Static

	Java	C++	
→ Variable	✗	✓	// Defined inside main fn
→ Data Members	✓	✓	// Belongs to a class
→ Functions	✓	✓	

```
fun()
{ int i=10;
  static int j=20;
  i++; j++;
  cout<<i; cout<<j;
}
```

- fun() // 21
- fun() // 22
- fun() // 23
- fun() // 24

Note:- static persists throughout the program but still Scope is within the function defined (Block)

Global	Local	Static	C++
Scope: everywhere Lifecycle: Throughout the program	Scope: inside fn Lifecycle: throughout the function	Scope: Inside fn Lifecycle: Throughout the program	

A Typical Example:-

```
void fun1() {
  static int i=10;
  cout<<i;
}

void fun2() {
  static int i=20;
  cout<<i;
}

int main(int argc, char ** argv)
{ fun1();
  fun2();
}
```

- O/P:

```
10
20
```

Static stays in class & not in object

Static fn: These fns can be called directly on class name without creating objects;

Applications: [General Utility functions (stateless)]
eg Math.pow() they don't depend on any state

Q1) Can a static fn use this pointer inside it?
→ No

Q2) Can a " " use non-static data member?
→ No

Q3) Can a non-static fn use static data member?
→ Yes

Static fn's can use → static data members
→ static inner classes

eg Linked List

```
{
  static class Node {
  }
  psvm(--) {
  }
}
```

```
C
  int d1
  static int d2
  C IC1 { -- }
  SC IC2 { -- }
  static void fun() {
    d2 ✓   d1 ✗
    IC2 o2 = new IC2(); ✓
    IC1 o1 = new IC1(); ✗
  }
  C obj = new C();
  obj.d1 = 100; ✓
  obj.IC1 = obj.new IC1(); ✓
}
```

Note:- Static Inner class can be called without creating an object of outer class.

For Static :- oc.ic2 o1 = new oc.ic2();

For Non-Static :- oc.ic1 o1 = o.new ic1(); { oc o = new oc(); }