Java means DURGA SOFT..

# CORE JAVA

# Material

Java™

India's No.1 Software Training Institute

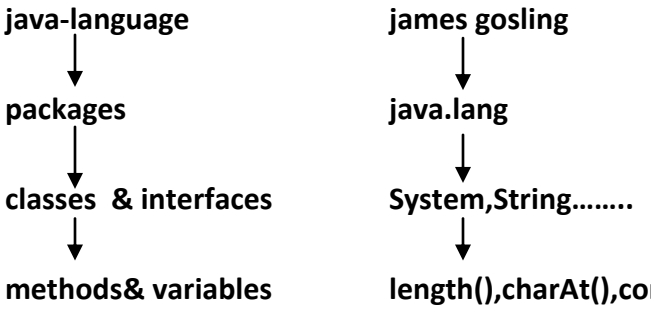
DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

# Packages

**java-language:-**

in java James Gosling is maintained predefined support in the form of packages and these packages contains classes & interfaces, and these classes and interfaces contains predefined methods & variables.




***java source code:-***

- java is a open source software we are able to download it free of cost and we are able to see the source code of the java.
- The source code location **C:\Program Files\Java\jdk1.7.0_75\src(zip file)** extract the zip file.
- Java contains 14 predefined packages but the default package in java if ***java.lang.*** package.

| | | | |
|---|---|---|---|
| **Java.lang** | **java.beans** | **java.text** | **java.sql** |
| **Java.io** | **java.net** | **java.nio** | **java.math** |
| **Java.util** | **java.applet** | **java.rmi** | |
| **Java.awt** | **java.times** | **java.security** | |

***Note : The default package in java is java.lang package.***

***Note : package is nothing but physical folder structure.***

***Types of packages:-***

There are two types of packages in java

1) Predefined packages.
2) User defined packages.

***Predefined packages:***

The predefined packages are introduced by James Gosling and these packages contains predefined classes & interfaces and these class & interfaces contains predefined variables and methods.

*Example:-java.lang, java.io ,java.util…..etc*

***User defined packages:-***

- The packages which are defined by user, and these packages contains user defined classes and interfaces.
- Declare the package by using ***package*** keyword.

  syntax : ***package package-name;***

example : ***package com.sravya;***

- Inside the source file it is possible to declare only one package statement and that statement must be first statement of the source file.

***Example-1:valid***
```
packagecom.sravya;
import java.io.*;
importjava.lang.*;
```

***Example-2:Invalid***
```
import java.io.*;
packagecom.sravya;
import java.io.*;
```

***Example-3:Invalid***
```
import java.io.*;
importjava.lang.*;
packagecom.sravya;
```

***Example-4:Invalid***
```
packagecom.sravya;
packagecom.tcs;
```

***some predefined package and it's classes & interfaces:-***

***Java.lang:-***The most commonly required classes and interfaces to write a sample program is encapsulated into a separate package is called java.lang package.

**java**
|------→**lang**
|--→String(class)
|--→StringBuffer(class)
|--→Object(class)
|--→Runnable(interface)
|--→Cloneable(interface)

Note:- the default package in the java programming is java.lang package.

***Java.io package:-***The classes which are used to perform the input output operations that are present in the java.io packages.

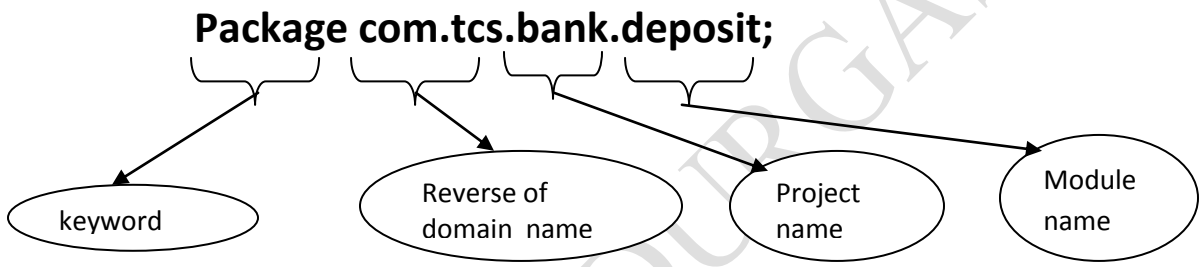
**java**
|------→**io**
|--→FileInputStream(class)
|--→FileOutputStream(class)
|--→FileReader(class)
|--→FileWriter(class)
|--→Serializable(interface)

***Java.net package:-***The classes which are required for connection establishment in the network that classes are present in the java.net package.

