

#Static

Any method or variable can be static.

↳ Accessible through same class or other classes without creating an object

↳ 'class name . member'

Static method/variable

class load \Rightarrow Memory

Allocate \Rightarrow

} works like a
global variable }

Static methods and variables can be shared just like global variables.

```
class A {
    static int a = 10;
    static int b = 20;
    int c = 30;
    static void show() {
        System.out.println("a: " + a + "\n" + b + " ");
    }
    static {
        System.out.println("Hello.");
    }
}
```

We can't directly access this variable since it is not static

On class loading,
load static values, execute static blocks.

```

public class Main {
    public static void main(String[] args) {
        A.show();
    }
}

```

to access the int variable "c" we need to create an object of A.

A ob = new A();

⇒ ~~ob~~ "ob.c"

< Static variables are assignable >

Can be changed at runtime

values

final int x = 20;

↑ cannot assign a new value.

↑ can be assigned only once (during initialization)

< Static has no this pointer >

< Static do not support "super" keyword >

< Static methods can be overloaded >

Inheritance

एक class को inherit कराया गया है, (जो)

- Super Class,
- Base class,
- Parent class.

यह class inherit कराता है,

- Sub class
- derived class
- Child class

<private attributes cannot be inherited>

<'default' acts like 'public'>

different package में under में

Inheritance using Methods

```
class A {  
    int i, j; { i("A")  
    void show1() {  
        System.out.println("i: " + i + "\n j: " + j);  
    }  
}
```

```
public class Main {  
    public static void main  
        (String[] args) {  
        B ob = new B();  
        B.set(10, 20, 30);  
        B.show();  
        B.show1();  
    }  
}
```

```
class B extends A {  
    int k;  
    void set(int i, int j, int k) {  
        this.i = i;  
        this.j = j;  
        this.k = k;  
    }  
    void show() {  
        System.out.println(i + j + k);  
    }  
}
```

Default constructor
calling sequence:

Super class

↳ Then subclass

A::

B::


```
class A {
```

```
    int i; j;
```

```
    A() { System.out.println("A"); }
```

```
}
```

```
class B extends A {
```

```
    B() { System.out.println("B"); }
```

```
}
```

⇒ Terminal:

A
B

inheritance

using constructors

```
class Area {
```

```
    int height, width;
```

```
    Area() { h=2, w=2; }
```

```
    Area(int height, int width) { this.height = height;  
                                  this.width = width; }
```

```
    Area(Area ob) { height = ob.height;  
                   width = ob.width; }
```

```
    void show() { System.out.println(height + width); }
```

```

class Rectangle extends Area {
    Rectangle() {
        super();
    }
    Rectangle(int height, int width) {
        super(height, width);
    }
    Rectangle(Rectangle Ob) {
        super(Ob);
    }
}

```

Question will be exceptional, max will be scenario-based questions

```

public class Main {

```

```

    psvm() {

```

```

        Rectangle Ob1 = new Rectangle();
        Ob1.show();

```

```

        Rectangle Ob2 = new Rectangle(4, 2);
        Ob2.show();

```

```

        Rectangle Ob3 = new Rectangle(Ob2);
        Ob3.show();

```

super.show()

↳ Area.show()

Superclass: Box → private height, width, depth

Subclass: Box1 → method volume()

└→ private weight

class Box {

private double height, width, depth;

Box () {

height = width = depth = 3;

}

Box(double height, double width, double depth) {

this.height = height;

this.width = width;

this.depth = depth;

}

Box(Box ob) {

height = ob.height;

width = ob.width;

depth = ob.depth;

}

void volume () {

System.out.println(

"Volume : " + (height * width * depth));

~~"in Depth : " +~~

}

```
class Box1 extends Box {
```

```
    private double weight;
```

```
    double width weight
```

```
    Box1() { Super(); } this.weight = weight; double weight
```

```
    Box1(double height, double width, double depth)
```

```
    { Super(height, width, depth); }
```

```
    this.weight = weight;
```

```
    Box1(Box ob) {
```

```
        Super(ob);
```

```
    }  
    void show() { System.out.println(weight); }
```

```
}
```

```
public class Main {
```

```
    public static void main(String[] args) {
```

```
        Box1 ob1 = new Box1(4);
```

```
        ob1.volume(); ob1
```

```
        Box1 ob2 = new Box1(5, 10, 15); 20
```

```
        ob2.volume();
```

```
        Box ob3 = new Box1(ob ob2);
```

```
        ob3.volume(); ob3.show();
```

```
    }
```

```
}
```