**JAVA**

Java Source File Structure-Introduction

Java can contains any number of classes and atmost one class shouod be public .

If the class is declared as public its file name should be same as the class name but if the the class is not declared as pulbic you can assign any name to the file no restriction is imposed .

More than one public class is not allowed in a single java program .

Its a misconception that class that has the main method should be the name of the java file there is no relation between them. But the class which is public should be the name of the java file.

Whenever we compile a java program , each class present in the program will have seperate .class file .

When the class does not have a main method and we try to execute it will throw up an error .

When we multiple class a , b , c , and d and we give the file name as yash and when we try to java yash it will throw an error as yash .class file is not created as there is no class name as yash in the file name [yash.java](http://yash.java)

So whenever we are executing a class the class main method is executed.

Java Source File Structure- Import Statement Introduction

java.util.ArrayList l = new java.util.ArrayList(); fully qualified name . So if we want to create 10 array list objects we have ti write this code 10 times , which will increase the length of the code . So to overcome this problem we use the import statement . By which we can directly write the short name as

import java.util.ArrayList;

ArrayList l = new ArrayList();

So import statement acts as a best typing shortcut .

They are two types of import statement

Explicit and implicit

Explicit : Example: import java.util.ArrayList; this feature is recommended as it improves the rate of plotting.

Implicit : import java.util.\* ; this decreases the rate of plotting. Not at all recommend .

Some people say that explicit is used when we have one or two classes in the program and implicit import is used when we have 100 of classes . But its a misconception its always recommended to use explicit import in java .

Import java.hdfc.\*;

Import java.icic.\*;

Account a = new Account ();

Loan l = new Loan ();

Now we are not able to distinguish from which bank the account is coming or from which bank the loan is coming . So to overcome this we should use explicit import .

Import java.hdfc.Account;

Import java.icic.Loan;

Account a = new Account ();

Loan l = new Loan ();

So it is highly recommend to use explicit import as it improves the rate of plotting of the code.

There are two packages for which import statement is not required

1. java.lang as it is by default present in the code

Example: String s = new String();

Thread t = new Thread();

Exception e = new Exception ();

StringBuffer sb = new StrungBuffer("Yash");

we dont need to use import statement for it as it classes and interfaces are by default present in the code .

2. default package i.e. cwd ( current working directory ) so any package present in the same directory we don't need to use import statement but if its not present in the cwd we must need to import it .

Whenever we are importing a package all the classes and interfaces present in that package are by default available not the subpackage . To use subpackage import statement must be used till that package . Ex : import -> util -> regex -> pattern to use the pattern we should write till import java.util.Regex; it will throw an if we write import java.util.\*; as pattern is present in regex package .

 Java Source File Structure- Package Statement

Package : a group of related items or things is nothing but package. In program a group of packed related classes and interfaces grouped together is nothing but package . So package is an encapsulation mechanism to group together all the related classes and interfaces in a single unit .

Example : java.sql package

Importance of package in java

Java have two date classes present in sql and util package .

So with the help of the package we can differentiate between which package s we want to use . i.e. unique identification of the componenets . Group of related classes in same unit . i.e. modularity.

It is highly mandatory to use package statement as it shows professionalism and also it acts as a security wall between the program and the other users . Package is a keyword in java .

Format to write package :

package com.(Name of the package )

(-d )say as hyphen d : destination to palce generated .class file .

package com.yash.jha

class Test

{

psvm(String args [])

{

Sopln("Package");

}

}

So to compile we can use javac [Test.java](http://test.java) but it will place the class file in cwd (current working directory ) cwd -> test.class

But to place it in package we will write

javac -d . [Test.java](http://test.java) this will put our .class file in packgae .

Instead of dot that we used after -d we can assign any available directory .

cwd -> com -> yash -> jha -> Test.class

Here the dot after hypen d represents cwd . If yash ,jha folders are not available, java by default create the same folder in the same folder structure manner .

To know the location of .class we can use dir [classname.java](http://classname.java)

javac -d . [Test.java](http://test.java)

So to compile a package use javac -d . [Test.java](http://test.java)

and to execute thr program use : java com.yash.jha.Test

Important Points about Package Statement

1. In any java source file i.e. java program atmost 1 package is allowed . If you use more than package statement compiler will give left and right 😂 . We can also write the package i.e. syntax of package as package pack1; com is not necessary. We can take any number of import statement.

