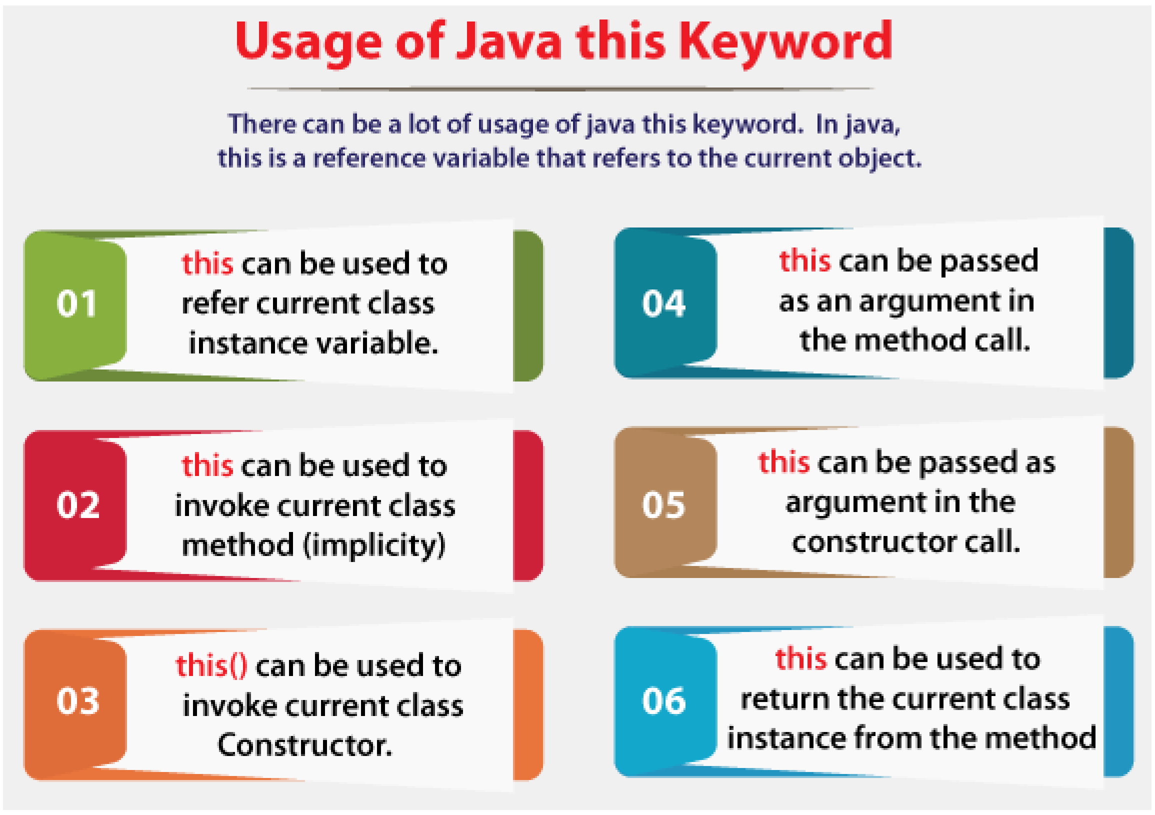
# this keyword

## We will cover the following

- this keyword
- refer to current class instance variable
- this: to invoke the current class method
- this: to invoke the current class constructor
- this: to pass as an argument in the method
- this keyword can be used to return the current class instance

