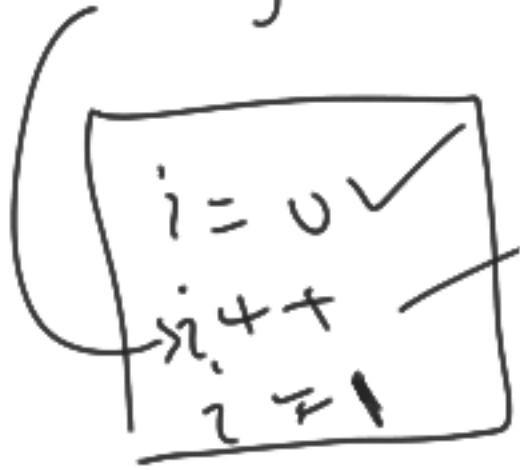


Emp 1 Obj



Emp 2



20 emp

... Obj 20

int count

count++
Emp 1

i = 0

count++
Emp 2

i = 0

Employee

Emp 3 →

i = 0

total no emp = 9 / Static
int
count; 0 ✓

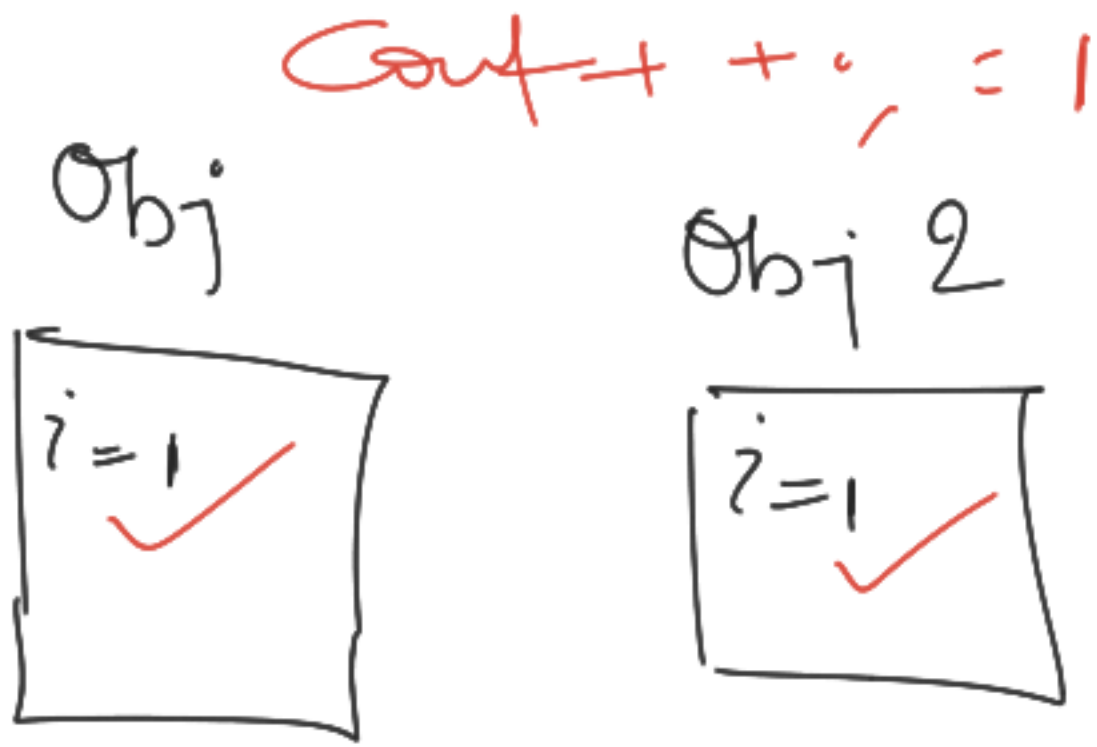
```

class S_NsVar{
    int i;
    static int count;
    S_NsVar(){
        i++;
    }
    public static void main(String arg[]){
        S_NsVar obj = new S_NsVar();
        System.out.println("i: "+(obj.i)); //0
        obj.i++;
        S_NsVar obj2 = new S_NsVar();

        System.out.println("i: "+(obj2.i)); //0
    }
}

```

Diagram illustrating the execution flow: Red arrows show the sequence of object creation and method calls. A blue arrow points to the `i++` in the constructor, and another blue arrow points to the `i++` in the `main` method. A red arrow points from the `main` method to the `main` method, indicating a recursive call or a loop.