2. import java.util.\*;

package pack1;

class test

{

  } this source file will not be compiled successfully as we have declared package after import which is not alllwed in java . In java the first statement should be package .

Package statement must be our first statement in our java program .

So the correct order is :

package statement;

import statement;

Class / enum / interface

Class , Enum or Interface Expected Error

The Java compiler displays the error message "class, interface, or enum expected" when it comes across a statement that it cannot interpret because it is not enclosed in a class, interface, or enum.

Here are a few reasons that could cause this error:

There are too many or too few curly braces in the code.

A method is defined outside of a class.

The class has not been declared.

Multiple packages are declared in the same file.

Here are a few ways to fix this error:

Check the curly braces to make sure there are the same number of opening and closing braces.

Move any methods that are defined outside of a class into a class.

Declare the class before using it.

Make sure that only one package is declared in each file.

Class Level Modifiers: public and default

It is important to specify the class modifier ss it tells the JVM about the accessibility of our class .

Modifier describe the behavior of the class.

So if we declared a class as public we can access it from anywhere i.e. within the package as well as outside the package . If the class is declared as default we can only access it within the package .

If the class is abstract we cannot create the object and vice versa . So if the class is not abstract instantiation ( object creation ) is possible.

If the class is final child creation is not possible and vice versa.

Which modifier are allowed for top level classes ?

public , abstract , default , final , strictfp(Strict floating point ) . For inner classes i.e. (classes declared inside a class ) we can use the above mentioned modifiers as well as the protected , private and static . So protected , private and static are only available for inner classes not for top level classes .

package pack1;

public class A

{

}

package pack2;

import pack1.A;

public class B

{ A a = new A();

     }

To access the other package we have to use the import statement. So this program will compile successfully .

If the class A was not delared as public and it was declared as a default class then the above program will be give error as , only public class is accessible within and outside the package and not the default class .

Sp the default class is accesoble within the package but not outside the package .

Default access is also known as package level access .

Abstract Modifier

There are abstract class and abstract methods .

In general, Abstract means partial implementation , not clear .

public class vechicle

{ public int getnoofwheels()

return ||

   }

So here we are not told about what is the type of vechile whether its a bike , bus , car . So here we dont know what we should return . So for this stituation abstract comes into play.

So when we dont about the implementation of a method in java we can use the abstract modifier to implement the method . So abstract method has only declaration not the implementation . As child classes are responsible for implementation. Abstract method declaration shoud end with semicolon not with curly braces .

Syntax for abstract method :

public abstract returntype methodname();

If we use curly braces for the abstract method it will give error stating that abstract method cannot have a body .

Note : if a class has atleast one abstract method then it is compulsory that the class should be declared as abstract class .

So access method is that method which has the declaration but not the implementation.

Abstract Class : a class that is not implemented is known as abstract class i.e. if we don't know proper implementation of the class we can happily use abstract class . If the class is declared as abstract we cannot create the object of the class as well we can call the memebers of the class. So a partially implemented class is declared as an abstract class .

So a partially implemented class is known as abstract class and it is impossible to create the object of the abstract class. That is instantiation of the class is not possible .

Relationship between Abstract class Vs Abstract method

If a class contains atleast one abstract method the class must be declared as abstract class . As when we try to create the object of the class and with then we try to call the method with the help of the object then we will not be able to get any result as the method is declared as abstract so it doesn't have any implementation thus it is important to declare the class as abstract if the class contains atleast one abstract method .

class test

{

public abstract int m1();

{

 Test t = new Test();

t.m1();

}

} so the above program will give error as the method is not implemented. So it is advisable to use the class as abstract.

If the class does not contain any abstract method we can still make the class as abstract class.

If the class is an abstract class and it has method lets saybit has two methods then it is the responsibility of the child class to impelment the method . If the child class implements only one method then its the responsibility of the next child class to implement the second method .

