Static Keyword :-

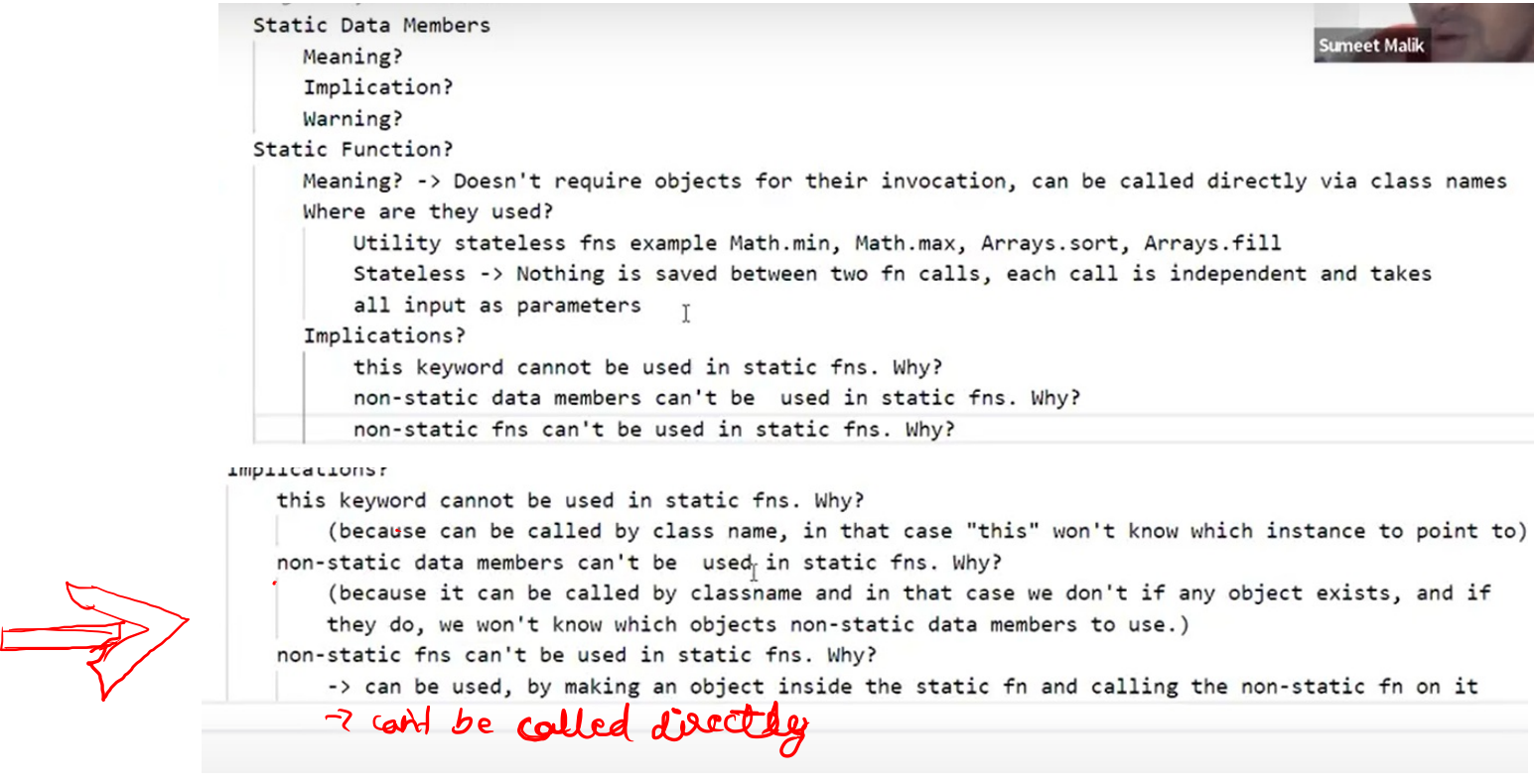
1) Static Data member
2) Static Function
3) Static class.

---

static Datamember :-

Common to all objects / shared by all objects.
Should be changed through class.
non-static members live in objects, static live in classes.
Every object has a connection to respective class to verify which datamembers to create in object allocated heapspace.