## this keyword

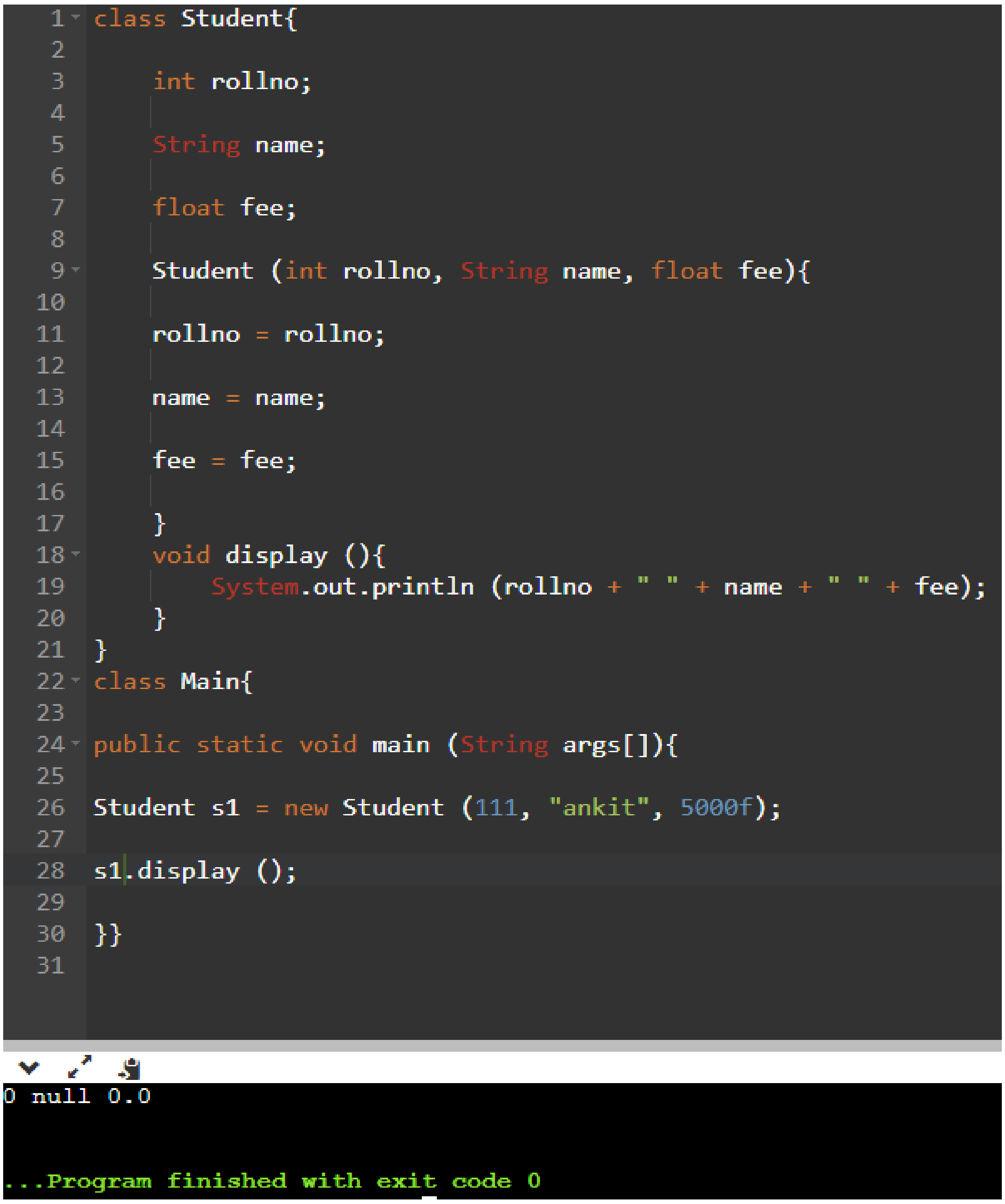
There can be a lot of usage of **Java this keyword**. In Java, this is a **reference variable** that refers to the current object.


Usage of Java this Keyword
There can be a lot of usage of java this keyword. In java, this is a reference variable that refers to the current object.
01 this can be used to refer current class instance variable.
02 this can be used to invoke current class method (implicity)
03 this() can be used to invoke current class Constructor.
04 this can be passed as an argument in the method call.
05 this can be passed as argument in the constructor call.
06 this can be used to return the current class instance from the method

## this: refer to a current class instance variable

there is ambiguity between the instance variables and parameters, this keyword resolves the problem of ambiguity.

