Final Keyword

*) Final is a keyword used in variable, method & class.

*) If we are using in variable then the variable becomes constant. We cannot change the value of the variable.

*) If we used in the method, the method cannot be override.

*) If we used in the class then the class will not be inherited

Final variable

```
class test
{
    psvm()
    {
        int i = 10;
        i = i + 20;
        sop(i);
    }
}
```

→

```
final int i = 10;
i = i + 20  ⇒ error.
sop(i);
```

*) we use final where we use any constant. force

e.g. $\pi = 3.141$, gravitational value = 9.8 meter/sec

Final method

```
class Demo
{
    void m1()
    {
        sop("I am in Demo");
    }
}
class test extends Demo
{
    void m1()
    {
        sop("I am in test");
    }
    psvm()
    {
    }
}
```

⇒ X

```
final class Demo
{
    final void m1()
    ;
}
```

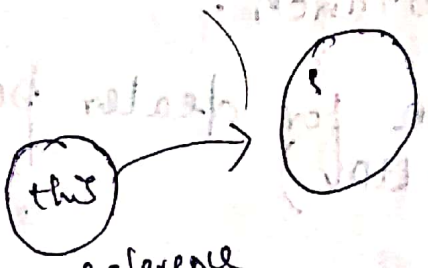
Final class

```
final class Demo
{
}
class test extends demo
{
    psvm()
    {
    }
}
```

X

This keyword.

*) this keyword is the Reference variable that refers to the current object.



*) class test
{ int i; → instance variable.
void input (int x) { → local variable
i° = x; i°; → local variable
this.i = i; → = instance variable.
Refer to current Reference variable or instance variable
}
void show()
{ sop(i);
}
}
class xyz
{ psvm( )
{ test t = new test();
t.input(10);
t.show();
} }

*) without this it will print 0.

*) This keyword can be used to refer current class instance variable.

*) This keyword can be used to invoke current class method (implicitly).

*) This keyword can be used to invoke current class constructor.

*) It can be used to pass as an argument in the method call.

*) used to pass as an argument in the constructor or call current class method

```
class test
{  int i;
   void display()
   { sop("Hello");
   }
   void show()
   { display();
   }
psvm()
{ test obj = new test();
  obj.show();
}
}
```

⇒ implicitly compiler will add this keyword to the call the instance method. i.e. this.display();

current class constructor.

```
class test
{
   test()
   { sop("no arguments");
     this(10);
   }
   test(int a)
   { sop("parameterized");
     this();
   }
psvm()
{
  test obj = new test();
}
```

call the default constructor.

argument in the method call

```
class test
{
  void m1(test ob)
  {
    S.o.p("I am in m1");
  }
  void m2()
  {
    m1(this);
  }
  p s v m ( )
  { test t = new test();
    t.m2();
  }
}
```

As an Argument.

used to pass as an argument in the constructor call.

```
class test
{
  test(test ob)
  {
  }
  :
}
  p s v m ( )
  {
  test t = new test(this);
```

User input through Scanner class :-

*) Scanner class is in java.util package.

```
import java.util.Scanner
Scanner s = new Scanner (System.in);
```

~~String name~~

→ s.o.p("Enter name");
String name = s.next();
↑ method in Scanner class.

→ s.o.p("Enter gender");
char g = s.next().charAt(0);

→ s.o.p("Enter age");
int age = s.nextInt();

→ s.o.p("Enter phone no");
long phoneno = s.nextLong();

Scanner class

Methods

nextBoolean ⟶ Used to read a Boolean value
nextDouble ⟶ " " " double
nextFloat ⟶ " " " float
nextInt ⟶ " " " Int
nextline ⟶ " " " a line

System.in

Standard, universally accessible input stream that developer used to read i/p from the terminal window. However system.in in java doesnot work alone To read the i/p with system.in, you must pass it as an argument to either the constructer of the Scanner class.