------ 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/relation 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

Interfaces

Interfaces declaration and implementation

Extend vs implement

Interface methods

Interface variables

Interface naming conflicts

Method naming conflicts

Variable naming conflicts

Marker interface

Adapter Class

Interface vs Abstract class vs concrete class

Difference between abstract class and Interface

Conclusion

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);

}

}

Static Modifier:-

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

We can declare inner classes with static modifier but not top level classes

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

Instance variable can be access from instance area direct but we can not access from static area directly

Static variable can be access from both static and instance area directly.

Case1:

Overloading concept is applicable for static method including main method but JVM will always called (String[] args) method.

Overloaded method we have to call explicitly and it will be executed just like a normal method

Case3:

Inheritance concept is applicable for static method including main method hence while executing child class if child does contain main method then parent class main method will be executed.

Case3:

Overiding concept is not applicable for static method it is method hiding.

Method Hiding:-

If child class contains a static method with same name as parent class method then it seems to be orverriding but it is method hiding

Native Modifier:

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

The methods which are updated in non-java are called are called native methods or foreign method.

For native methods implementation is already available and we are not responsible to provide implementation hence native method declaration should compulsory ends with semicolon.

Public native void methodOne()----invalid

Public native void methodOne();---valid

• For native methods implementation is already available where as for abstract methods implementation should not be available child class is responsible to provide that, hence abstract native combination is illegal for methods.

• We can't declare a native method as strictfp because there is no guaranty whether the old language supports IEEE754 standard or not. That is native strictfp combination is illegal for methods. • For native methods inheritance, overriding and overloading concepts are applicable.

• The main advantage of native keyword is performence will be improves.

• The main disadvantage of native keyword is usage of native keyword in java breaks platform independent nature of java language.

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 we can resolve consistency problem.

Interfaces:

Interface will defines the rules/guidelines/contract that its child/implementation class should follow.

Or

Interface is considered as contract between client and services provider.

Example:

SunMincrosystem has provided JDBC API database vendor has to provide its implementation.

sunMicrosystem has provide Servler API web Application vendor has to provide its implementation.

Declaration and Implementation:

Note:

1.Whenever we are implementing an interface compulsory for every method of that interface we should provide the implantation otherwise we should declarre the class as abstract so that where child class should provide the implementation of remaining method.

2.whenver we are implementing an interface method compulsory we should declare the method as public otherwise we will get the compile time.

Extend vs implement

A class can extend only one class at a time

A class can implement any no of interfaces at a time

Interface Method.

Every Method present inside the interface is always public abstract whether we are declaring or not.

public interface Test {

public abstract void m1();//

void m2();//same

}

Public:to make this method available for every implementation class.

Abstract:implementation classes are responsible to provide its implementation

As every method present inside the interface is public and abstract so we can not use the following modifier

Private protected final,strictfp,synchozided etc,

Interface variable

An interface can contatain variables

The main purpose of interface variable is to define requirement level constant

Every interface variable is always public static final whether we are declaring or not

Interface test{

Int x=10;

Public static final int x=10//both declaration are same

}

Public:to make it available for every implementation class

Static:without existing object also we have to access this variable

Final:implementation class can access this value but cannot modify

Hence inside interface the following declarations are equal.

int x=10; public int x=10;

static int x=10; final int x=10;

public static int x=10;

public final int x=10;

static final int x=10;

public static final int x=10;

all are equal

As every interface variable by default public static final we can't declare with the following modifiers. Private Protected Transient Volatile

For the interface variables compulsory we should perform initialization at the time of declaration only otherwise we will get compile time error.

Interface variable can be accessed in implementation class but can not be modified

Method Naming conflicts:

Case1:

If two interface contains method with same signature and return type then in the implementation class only one method implementation is enough.

Interface A{

Void m1();

}

Interface B{

Void m2();

}

Class Test implements A,B{

Public void m2(){

Sysout(“m2 method called()”)

}

}

Case 2:

If two interface contain Method with same name but with different argument then in the implementation class we should provide the implementation for both methods and these method act as overloaded methods.

Eg

Interface A{

Void m1();

}

Interface B{

Void m1(int i);

}

Class Test implements A,B{

Void m1(){

Sysout(“m1()”)

}

Void m1(int i){

Sysyout(“m1”+i);

}

Case 3

If two interface contain a method with same signature but different return type then it is not possible to implement both interface simultaneously.

Eg:

Interface A{

Void m1()

}

Interface B{

int m1()

}

Class Test implements A,B{

}//not valid

Variable Naming conflicts:

Two interface can contain a variable with same name and there could be chance of variable naming conflicts and we can resolve this by using interface name.

Eg:

Interface A{

Int i=10;

}

Interface B{

Int i=20;

}

Class Test implements A,B{

Sysout(A.i);//10

Sysout(B.i);//20

}

//interview

Marker Interface:

If an interface does not contain any method and by implementing that interface if out object will get ability such type of interface is called marker interface or Tag interface or ability interface

Examle of marker interface

Serializable,

Cloneable

Random Access

//these are marked for some extra ability

By implementing serializable interface we can object across the network and we can save the state of an object into a file.

By implementing Cloneable Interface our object in a position to provide exactly duplicate cloned object.

Without having any methods in marker interface how objects will get ability?

Internally JVM is responsible to provide required ability

Why JVM is providing required ability ?

To reduce the complexity of programming

Can we create our own marker interface?

Yes but we need to customize the JVM.

Adapter Class:

Adapter class is simple java class that implements an interface only with empty implementation for every method.