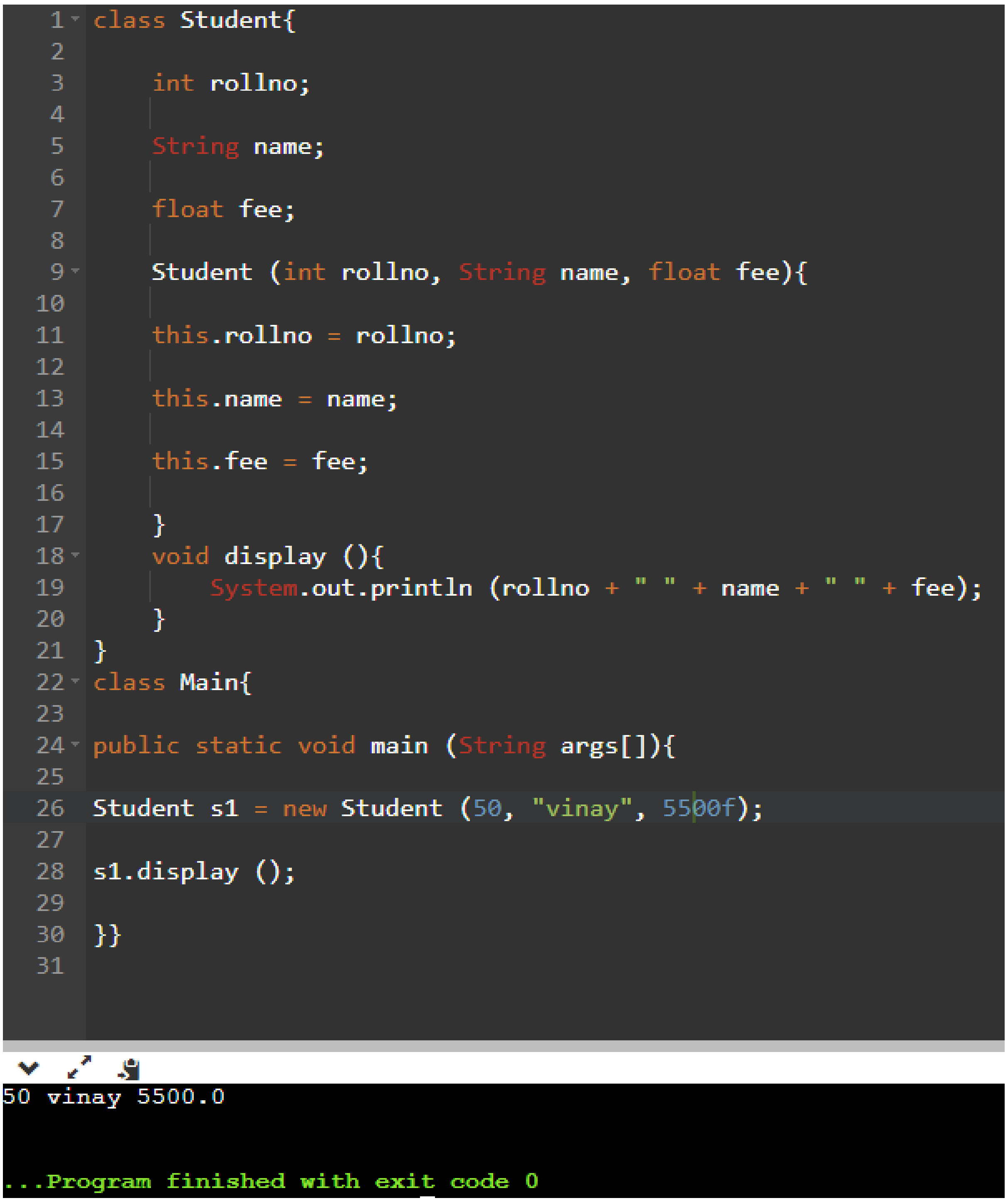
Let's take the case when we do not use this keyword then what happens actually



```
class Student{

    int rollno;

    String name;

    float fee;

    Student (int rollno, String name, float fee){

    rollno = rollno;

    name = name;

    fee = fee;

    }
    void display (){
        System.out.println (rollno + " " + name + " " + fee);
    }
}
class Main{

public static void main (String args[]){

Student s1 = new Student (111, "ankit", 5000f);

s1.display ();

}}
```

```
0 null 0.0


...Program finished with exit code 0
```

now you can notice that the values of roll no, name and fee do not change even we have assigned the values in the constructor. this is because the compiler is not able to distinguish local variables and instance variables because of the same names. this issue gets resolved using this keyword.

