OBJECT ORIENTED PROGRAMMING

Affefah Qureshi

Department of Computer Science

Iqra University, Islamabad Campus.

USING OBJECTS AS PARAMETERS

```
// 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
```

EXAMPLE

```
Output:

Volume of mybox1 is 3000.0
Volume of mybox2 is -1.0
Volume of cube is 343.0
Volume of clone is 3000.0
```

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

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

Box(double w, double h, double d) {
      width = w;
      height = h;
      depth = d;    }

Box() {
      width = -1;
      height = -1;
      depth = -1;    }

Box(double len) {
      width = height = depth = len;    }
```

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

class OverloadCons2 {
   public static void main(String args[]) {
Box mybox1 = new Box(10, 20, 15);
      Box mybox2 = new Box();
      Box mycube = new Box(7);
      Box myclone = new Box(mybox1);

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

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

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

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

ARGUMENT PASSING

- there are two ways that a computer language can pass an argument to a subroutine.
  - *call-by-value*
  - *call-by-reference*
- Java uses both approaches, depending upon what is passed.
- In Java, when you pass a primitive type to a method, it is passed by value.
- When you pass an object to a method, it is call-by-reference.
- *When a primitive type is passed to a method, it is done by use of call-by-value. Objects are implicitly passed by use of call-by-reference.*

EXAMPLE

Output:

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

```
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);    } }
```

RETURNING OBJECTS

```
Output:

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

A method can return any type of data, including class types.

```
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); } }
```