Constructor Overloading
==========

WAP to create a class called **Box** having 3 integer data members called l,b,h for storing length , breadth and height of the Box.

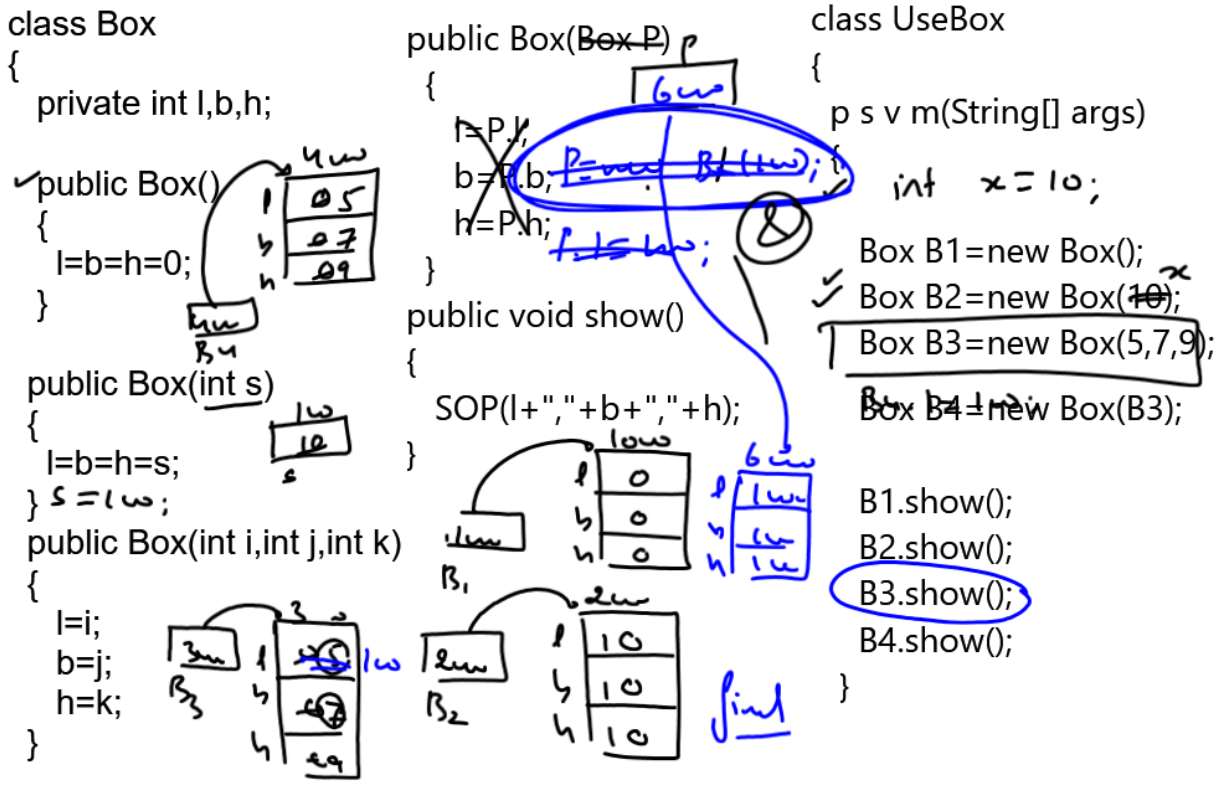
Also provide following CONSTRUCTORS/METHODS in your class:
1. A NON-PARAMETRIZED CONSTRUCTOR for initializing l,b,h to 0.

2. A SINGLE PARAMETRIZED CONSTRUCTOR for initializing l,b,h to the same value passed as argument , like a cube.

3. A TRIPLE PARAMETRIZED CONSTRUCTOR to initialize l,b,h to 3 diferent values passed as argument.

4. A CONSTRUCTOR which accepts a BOX object and copies it to another BOX object.

5. A method called show for displaying l,b,h

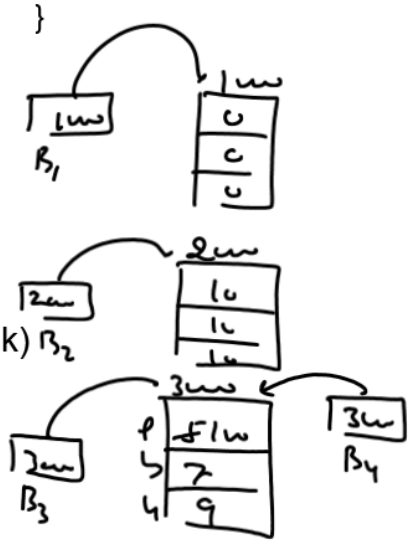


```
class Box
{
  private int l,b,h;

  public Box()
  {
    l=b=h=0;
  }

  public Box(int s)
  {
    l=b=h=s;
  }
  public Box(int i,int j,int k)
  {
     l=i;
     b=j;
     h=k;
  }
```