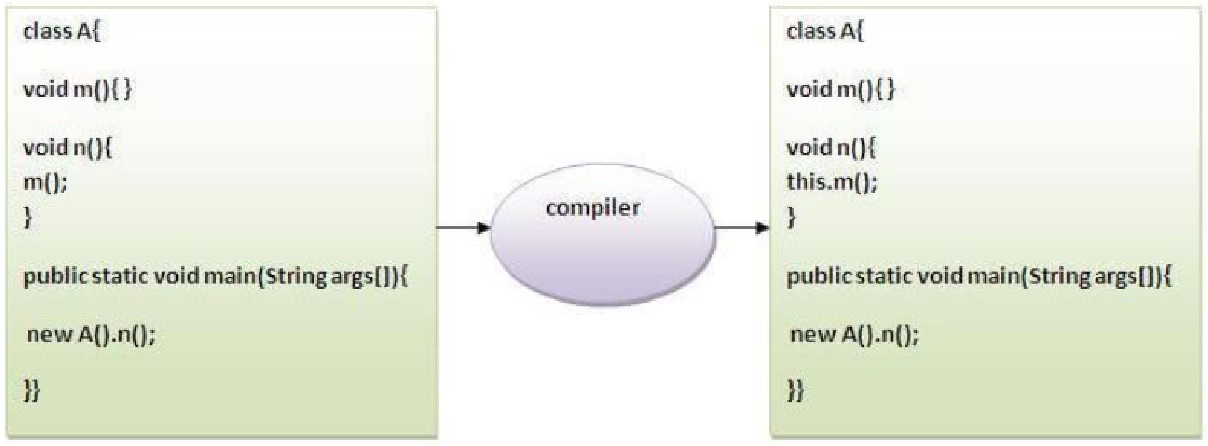


```java
class Student{

    int rollno;

    String name;

    float fee;

    Student (int rollno, String name, float fee){

    this.rollno = rollno;

    this.name = name;

    this.fee = fee;

    }
    void display (){
        System.out.println (rollno + " " + name + " " + fee);
    }
}
class Main{

public static void main (String args[]){

Student s1 = new Student (50, "vinay", 5500f);

s1.display ();

}}
```

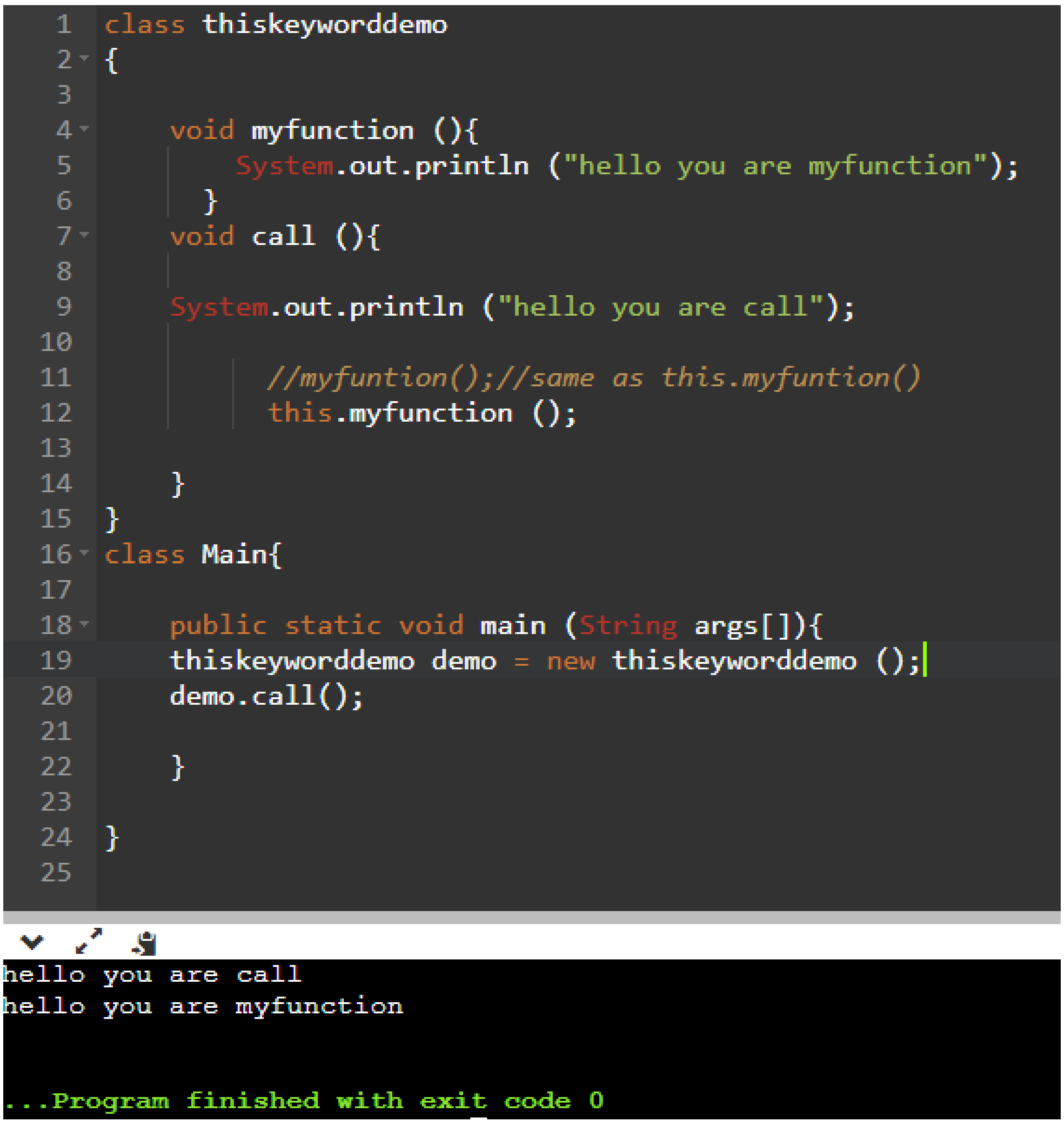
```
50 vinay 5500.0


...Program finished with exit code 0
```