Static function:- A function which doesn't need object to run.




```
Static Data Members
    Meaning?
    Implication?
    Warning?
Static Function?
    Meaning? -> Doesn't require objects for their invocation, can be called directly via class names
    Where are they used?
        Utility stateless fns example Math.min, Math.max, Arrays.sort, Arrays.fill
        Stateless -> Nothing is saved between two fn calls, each call is independent and takes
        all input as parameters
    Implications?
        this keyword cannot be used in static fns. Why?
        non-static data members can't be  used in static fns. Why?
        non-static fns can't be used in static fns. Why?
```

```
Implications?
    this keyword cannot be used in static fns. Why?
        (because can be called by class name, in that case "this" won't know which instance to point to)
    non-static data members can't be  used in static fns. Why?
        (because it can be called by classname and in that case we don't if any object exists, and if
        they do, we won't know which objects non-static data members to use.)
    non-static fns can't be used in static fns. Why?
        -> can be used, by making an object inside the static fn and calling the non-static fn on it
```

→ can be called directly

```
Can non-static fn, use static data member? Why? -> Yes, because we have extra information available (not
less)
Can non-static fn, use static fn? Why? -> same as above.
```

Inheritance: depends upon Compiler & JRE

1. Compiler watches left hand side. (Reference)
2. Runtime watches right hand side.
3. If conflict at same level, then reference breaks the conflict.

```
Run | Debug
public static void main(String[] args) {
        C obj = new C();
        System.out.println(((P)obj).d);
        System.out.println(obj.d);
        System.out.println(obj.d1);
        System.out.println(obj.d2);

        obj.fun();
        obj.fun1();
        obj.fun2();
}
```

```
1
2
10
20
C fun
P fun1
C fun2
```

Case 1, 2

C obj = new C

P obj = new P

Obvious ones.

```
static class P{
        int d = 1;
        int d1 = 10;

        void fun() { System.out.println(x: "P fun"); }
        void fun1() { System.out.println(x: "P fun1"); }
}

static class C extends P{
        int d = 2;
        int d2 = 20;

        void fun() { System.out.println(x: "C fun"); }
        void fun2() { System.out.println(x: "C fun2"); }
}
```

```
public static void main(String[] args) throws NumberFormatException, IOException {
    // case 2 => C and C
    C obj = new C();
    System.out.println(obj.d); // conflict (object has P.d and C.d, resolved by ref -> C)
    System.out.println(((P)obj).d); // conflict (object has P.d and C.d, resolved by ref -> P)
    System.out.println(obj.d1); // available only in P
    System.out.println(obj.d2); // available only in C

    obj.fun(); // available at C first
    obj.fun1(); // available only in P
    obj.fun2(); // available only in C
}
```

P obj = new C

100 ↑ limit

1000 balance

Case 3

Data members → Reference decides
functions → Reference + Runtime decide.

Reference doesn't allow as d2 is present in the object created in C but cant be seen through Class/Reference of P.
Preference doesn't have the vision.

```
Run | Debug
21      public static void main(String[] args) {
22          P obj = new C();
23          // System.out.println(((C)obj).d2); not Allowed as d2 cant be see through P's reference
24          System.out.println(((C)obj).d); // Typecaseted to reference C we get C's d .
25          System.out.println(obj.d); // conflict gets resolved by reference P
26          System.out.println(obj.d1); //only P has d1
27          //System.out.println(obj.d2); not Allowed as d2 cant be see through P's reference
28
29          obj.fun(); //functios always go from bottom to top so we get C's fun
30          obj.fun1(); // only P has fun1
31          // obj.fun2();
32          ((P)obj).fun();
        }
```

⇒ C fun ⇒ method has overridden in runtime.

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL

PS D:\cpp\DSA\practice> cd "d:\cpp\DSA\practice\" ; if ($?) { javac inheritance.java } ; if ($?) { java inheritance
2
1
10
C fun
P fun1
```

Reference → decides allowed or not.
Object decides which function / data will run.
Here fun() has a conflict.

Reference allows fun as it is present in base class {P}

Runtime / Object / Right runs fun of class C {C fun} as it decides what to run.

Check NEXT PAGE

only functions get override

NOT

DATA MEMBERS.

Therefore while running function func() which is present in P and C in the case of P obj = new C() the func() returns "C fun" as C overrides P fun.

Therefore for conflicts in data members is resolved by Reference but conflicts in functions are resolved in runtime. → Method Overriding

Case 4

C obj = new P

not allowed

P {

}

P

↓

C

more things could get allowed through Reference C.

eg:- obj.d2

C extends P {

But object is of type P which doesn't contain any d2 variable in itself.

}