**KEYWORDS**

Keywords are words that have already been defined for Java compiler. They have special meaning for the compiler. Java Keywords must be in your information because you can not use them as a variable, class or a method name.

You can't use keyword as identifier in your Java programs, its reserved words in Java library and used to perform an internal operation.

|  |  |  |  |
| --- | --- | --- | --- |
| abstract | assert | boolean | break |
| byte | case | catch | char |
| class | const | continue | default |
| do | double | else | enum |
| extends | final | finally | float |
| for | goto | if | implements |
| import | instanceof | int | interface |
| long | native | new | package |
| private | protected | public | return |
| short | static | strictfp | super |
| switch | synchronized | this | throw |
| throws | transient | try | void |
| volatile | while | true | false |
| null |  |  |  |

true, false and null are not reserved words but cannot be used as identifiers, because it is literals of built-in types.

**Static Keyword**

* Static is a keyword that acts as a non-access modifier in Java that is used mainly to manage memory.
* The variable or Method that are marked static belongs to the Class rather than to any particular instance.
* A Static method cannot access an instance variable.
* If a Class contains any static blocks, then that block will be executed only when the Class is loaded in JVM.

Programmers can apply the Java keyword static with different programming objects like:

* variables
* methods
* initialization - block
* nested class

**Static keyword is unacceptable to the following**

* Class
* Constructor
* Local inner classes
* Inner class methods
* Instance variables
* Local Variables
* Interfaces

1. Static variables or methods can be invoked without having an instance of the class.
2. Only a class is needed to call up a static method or a static variable. If you declare any variable as static, it is known static variable.
3. The static variable can refer to a common property of all objects (that is not unique for each object), e.g. company name of employees, college name of students, etc.
4. Memory in a static variable is reserved only once in a class area at the time of class loading. One advantage of using static is that it increases the efficiency of the memory.

**Rules for using Static Variables**

* Variables or methods belong to class rather than to any particular instance
* A static method cannot access a non-static variable of a class nor can directly invoke non-static method
* Static members can be applied without creating or referencing an instance of the class
* A static variable will be shared by all instances of that class which will result in only one copy

**Program for Static in Java**

**Static Variable:**

//Java Program to demonstrate the use of static variable

class Student{

int rollno;//instance variable

String name;

static String college ="ITS";//static variable

//constructor

Student(int r, String n){

rollno = r;

name = n;

}

//method to display the values

void display (){System.out.println(rollno+" "+name+" "+college);}

}

//Test class to show the values of objects

public class TestStaticVariable1{

public static void main(String args[]){

Student s1 = new Student(111,"Karan");

Student s2 = new Student(222,"Aryan");

//we can change the college of all objects by the single line of code

//Student.college="BBDIT";

s1.display();

s2.display(); } }

Output:

111 Karan ITS

222 Aryan ITS

**Static method:**

**Restrictions for the static method**

* There are two main restrictions for the static method. They are:
* The static method can not use non static data member or call non-static method directly.
* this and super cannot be used in static context.

**public** **class** StaticExample {

**static** **void** add() {

System.***out***.println("g");

}

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

*add*();

}

}

**3) Java static block**

* Is used to initialize the static data member.
* It is executed before the main method at the time of classloading.

Example of static block

class A2{

static{System.out.println("static block is invoked");}

public static void main(String args[]){

System.out.println("Hello main");

}

}

Output:

static block is invoked

Hello main

**Final Keyword**

Final is a keyword in Java that is used to restrict the user and can be used in many respects. Final can be used with:

* Class
* Methods
* Variables

**Class Declared as Final:**

When a class is declared as final, it cannot be extended further. Here is an example what happens within a program

final class stud {

// Methods cannot be extended to its sub class

}

class books extends stud {

void show() {

System.out.println("Book-Class method");

}

public static void main(String args[]) {

books B1 = new books();

B1.show();

}

}

This will show an error that:

error: cannot inherit from final stud

class books extends stud{

^

This is because classes declared as final cannot be inherited.

**Method Declared as Final:**

* A method declared as final cannot be overridden; this means even when a child class can call the final method of parent class without any issues, but the overriding will not be possible.
* Here is a sample program showing what is not valid within a Java program when a method is declared as final.

class stud {

final void show() {

System.out.println("Class - stud : method defined");

}

}

class books extends stud {

void show() {

System.out.println("Class - books : method defined");

}

public static void main(String args[]) {

books B2 = new books();

B2.show();

}

}

**Variable Declared as Final:**

* Once a variable is assigned with the keyword final, it always contains the same exact value.
* Again things may happen like this; if a final variable holds a reference to an object then the state of the object can be altered if programmers perform certain operations on those objects, but the variable will always refer to the same object.
* A final variable that is not initialized at the time of declaration is known as a blank final variable. If you are declaring a final variable in a constructor, then you must initialize the blank final variable within the constructor of the class. Otherwise, the program might show a compilation error.

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class stud {

final int val;

stud() {

val = 60;

}

void method() {

System.out.println(val);

}

public static void main(String args[]) {

stud S1 = new stud();

S1.method();

}}

This Keyword:

* this is another Java keyword which as a reference to the current object within an instance method or a constructor — the object whose method or constructor is being called.
* By the use of this keyword, programmers can refer to any member of the current object within an instance method or a constructor.
* This keyword can be used to refer to the current object, and it always acts as a reference to an object in which method was invoked.

There are the various uses of this keyword in Java.

These are:

* For referring current class instance variable, this keyword can be used
* To invoke current class constructor, this() is used
* this can be passed as message argument in a method call
* In a constructor call, this can be passed as argument
* For returning current class instance, this keyword is used

class Emp {

int e\_id;

String name;

Emp(int e\_id, String name) {

this.e\_id = e\_id;

this.name = name;

}

void show() {

System.out.println(e\_id + " " + name);

}

public static void main(String args[]) {

Emp e1 = new Emp(1006, "Karlos");

Emp e2 = new Emp(1008, "Ray");

e1.show();

e2.show();

}

}

**this() : to invoke 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.

**Calling default constructor from parameterized constructor:**

class A{

A(){System.out.println("hello a");}

A(int x){

this();

System.out.println(x);

}

}

class TestThis5{

public static void main(String args[]){

A a=new A(10);

}}

Test it Now

Output:

hello a

10

**Calling parameterized constructor from default constructor:**

class A{

A(){

this(5);

System.out.println("hello a");

}

A(int x){

System.out.println(x);

}

}

class TestThis6{

public static void main(String args[]){

A a=new A();

}}

Output:

5

hello a

**Super Keyword:**

* Super is a keyword of Java which refers to the immediate parent of a class and is used inside the subclass method definition for calling a method defined in the superclass.
* A superclass having methods as private cannot be called.
* Only the methods which are public and protected can be called by the keyword super.
* It is also used by class constructors to invoke constructors of its parent class.

Syntax:

super.<method-name>();

**Usage of superclass**

* Super variables refer to the variable of a variable of the parent class.
* Super() invokes the constructor of immediate parent class.
* Super refers to the method of the parent class.

Here is an example of Java program which uses the super keyword

Example:

class employee {

int wt = 8;

}

class clerk extends employee {

int wt = 10;  //work time

void display() {

System.out.println(super.wt); //will print work time of clerk

}

public static void main(String args[]) {

clerk c = new clerk();

c.display();

}

}

Output:

8