## this: to invoke the current class method

this can be used to invoke the current class method.  But If you don't use this keyword, the compiler automatically adds this keyword while invoking the method. Let's see the example

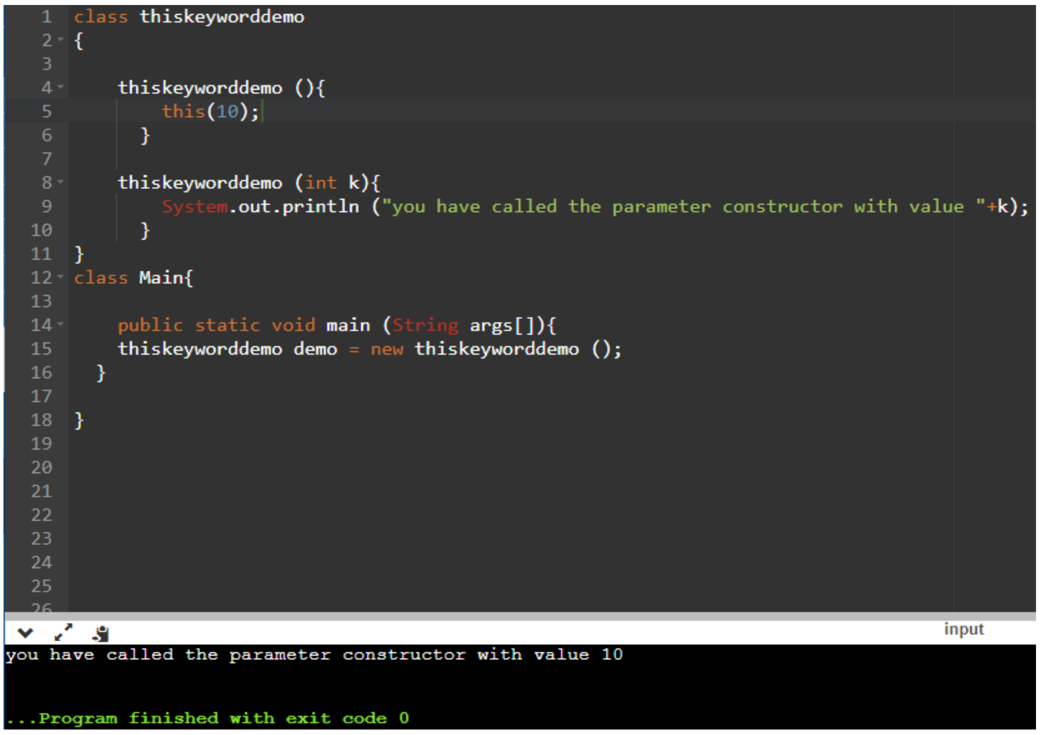

```
class A{

void m(){}

void n(){
m();
}

public static void main(String args[]){

 new A().n();

}}
```

```
class A{

void m(){}

void n(){
this.m();
}

public static void main(String args[]){

 new A().n();

}}
```


```
class thiskeyworddemo
{

    void myfunction (){
        System.out.println ("hello you are myfunction");
     }
    void call (){

    System.out.println ("hello you are call");

          //myfuntion();//same as this.myfuntion()
          this.myfunction ();

    }
}
class Main{

    public static void main (String args[]){
    thiskeyworddemo demo = new thiskeyworddemo ();
    demo.call();

    }

}

