Create a super class.
class A {
     A() {
          System.out.println("Inside A's constructor.");
     }
}

// Create a subclass by extending class A.
class B extends A {
     B() {
          System.out.println("Inside B's constructor.");
     }
}

// Create another subclass by extending B.
class C extends B {
     C() {
          System.out.println("Inside C's constructor.");
     }
}

class CallingCons {
     public static void main(String args[]) {
          C c = new C();
     }
}
```

Output:

```
Inside A's constructor
Inside B's constructor
Inside C's constructor
```

Program-09-Override.java

```
// Method overriding.
class A {
     int i, j;
     A(int a, int b) {
          i = a;
          j = b;
     }
```

```
     // display i and j
     void show() {
          System.out.println("i and j: " + i + " " + j);
     }
}

class B extends A {
     int k;

     B(int a, int b, int c) {
          super(a, b);
          k = c;
     }

     // display k - this overrides show() in A
     void show() {
          System.out.println("k: " + k);
     }
}

class Override {
     public static void main(String args[]) {
          B subOb = new B(1, 2, 3);

          subOb.show(); // this calls show() in B
     }
}
```

Output:

```
k: 3
```

Program-10-Override.java

```
// Methods with differing type signatures are overloaded - not
// overridden.
class A {
     int i, j;

     A(int a, int b) {
          i = a;
          j = b;
     }

     // display i and j
     void show() {
          System.out.println("i and j: " + i + " " + j);
     }
}
```

```
// Create a subclass by extending class A.
class B extends A {
     int k;

     B(int a, int b, int c) {
          super(a, b);
          k = c;
     }

     // overload show()
     void show(String msg) {
          System.out.println(msg + k);
     }
}

class Override {
     public static void main(String args[]) {
          B subOb = new B(1, 2, 3);
          subOb.show("This is k: "); // this calls show() in B
          subOb.show(); // this calls show() in A
     }
}
```

Output:

```
This is k: 3
i and j: 1 2
```

Program-11-Dispatch.java

```
// Dynamic Method Dispatch
class A {
     void callme() {
          System.out.println("Inside A's callme method");
     }
}

class B extends A {
     // override callme()
     void callme() {
          System.out.println("Inside B's callme method");
     }
}

class C extends A {
     // override callme()
     void callme() {
          System.out.println("Inside C's callme method");
     }
}
```

```
class Dispatch {
     public static void main(String args[]) {
          A a = new A(); // object of type A
          B b = new B(); // object of type B
          C c = new C(); // object of type C
          A r; // obtain a reference of type A

          r = a; // r refers to an A object
          r.callme(); // calls A's version of callme
          r = b; // r refers to a B object
          r.callme(); // calls B's version of callme
          r = c; // r refers to a C object
          r.callme(); // calls C's version of callme
     }
}
```

Output:

```
Inside A's callme method
Inside B's callme method
Inside C's callme method
```

Program-12-FindAreas.java

```
// Using run-time polymorphism.
class Figure {
     double dim1;
     double dim2;

     Figure(double a, double b) {
          dim1 = a;
          dim2 = b;
     }

     double area() {
          System.out.println("Area for Figure is undefined.");
          return 0;
     }
}

class Rectangle extends Figure {
     Rectangle(double a, double b) {
          super(a, b);
     }

     // override area for rectangle
     double area() {
          System.out.println("Inside Area for Rectangle.");
          return dim1 * dim2;
     }
}
```

```
class Triangle extends Figure {
     Triangle(double a, double b) {
          super(a, b);
     }

     // override area for right triangle
     double area() {
          System.out.println("Inside Area for Triangle.");
          return dim1 * dim2 / 2;
     }
}

class FindAreas {
     public static void main(String args[]) {
          Figure f = new Figure(10, 10);
          Rectangle r = new Rectangle(9, 5);
          Triangle t = new Triangle(10, 8);

          Figure figref;

          figref = r;
          System.out.println("Area is " + figref.area());

          figref = t;
          System.out.println("Area is " + figref.area());

          figref = f;
          System.out.println("Area is " + figref.area());
     }
}
```

Output:

```
Inside Area for Rectangle.
Area is 45
Inside Area for Triangle.
Area is 40
Area for Figure is undefined.
Area is 0
```

Program-13-AbstractDemo.java

```
// A Simple demonstration of abstract.
abstract class A {
     abstract void callme();

     // concrete methods are still allowed in abstract classes
     void callmetoo() {
          System.out.println("This is a concrete method.");
     }
}
```

```
class B extends A {
     void callme() {
          System.out.println("B's implementation of callme.");
     }
}

class AbstractDemo {
     public static void main(String args[]) {
          B b = new B();
          b.callme();
          b.callmetoo();
     }
}
```

Program-14-AbstractAreas.java

```
// Using abstract methods and classes.
abstract class Figure {
     double dim1;
     double dim2;

     Figure(double a, double b) {
          dim1 = a;
          dim2 = b;
     }

     // area is now an abstract method
     abstract double area();
}

class Rectangle extends Figure {
     Rectangle(double a, double b) {
          super(a, b);
     }

     // override area for rectangle
     double area() {
          System.out.println("Inside Area for Rectangle.");
          return dim1 * dim2;
     }
}

class Triangle extends Figure {
     Triangle(double a, double b) {
          super(a, b);
     }

     // override area for right triangle
     double area() {
          System.out.println("Inside Area for Triangle.");
          return dim1 * dim2 / 2;
```

```
    }
}

class AbstractAreas {
    public static void main(String args[]) {
        // Figure f = new Figure(10, 10); // illegal now
        Rectangle r = new Rectangle(9, 5);
        Triangle t = new Triangle(10, 8);

        Figure figref; // this is OK, no object is created

        figref = r;
        System.out.println("Area is " + figref.area());

        figref = t;
        System.out.println("Area is " + figref.area());
    }
}
```

RAJALAKSHMI
ENGINEERING COLLEGE

Website : www.rajalakshmi.org

Elearning : www.rajalakshmiengg.com

REC

Give a man a fish and you feed him for a day.

Teach him how to fish and you feed him for a lifetime.