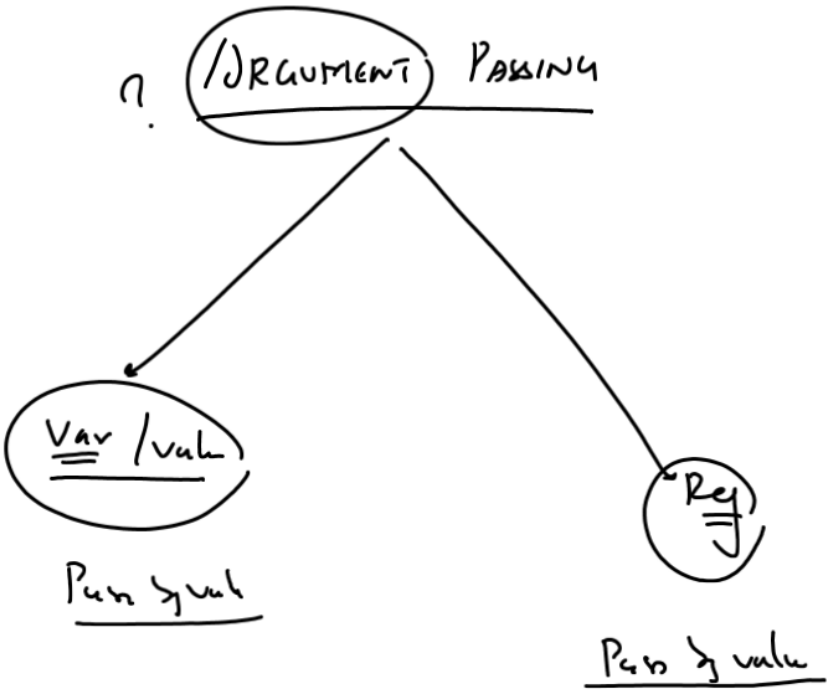
```
public void show()
{
   SOP(l+","+b+","+h);
}
}
```




```
class UseBox
{
  p s v m(String[] args)
  {

    Box B1=new Box();
    Box B2=new Box(10);
    Box B3=new Box(5,7,9);

    Box B4=B3;
    B4.l=1000;
    B1.show();
    B2.show();
    B3.show();
    B4.show();
  }
}
```



Argument Passing in Java
==========
In Java we can pass two types of arguments to a method and they are :

1. Variables (Primitive datatypes like int, char , float, etc.)

3. References (Array reference and object reference)

1. Passing Variables
==========
Whenever we pass any variable as argument to a method then Java passes its value. This value is then received by the formal argument declared in the method's argument list. So now, both the arguments, actual and formal, have the same value but their address are different. So if we make any change in the value of formal argument then it will never effect the value of actual argument. **Thus we say that in Java variables are always passed using pass by value.**

1. Passing References
==========
The 2nd type of argument which we can pass to any method in Java is reference (Array reference or object reference). Whenever we pass any reference as argument to a method then Java passes the address of the object or array pointed by that reference. This address is then received by formal reference declared in the argument list of the method. So now, both the references, actual and formal, point to the same object. Now if we will change the value of the object using formal reference then it wil surely change the value of original object and due to this many beginners in Java start thinking that it is pass by reference. But this is **not true.**

This is because if we will change the formal reference itself then it will neither effect original object nor the actual reference. Thus **even references in Java are passed using pass by value.**

So the final conclusion is that **Java doesn't support pass by reference and everything is passed using pass by value.**

Q. Why Java, being such a powerful langauge, doesn't suuport pass by reference?

==========

Ans: Java is a language which is primarly used for developing web applications and is very popular in banking and insurance sector. Now if the developers of Java supported pass by reference then it would have been a security threat for Java application running on servers because using pass by reference we can easily manipulate original value. So in the favor of security the developers of Java decided not to support pass by reference.

```
class Num {
   private int a,b;
   public void setNum(int i,int j){
      a=i;
      b=j;
   }
   public void showNum(){
      System.out.println("a="+a
+",b="+b);
   }
   public void increment(Num P){
      P.a++;
      P.b++;
   }
}
```

```
class UseNum {
   public static void main(String[] args) {
      Num N=new Num();
      N.setNum(10,20);
      System.out.println("Before incrementing:");
      N.showNum();
      Num Temp=new Num();
      Temp.increment(N);
      System.out.println("After incrementing:");
      N.showNum();
   }
}
```