```
```
hello you are call
hello you are myfunction


...Program finished with exit code 0
```

## this: to invoke the current class constructor

The this() constructor call can be used to invoke the current class constructor. It is used to reuse the constructor. In other words, it is used for constructor chaining.

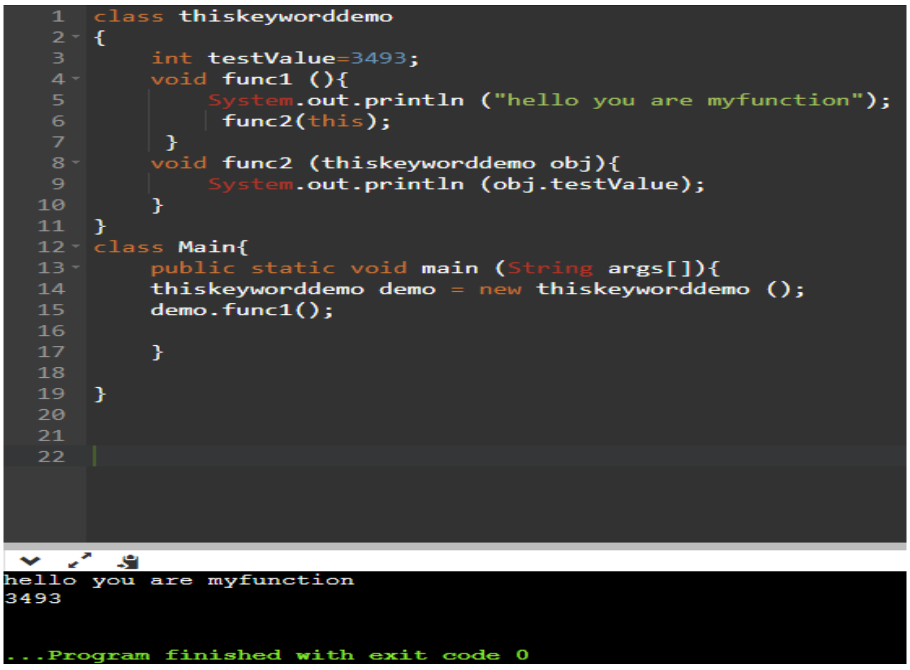


```
class thiskeyworddemo
{

    thiskeyworddemo (){
        this(10);
    }

    thiskeyworddemo (int k){
        System.out.println ("you have called the parameter constructor with value "+k);
    }
}
class Main{

    public static void main (String args[]){
    thiskeyworddemo demo = new thiskeyworddemo ();
  }

}
```

```
you have called the parameter constructor with value 10


...Program finished with exit code 0
```

## this: to pass as an argument in the method

The this keyword can also be passed as an argument in the method. It is mainly used in the event handling. Let's see the example:

