*This Keyword***-**

It is used to refer current class variables, method and constructor.

Note- this keyword is not used in static context

Why?

**# Program-1** use of current class variable.

**class** Parent {

**int** x = 20;

}

**class** Child **extends** Parent {

**int** x = 25;

**void** test() {

**int** x = 30;

Child c = **new** Child();

//case 1

System.***out***.println("By creating objects=" + c.x);

//case 2

System.***out***.println("By using this keyword=" + **this**.x);

}

}

**public** **class** TestMain {

**public** **static** **void** main(String[] args) {

Child c= **new** Child();

c.test();

}

}

Output-

By creating objects=25

By using this keyword=25

In case 1, we are calling the x variable of child class for that purpose we loading the whole class that is not good programmer approach. For use of single variable, we should go for this keyword in java.

In case 2, we are trying to calling the x variable of current class by using the keyword that is the best approach because we are not wasting the memory here.

**# Program 2**-

2.1 program for use of current class method.

**class** Parent {

**void** test() {

System.***out***.println("Parent class method.");

}

}

**class** Child **extends** Parent {

**void** test() {

System.***out***.println("Child class method");

}

**void** demo() {

**this**.test();

}

}

**public** **class** TestMain {

**public** **static** **void** main(String[] args) {

Child c= **new** Child();

c.demo();

}

}

2.2

Output-

Child class method.

Super Keyword-

It is used to refer immediate parent class object, method and constructor.

Why?

# **Program-1** for use of immediate parent class objects.

**public** **class** Parent {

**int** x= 20;

}

**class** Child **extends** Parent {

**int** x = 25;

**public** **void** test() {

**int** x= 30;

//case1

Parent p = **new** Parent();

System.***out***.println("Parent class x variable=" + p. x);

//case 2

System.***out***.println("Immediate super class of child class x variable" + **super**. x);

}

}

**public** **class** TestMain {

**public** **static** **void** main(String[] args) {

Child c= **new** Child();

c.test();

}

}

Output-

Parent class x variable=20

Immediate super class of child class x variable20

In Scenario 1, we are calling the roll no of parent class for that purpose we loading the whole class that is not good programmer approach. For use of single variable, we should go for super keyword in java.

In Scenario 2, we are trying to print the roll no of immediate super class by using the super keyword that is the best approach because we are not wasting the memory here.

**# Program 2**- program for use of immediate super class method.

**class** Parent {

**void** test() {

System.***out***.println("Parent class method.");

}

}

**class** Child **extends** Parent {

**void** test() {

**super**.test();

}

}

**public** **class** TestMain {

**public** **static** **void** main(String[] args) {

Child c= **new** Child();

c.test();

}

}

Output-

Parent class method.

**Difference between super(),this() and super, this keyword:**

|  |  |
| --- | --- |
| **super() this()** | **super, this** |
| It is used for calling constructors | It is used to call method and variables |
| Using this, we can call immediate parent class and current class constructors | Using this, we can call immediate parent class and current class variables and methods. |
| It can be used only inside constructors | It can be used within instance(non-static) area only. |

*Point to be remember*

* We can use super() (or) this() only inside constructor. If we are using anywhere else we will get compile time error.
* We can use either super() (or) this() but not both simultaneously.
* We can use either super() (or) this() but not both simultaneously.

## *Final keyword*

We can apply final to variables, method and class

*a. .. Final variable*

A variable which is declared with final keyword called as final variables.

Once you assigned any value to that variables then it won’t be changed. It works like constants in java.

How to declare the final variables- final int a=5;

Example-1

**class** FinalDemo {

**public static void** main(String args[]) { **final int** a = 5; System.***out***.println(a);

}

}

Example-2

**class** FinalDemo {

**public static void** main(String args[]) {

**final int** a = 5;

**for** (**int** a = 5; a <= 10; a++) { System.***out***.println(a);

}

}

}

In this example, we will get compile time error, final variable values does not changed.

*b.Final method*

Method which is defined with final keyword called as final method. How to declare the final method-

**public final void** test(){

//business logic here.

}

Note- Final method cannot be overridden.

Example-3

**class** X {

**final void** test(){

System.out.println(“This is x **class**-test method”);

}

**class** Y **extends** X {

**final void** test(){

System.out.println(“This is y **class**-test method”);

}

**public static void** main(String args[]) {

X x = **new** Y();

x.test();

}

}

In this example, we will get compile time error final method cannot be override final method from X

*c. Final class*

The class which is defined with final keyword called as final class.

How to declare the final class

**final class** Test {

// business logic

}

How you stop others from inheriting your class- By making class as final.

**final class** X {

**public final void** test() {

System.***out***.println("x class-test method");

}

}

**class** Y **extends** X {

}

**public class** FinalKeyword {

**public static void** main(String[] args) {

Y y = **new** Y();

y.test();

}

}