------ Declaration and Access Modifier----

30-06-2023

Only access Modifier for top level classes

What is difference between access specifier and access modifier

Public classes

Default classes

Final Modifier

Final method

Final class

Abstract modifier

Abstract Methods

Abstract Classes

What is difference between abstract class and abstract method?

What is difference between abstract and final?

Strictfp

What is difference between strictfp and abstract?

Member Modifier

Public Member

Private Member

Default Member

Protected Member

Comparison of public,private,default,protected

Final variables

Final instance variable

At the time of Declaration

Inside Instance block

Inside constructor

Final static variable

At the time of Declaration

Inside static block

Final Local Variable

Formal Parameter

Static modifier

Native Modifier

Synchronized modifier

Transient Modifier

Volatile modifier

Summary of modifier

What is modifier?

Modifier will talk about the visibility of classes and methods means that class or method access from anywhere or not.

Class Modifer

Whenever we are writing a java program we have provide some information about our classs to jvm like

1.whether class is accessible from anywhere or not

2.whether object creation is possible or not

3.whether child class creation is possible or not.

We can specify these information by using corresponding access modifier and only application modifier for top level classes are

Public,abstact,final,default,strictfp

If we declare class with any other access modifier we will get compile time error.

Eg

Private class A{

Main(){

}

}

This code will give compiler time error as we cant declare the class with private access modifier.

But for inner classes following access modifier are allowed

Private,protected,static.

---Public classes---

If we declare a class as public then that class we can access from anywhere that we can access it within the package and outside the package also.

Eg

Package test;

Public Class Test1{

Public void m1(){

Sysout(“m1”);

}

}

Package test2;

Public class Test2{

Public void m2(){

Sysout(“m2”)

Main(){

Test t1=new Test();

T1.m1()//able to access m1() method of Test1 class outside the package if class declare as public

}

}

}

Default class

If we declare a class as default then that class is only accessible within the current package and it is also known as package level access.

Final Mofifier.

Final is the modifier applicable classes ,method and variable.

Final Method:

-Whatever the method parent has by default available to its child

-if we want child class cannot override implementation of parent class in such we have to declare method as final

Final class:

If we declare the class with final keyword then we cannot create child for that class means inheritance concept is not applicable for final clasess.

Every method present inside final class is always final whether we are declaring or not but every variable need not to be final.

The main advantage of final modifier is we can achive security but we are missing key benefit of opps that polymorphism.

Abstarct Modifier:

Abstract is the modifier applicable only for method and classes.

Abstract Method

If we don’t know anything about the implementation then we should declare the method as abstaract where child classes are responsible to provide implementation for it.

Abstarct class.

For any java class if we are not allow to create the object such type of class we should declare with abstract.

Strictfp:

Strictfp is the modifier applicable for classes and methods but not for variable

Generally values for arithmetic floating point values from platform to platform to achive platfrom independent values

Suppose

10/3=3.33(window)

10/3=3.34(mac os)

10/3=3.35

(IEEE754)

Strictfp Class test{

Public void strictfp m1(){

Sysout(10/3)

}

}

Difference between strictfp and abstract?

If we declare any method with strictfp modifier then we should always provide the implementation but if we declare a method with abstract then we don’t need to worry about implementation.

Public Member

If a member is declared as public then that member we can access from anywhere.But before checking Member Visiblity we have check class visibility.

Program--1

Package test;

Public class A{

Public void m1(){

Sysout(“m1() ”);

}

}

Program—2

Package test2;

Public class B{

Main(){

A a=new A();

a.m1();//here I am able to access m1() method of A class as it is declared as public and also corresponding class is public.

}

}

Private member:

If a member is declared as private then we can access it only within the current class and we can’t access it from outside the class.

Private methods are not visible outside the classes so we cant declared the private method as abstract.mean private abstract combination is illegal.

Protected Member:

If a member is declared as protected then that member we can access within the current package anywhere but outside the package we can access only in child classes.

