Object Passing

Unit-1

- Obj passing is done when we need to perform operations b/w 2 obj.

eg 1) Copy 1 obj to another.
2) Add 2 obj //(add var of 2 obj)
3) compare var of 2 obj.
etc

---

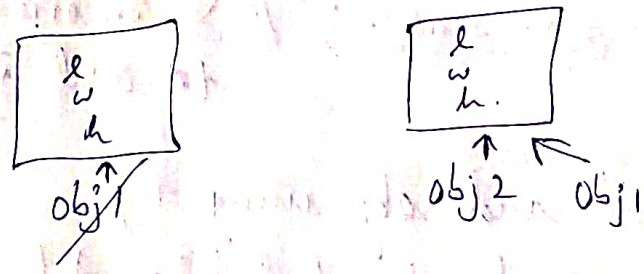
1) Suppose we wish to copy obj2 to obj1

If we copy like this

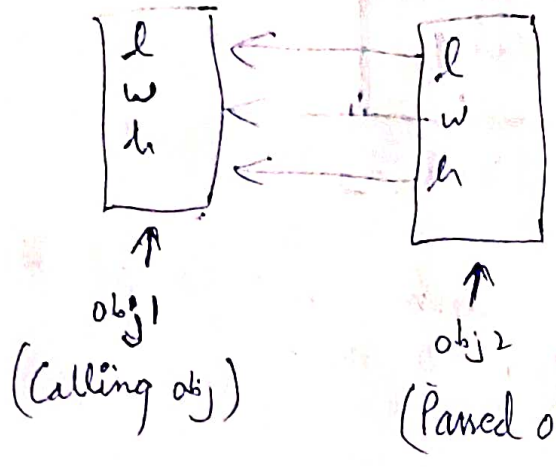
```
obj1 = obj2; //X
```

} Incorrect way to copy obj.

Then obj will not be copied



- obj → ref var. when = is used both point to same obj.
- How do we copy obj.
  - To copy obj, we need to copy var separately.



// Copy l, w, h of passed obj to l, w, h of calling obj

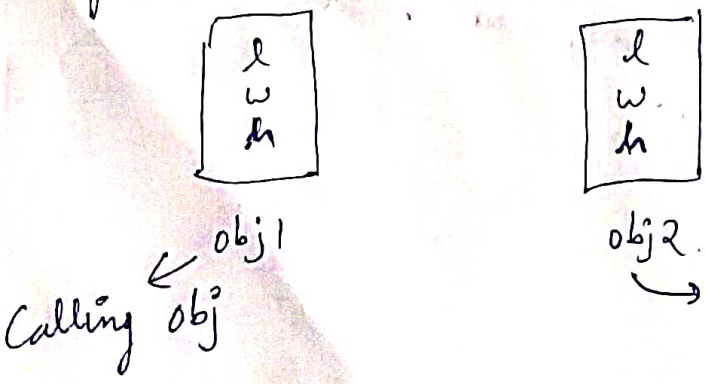
```
l = passedobj.l;
w = passedobj.w;
h = passedobj.h;
```
} Correct way to copy obj

---

2). Suppose we wish to add 2 obj; then this is not the correct way to add two objects

```
obj1 + obj2; // X Incorrect way to add 2 obj
```

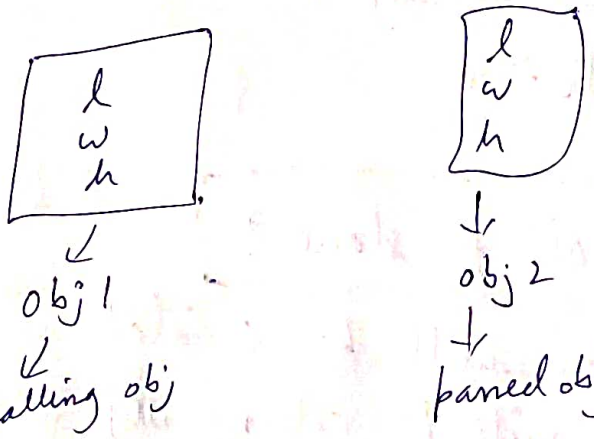
- We can't add obj using +, + only works for Primitive Data Types.
- To add obj, we need to add the var of 2 obj separately



```
l + parsedobj.l ;
w + parsedobj.w ;
h + parsedobj.h ;
```
} // Correct way to add 2 obj.

