**List of java Program**

**Program 11**

// 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));

}

}

This program generates the following output:

ob1 == ob2: true

ob1 == ob3: false

**Program 12**

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

class Box

{

double width;

double height;

double depth;

// Notice this constructor. It takes an object of type Box.

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); // create copy of 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 13**

// Primitive 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);

}

}

The output from this program is shown here:

a and b before call: 15 20

a and b after call: 15 20

**Program 14**

// 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);

}

}

This program generates the following output:

ob.a and ob.b before call: 15 20

ob.a and ob.b after call: 30 10

**Program 15**

// 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);

}

}

The output generated by this program is shown here:

ob1.a: 2

ob2.a: 12

ob2.a after second increase: 22

**Program 16**

// 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 from this application is shown here:

display: outer\_x = 100

**Program 17**

// 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();

}

}

The output from this version of the program is shown here.

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