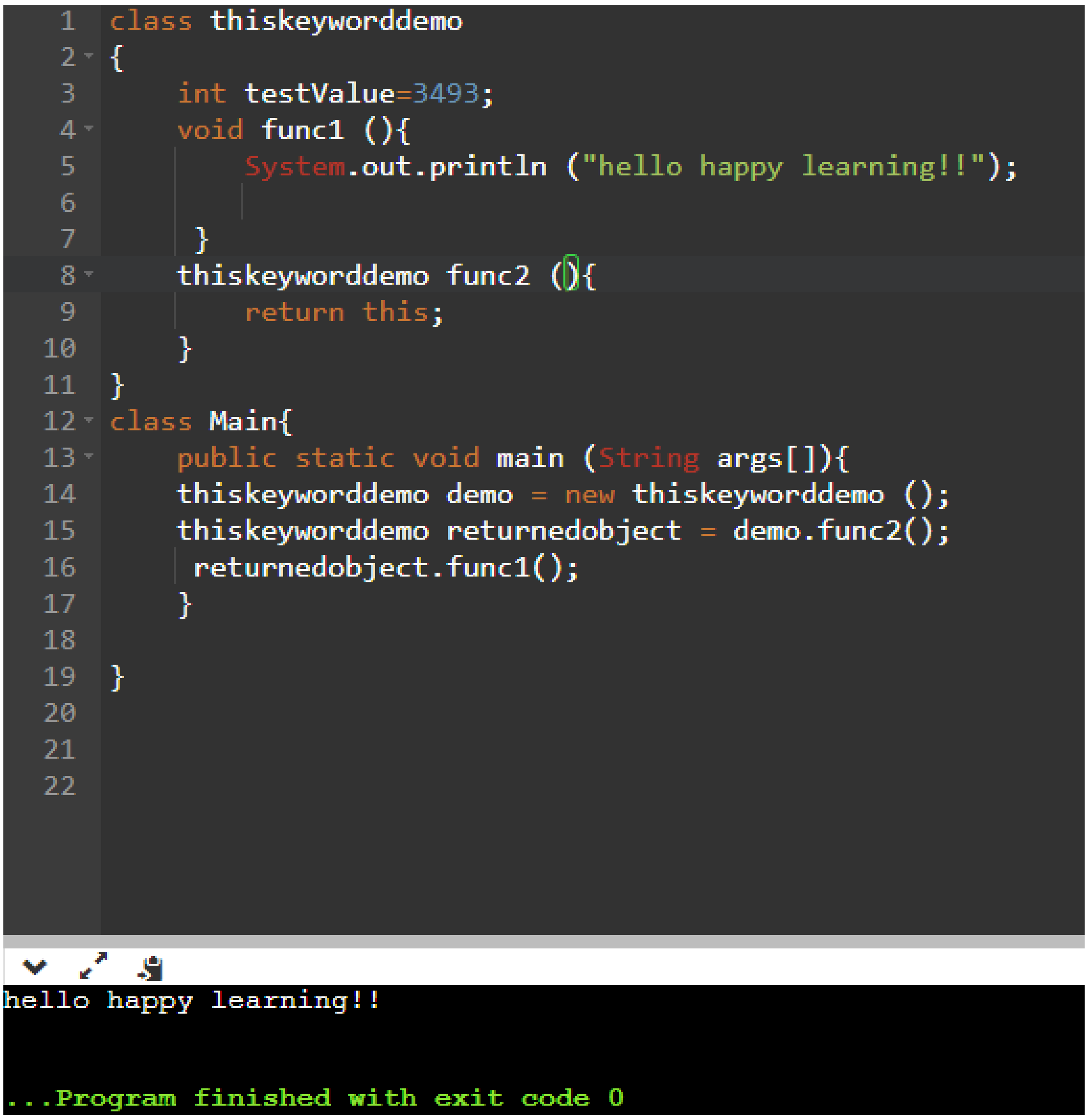


```
class thiskeyworddemo
{
        int testValue=3493;
        void func1 (){
            System.out.println ("hello you are myfunction");
             func2(this);
         }
        void func2 (thiskeyworddemo obj){
            System.out.println (obj.testValue);
        }
}
class Main{
        public static void main (String args[]){
        thiskeyworddemo demo = new thiskeyworddemo ();
        demo.func1();

        }

}
```

```
hello you are myfunction
3493

...Program finished with exit code 0
```

## this keyword can be used to return the current class instance

We can return this keyword as a statement from the method. In such a case, the return type of the method must be the class type (non-primitive). Let's see the example:



```
class thiskeyworddemo
{
     int testValue=3493;
     void func1 (){
          System.out.println ("hello happy learning!!");

      }
     thiskeyworddemo func2 (){
          return this;
     }
}
class Main{
     public static void main (String args[]){
     thiskeyworddemo demo = new thiskeyworddemo ();
     thiskeyworddemo returnedobject = demo.func2();
      returnedobject.func1();
     }

}
```

```
hello happy learning!!


...Program finished with exit code 0
```