---

3) Suppose we need to compare the volume of 2 objects, then again we need to use obj passing.




```
if( l * w * h > parseobj.l * parsedobj.w
                * parsedobj.h )
{
    Sopln ("Calling obj's vol is larger");
}
else
{
    Sopln ("Parsed obj vol is higher");
}
```

Q) What is obj passing? Why is it done?

A) 1) obj passing means passing obj as param to functions.

2) obj p is done when we need to perform operations b/w obj eg- add 2 obj, compare 2 obj etc.

Q) How is obj p done?

A). A function will be created
. One obj will be calling obj & other obj will be passed obj.

. eg obj1.addobj(obj2);
   - obj1 → Calling obj
   - addobj → fun name
   - obj2 → passed.obj

. Var of calling obj r l, w, h
   " " passed obj r passedobj.l;
   passedobj.w;
   passedobj.h;

- A function will be defined inside class which will contain the operation b/w calling obj & passed obj.

- eg public void addobj(box passedobj){

```
Soln(l + passedobj.l);
Soln(w + passedobj.w);
Soln(h + passedobj.h);
}
```

---

1) WAP to print the sum of 2 Box objects using obj passing

```
class box{
    private int l, w, h;
    //set() → define set
    //get() →  "   get
    public void addobj(box passedobj){
        Soln(l + passedobj.l);
        Soln(w + passedobj.w);
        Soln(h + passedobj.h);
    }
}
//box ends.
```

//obj Passing fun defined here.

l → var of calling obj

passedobj.l → var of passed obj.

```
class demo {
    PSVM (---) {
        //create 2 obj
        box obj1 = new box();
        box obj2 = "    "   ;

        obj1.set(1,2,3);
        obj2.set(4,5,6);

        obj1.addObj(obj2); // fun w/ obj passing is called.
    }
}
```

Calling obj — Passed obj

---

2) WAP to copy 1 box ~~to~~ obj to another.

```
class box {
    //declare var - l, w, h.
    //set()  } define
    //get()
    public void copyObj(box passedObj) {
        l = passedObj.l;
        w = "    .w;
        h = "    .h;
    }
}
```

// fun w/ obj passing

Calling obj — Passed obj

```
class demo {
    psvm(---) {
        box obj1 = new box();
        obj1.set(1,2,3);
        box obj2 = new box();
        obj2.copyobj(obj1);
    }
}
```

Passed obj

Calling obj ← Passed obj copied to calling obj

---

3) WAP to compare volume of 2 box obj.

```
class box2
// define set, get & var.
public static void comparevol(box2 panedobj){
    if(l*w*h > panedobj.l * panedobj.w * panedobj.h)
    { Sopln("Calling obj is larger");
    }
    else
    { Sopln("paned obj is larger");
    }
}
}
```

```
c demo {
    PSUM(...) {
        //create 2 obj;
        // obj1.set(1,2,3);
          obj2.set(5,6,7);

        obj1.compareval(obj2);
    }
}
```

obj1 ↓ Calling obj

obj2 ↓ Passed obj..

---

4) Obj passing can also be done on a constructor.

- Create a copy constructor which copies 1 obj to another obj.
- In a copy constructor passed obj is copied to calling obj.

```
class box {
    // declare var, set(), get()
    box(box passedobj) {
    ↓
    Copy constr to copy passedobj to calling obj
    l = passedobj.l;
    w = passedobj.w;
    h = passedobj.h;
    }
}
```

c s{ psvm(...){

~~#the~~

box obj1 = new box(1,2,3);

box obj2 = new box(obj1);

↓

Const. is called &

obj1 is copied to obj2

↓ passed obj

↳ calling obj.