-we can access protected member within the current package anywhere either by child reference or parent reference.

-outside the package we can access protected member only in the child classes and with child reference we can not access it by using parent reference.

Default Member:

If a member is declared as default then that member we can access it only within the current package hence default member is also known as package level access.

Comparison of public,private,protected,default.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| visibility | Public | Private | Protected | Default |
| Within the same  Class | Yes | Yes | Yes | yes |
| From child classes of same package | Yes | No | Yes | Yes |
| From Non-child classes of same Package | Yes | No | Yes | Yes |
| From child classes outside the package | Yes | No | Yes | No |
| From non-child class outside the  Package | Yes | No | No | No |
|  |  |  |  |  |

-The most accessible modifier is public

-The least accessible modifier is private.

Recommended access modifier for variable is private and for method is public.

Instance variable

If the values of a varible varies from object to object such type of variable called instance variable and for instance variable if we don’t perform initialization then JVM will provide the default value.

Final instance variable

If we declare the instance variable as final then JVM wont provide any default values and programmer should perform the inititialization of it before constructor completion.

We can perform initialization for final variable in various places.

At the time of declaration

Class A{

Final int i=10;

}

Inside instance block

Class A{

Final int i;

{

i=10;

}

}

Inside constructor

Class A{

Final int I;

A(){

i=10;

}

}

If we perform initialization anywhere else we are going to get compile time error.

Static Variable:

If the values of a variable not varied from object to object such type of variable is not recommened to declare as static it is recommened to declared as static.

In this case instance variable for every object one copy will be created but in the case of static variable single copy will be created at class level and shared by every object of that class.

If we declaring a static variable JVM will provide the default Value.

But if we declare the static variable as final then such case we should the default value whether we are using it or not otherwise it will give compile time error.

Rule:

We should perform the initialization of static final variable before class loading completion means these places are possible to perform initization—

At the time of decration

Class A{

Static final int a1=10;

}

Inside static Block

Class A{

Final int i;

Staic {

i=10;

}

Final local variable

Sometimes to meet the temporary requirement of programmer we are declaring a variable inside a method or block or constructor such type of variable is called local variables.

For local variable JVM wont provide any default value we should perform initialization of local variable before using it.

If we declare the local variable as final then also we should perform initialization of it before using it otherwise we will get compile time error.

The only applicable modifier for local variable is final and we can t use any other modifier on local variables.

Formal Parameter:

Formal parameter act as local variable of method hence we can declare local variable as final

If we declare formal parameter as final then we cant change its values.

Ex

Class A{

Public void add(final int x,final int y){

X=100;//compiler time error

Y=200

}

Main(){

Add(10,20);

}

}

Synchronized modifier:

Synchronized is the modifier applicable for methods and blocks not for classes and variable.

If a method or block is declared using synchrized keyword then at a time only one thread is allow to access that method or block in the given object.

The main advantage of synchrized keyword is we can resolve data inconsistency problem.

But the main disadvantage of synchronized keyword is that it increased the waiting time of other threads and effect the performance so if there is specific requirement we should not use synchrozed keyword.

Transient modifier:

Transient is the modifier applicable only for variables but not for methods and classes.

During the process of serialization if we don’t want to expose the original value of a variable to meet security constraint such type of variable we should declare with transient keyword.So During the process of serialization JVM will ignore the original value and save default value.

Static variable are not part of object so declaring a static variable as transient there is no impact or there is no use.

If we declare the final variable as transient it will participate in serialization with its original values.

Volatile modifier:

Volatile is the modifier applicable only for variable not for classes and methods

If the value of a variable keep on changing such type of variable we should declare as volatile.

If a variable is declared with volatile keyword then for every thread JVM will create a separate local copy all intermediate modification will take places in local copy instead of master copy.

Once the value got finalized before terminating thread that final value will be updated to master copy.

The main advantage of volatile keywod is