Output

i = 1 ✓
i = 1

Count = 1
Count = 2
= 20

Static Variable

Class Variable
can be called with
Class Name.Var.

```

class Abc {
    void display () {
        S_NsVar.count++;
    }
}

```

```

class S_NsVar{
  int i;
  public static int count;
  S_NsVar(){
    i++;
    count++;
  }
  public static void main(String arg[]){
    S_NsVar obj = new S_NsVar();
    System.out.println("i: "+obj.i); //1

    System.out.println("Count: "+count);
    S_NsVar obj2 = new S_NsVar();

    System.out.println("i: "+(obj2.i)); //1

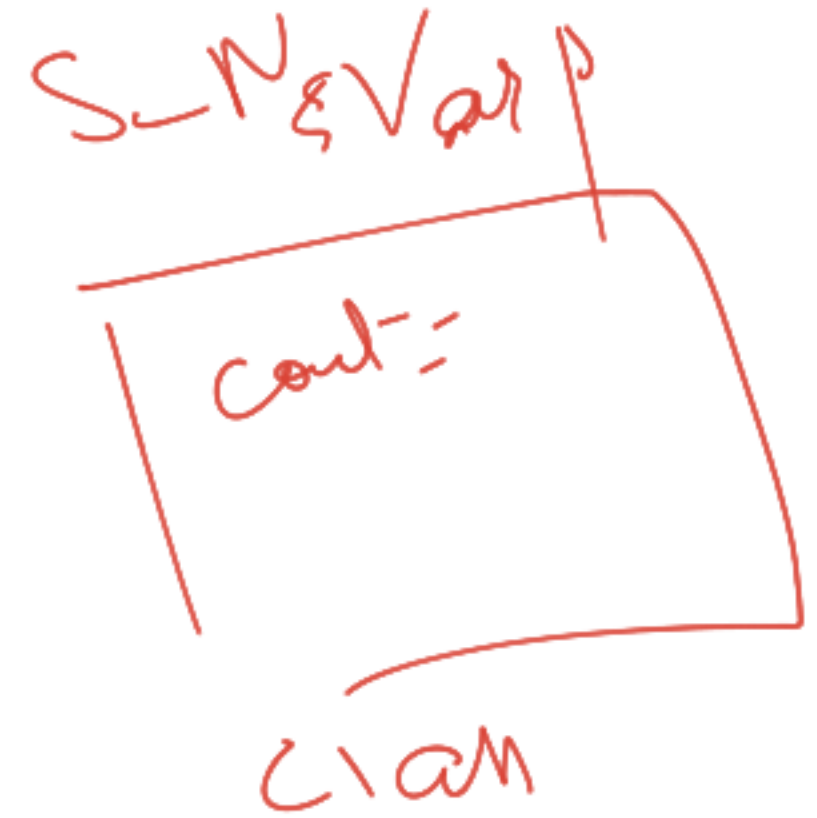
    System.out.println("Count: "+(S_NsVar.count));

    Abc abc = new Abc();
    abc.display();
  }
}

```



non-Static



Abc = display();

```

class Abc{
  static void display(){
    System.out.println("Count: "+(S_NsVar.count));
  }
}

```

S_NsVar obj = new S_NsVar();
 sout("Count-Abc: " + obj.count);

Abstract — method → method with no body

Abstract class → A class which contains
at least one abstract method

Abstract class — no obj can create

to use → inherit

abstract class Abc {

abstract void greet(); // Abstract method

void sayGM () {

System.out.println("Good Morning"); // Non-Abstract

}
class Def extends Abc {

void greet() {
}


```

abstract class Abs{
    abstract void greet();

    void sayGm(){
        System.out.println("Good Morning");
    }
}

class Def extends Abs{

    void sayGa(){
        System.out.println("Good afternoon");
    }
}

class Main extends Abs{
    void greet(){
        System.out.println("You're welcome");
    }
    public static void main(String[] args) {
        Main obj = new Main();
        obj.greet();
        obj.sayGm();
    }
}

```

Abc {
 . greet() → Abs
 sayGm() { }
 }
 ↓
 Def → Abc {
 ↑ greet() → Abs
 sayGm() { };
 }

main { ✓ - Abs
 greet() { }
 main() { }
 }

Interface ☀