If we are implementing interface directly then for each and every method we should provide the implementation whether it is required or not.this approach increased length of the code and reduces readability.

interfaec X{

void m1();

void m2();

;;;;;;;

;;;;;;;

void m100();

}

//Adapter class

class Test implements X{//adapter class

void m1(){

}

void m2(){

}

;;;;;;

void m100(){

}

}

class Test1 extends Test{

void m1(){

//new logic

}

}

Adapter class is a simple java class that implements an interface and provide empty implementation for all the methods.

interface:

if dont know anything about implementation then we should go for interface

abstract class

if know about the implmentation partially then we should go for abstract class

concreate class

if know about the implementation completely then we should go for concrete class.

//Interview

|  |  |
| --- | --- |
| Interface | Abstarct class |
| 1.if we don't' know anything about implementation just we have requirement specification then we should go for interface. | If we are talking about implementation but not completely (partial implementation) then we should go for abstract class. |
| Every method present inside interface is always public and abstract whether we are declaring or not. | Every method present inside abstract class need not be public and abstract. |
| We can't declare interface methods with the modifiers private, protected, final, static, synchronized, native, strictfp. | There are no restrictions on abstract class method modifiers. |
| Every interface variable is always public static final whether we are declaring or not. | Every abstract class variable need not be public static final. |
| Every interface variable is always public static final we can't declare with the following modifiers. Private, protected, transient, volatile. | There are no restrictions on abstract class variable modifiers. |
| For the interface variables compulsory we should perform initialization at the time of declaration otherwise we will get compile time error. | It is not require to perform initialization for abstract class variables at the time of declaration. |
| Inside interafec we cant write instance and static block | Inside abstract class we can take both static and instance blocks. |
| Inside interface we can't take constructor. | Inside abstract class we can take constructor. |

Why abstract class can contain constructor where as interface cannt conatin constructor?

The main purpose of constructor is to perform initialization of an object i.e., provide values for the instance variables, Inside interface every variable is always final,static and there is no chance of existing instance variables. Hence constructor is not required for interface.

But abstract class can contains instance variable which are required for the child object to perform initialization for those instance variables constructor is required in the case of abstract class.

Java Modifiers Important Questions

1) For the Top Level Classes which Modifiers are Allowed?

2) Is it Possible to Declare a Class as static, private, and protected?

3) What are Extra Modifiers Applicable for Inner Classes when compared with Outer Classes?

4) What is a final Class?

5) Explain the Differences between final, finally and finalize?

6) Is Every Method Present in final Class is final?

7) Is Every Variable Present Inside a final Class is final?

8) What is abstract Class?

9) What is abstract Method?

10)If a Class contain at least One abstract Method is it required to declared that Class Compulsory abstract?

11)If a Class doesn’t contain any abstract Methods is it Possible to Declare that Class as abstract?

12)Whenever we are extending abstract Class is it Compulsory required to Provide Implementation for Every abstract Method of that Class?

13)Is final Class can contain abstract Method?

14)Is abstract Class can contain final Methods?

15)Can You give Example for abstract Class which doesn’t contain any abstract Method?

16)Which of the following Modifiers Combinations are Legal for Methods?

18)What is the Difference between abstract Class and Interface?

19)What is strictfp Modifier?

20)Is it Possible to Declare a Variable with strictfp?

21)abstract - strictfp Combination, is Legal for Classes OR Methods?

22)Is it Possible to Override a native Method?

23)What is the Difference between Instance and Static Variable?

24)What is the Difference between General Static Variable and final Static Variable

25)Which Modifiers are Applicable for Local Variable?

26)When the Static Variables will be Created?

27)What are Various Memory Locations of Instance Variables, Local Variables and Static Variables? 28)Is it Possible to Overload a main()?

29)Is it Possible to Override Static Methods?

30)What is native Key Word and where it is Applicable?

31)What is the Main Advantage of the native Key Word?

32)If we are using native Modifier how we can Maintain Platform Independent Nature?

33)How we can Declare a native Method?

34)Is abstract Method can contain Body?

35)What is synchronized Key Word where we can Apply?

36)What are Advantages and Disadvantages of synchronized Key Word?

37)Which Modifiers are the Most Dangerous in Java?

38)What is Serialization and Explain how its Process?

39)What is Deserialization?

40)By using which Classes we can Achieve Serialization and Deserialization?

41)What is Serializable interface and Explain its Methods?

42)What is a Marker Interface and give an Example? 4

3)Without having any Method in Serializable Interface, how we can get Serializable Ability for Our Object?

44)What is the Purpose of transient Key Word and Explain its Advantages?

45)Is it Possible to Serialize Every Java Object?

46)Is it Possible to Declare a Method, a Class with transient?

47)If we Declare Static Variable with transient is there any Impact?

48)What is the Impact of declaring final Variable a transient?

49)What is volatile Variable?

50)Is it Possible to Declare a Class OR a Method with volatile?

51)What is the Advantage and Disadvantage of volatile Modifier?

Interfaces Important Questions

1. What is Interface?
2. What is Difference between Interface and Abstract Class?

// 3) When we should Go for Interface and Abstract Class and Concrete Class?

4) What Modifiers Applicable for Interfaces?

5) Explain about Interface Variables and what Modifiers are Applicable for them?

6) Explain about Interface Methods and what Modifiers are Applicable for them?

7) Can Java Class implement any Number of Interfaces?

8) If 2 Interfaces contains a Method with Same Signature but different Return Types, then how we can implement Both Interfaces Simultaneously?

9) Difference between extends and implements Key Word?

10)We cannot Create an Object of Abstract Class then what is Necessity of having Constructor Inside Abstract Class?

11)What is a Marker Interface? Give an Example?

12)What is Adapter Class and Explain its Usage?

13)An Interface contains only Abstract Methods and an Abstract Class also can contain only Abstract Methods then what is the Necessity of Interface?

14)In Your Previous Project where You used the following Marker Interface, Abstract Class, Interface and Adapter Class?