Ex :

abstract class Vechicle

{

    public abstract int getnoofwheels();

}

class Bus extends Vechicle

{

     public int getnoofwheels()

       {

             return 6;

         }

}

class Auto extends Vechicle

{

     public int getnoofwheels ()

      {

          return 3;

      }

}

class Test

{

     psvm (String [] args)

     {

         Bus b = new Bus();

         Sopln(b.getnoofwheels());

         Auto a = new Auto();

         Sopln(a.getnoofwheels());

     }

}

public abstract int getnoofwheels(); if we don't write this in our code still the program will run so whats the advantage of writing it ? The main advantage is when we write abstract method its compulsory for the next child class to provide the implementation for the method . But if we dont declare the method as abstract then the child class may or may not provide implementation for the method .

Member modifiers : public , default ,private ,protected

Member means it may be a variable or may be a method .

If the class is not declared as public but its member is public and we try to access these memeber outside the package the compiler will give error . So to access a member outside the package the class must be declared as public .

If the member is declared as default we can only access it within the same package . Therefore it's also called package level.

If the memeber is declared as private we can access it within the same class . Therefore it's also called class level.

Data memebers are known as variables .

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

If a memeber is declared as protected you can acess it within the current package but outside package it is only accessible in child classess .

So we can write protected as

Protected = <default modifier> + kids (only in child classess)

class A

{

    protected void m1()

    {

        Sopln(" A class protected method ");

        }

}

class B extends A

{

  psvm(String args[])

    {

A a = new A ();

a.m1();

B b = new B();

b.m1();

A a1 = new B();

a1.m1();

}

} Parent reference(a1) can be used to hold child objects(new B()) .

package pack1;

public class A

{

    protected void m1()

    {

        Sopln(" A class protected method ");

        }

}

package pack2;

import pack1.A;

public class B extends A

{

  psvm(String args[])

    {

A a = new A ();

a.m1();

B b = new B();

b.m1();

A a1 = new B();

a1.m1();

}

} here

A a = parent class reference new A () : parent class Object .

A a1 = parent class reference new B () : child class Object .

B b = child class reference new B () : child class Object .

So when we use protected method it is accessible within the same package and if it is another package we can access it by using child classess but the extra masala here is we can only use child reference to access to the method . If we use parent reference to access the method the compiler will give left and right and the error will be displayed as if we consider the above program as : m1 has a protected access in pack1.A.

Summary for Access Modifiers

private < default < protected < public

Interface : Introduction:

Interface : any service requirements specifications or any contact b/w client and service provider is called an interface.

Every method present inside a i terface is by default is public and abstract whether it is declared or not. Thus interface is considered as 100% pure abstract class . But since frim 1.8 and 1.9 JDK private , statuc and default methods was allowed. So we can ignore that its 100% abstract class .

Whenever we are implementing any interface method compulsory we should declare that mwthod as public .

interface Interef

{

public void m1();

public void m2();

}

class ServiceProvider implements Interef

{

void m1()

{

}

Here we will encounter 2 error(s).

1. it should always be declared as public as if we do not declared it as public we will get an error as weaker access as its access modifier is deafult. So compulsory it should be declared as public

2. Whenever we are implementing an interface for each and every method of the abstract method of the interface we should provide an implementation. But we didn't provide implementation of the method m2 .

There are 2 Solution to it (2) are : provide implementation for the method m2.

Another way is that by declaring class as abstract .

There are 2 Correct way to do it .

1. interface Interef

{

public void m1();

public void m2();

}

abstract class ServiceProvider implements Interef

{

void m1()

{

}

}

2. interface Interef

{

public void m1();

public void m2();

}

class ServiceProvider implements Interef

{

public void m1()

{

}

public void m2()

{

}

}

Another way by using child class

interface Interef

{

public void m1();

public void m2();

}

class ServiceProvider implements Interef

{

public void m1()

{

}

}

class SubServiceProvider extends ServiceProvider

{

public void m2()

{

}

}

So here we can either declare the child class as abstract or provide implementation for the method m2.

Data Hiding :

class Account

{

private double balance ;

public double getbalance ()

{

 // VALIDATION

if valid

return balance ;

else

You cannot acces

}

} by declaring variable as private we can implement data hiding . The advantage of data hiding is that it provides security to our data .

ABSTRACTION :

Hiding internal implementation and highlighting the set of services we are