**java**
|------→**net**
|--→Socket(class)
|--→ServerSocket(class)
|--→URL(class)
|--→SocketOption(interface)

***Package name coding conventions :-(not mandatory but we have to fallow)***

1) The package name must reflect with organization domain name***(reverse of domain name).***
   **Domain name:- www.tcs.com**
   **Package name:- Package com.tcs;**
2) Package name must reflect with project name.
   **Project name :- bank**
   **package :- Package com.tcs.bank;**
3) The project name must reflect with project module name.
   **Domain name:- www.tcs.com**
   **Project name:- bank**
   **Module name:- deposit**
   **package name:- Package com.tcs.bank.deposit;**




## ***Advantages of packages:-***

**company name : tcs**
**project name : bank**

**module-1***Deposit*

```
com
 |-->tcs
        |-->bank
              |-->deposit
                    |--->.class files
```

**module-2***withdraw*

```
com
 |-->tcs
        |-->bank
              |-->withdraw
                    |--->.class files
```

**Module-3***moneytranfer*

```
com
 |-->tcs
        |-->bank
              |-->moneytranfer
                    |--->.class files
```

**module-4***accountinfo*

```
com
 |-->tcs
        |-->bank
              |-->accountinfo
                    |--->.class files
```

1) ***It improves parallel development of the project.***
2) ***Project maintenance will become easy.***
3) ***It improves sharability of the project.***
4) ***It improves readability.***
5) ***It improves understandability.***

***Note :- In real time the project is divided into number of modules that each and every module is nothing but package statement.***

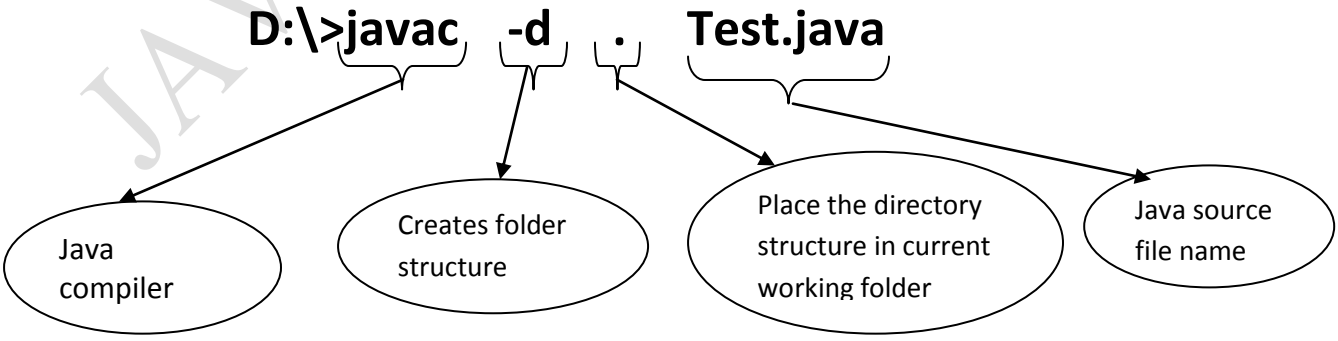

***Example-1:-***

***Step-1: write the application with package statement.***

```
packagecom.sravya.java.corejava;
class Test
{       public static void main(String[] args)
        {       System.out.println("package first example");
        }
}
class A
{}
class B
{}
interface It
{}
```

***Step-2: compilation process***

If the source file contains the package statement then compile that application by using fallowing command.



**Step-3:- folder Structure.**

```
com
 |-->sravya
        |-->java
        |-->corejava
                |-->Test.class
                |-->A.class
        |-->B.class
                |-->It.class
```

***Step-4:-execution process.***

Execute the .class file by using fully qualified name(***class name with complete package structure***)

***javacom.sravya.java.corejava.Test***

*output :* ***package first example***


***Example-2:-***

***Error-1 :-***

- *If it is a predefined package or user defined package Whenever we are using other package classes then must import that package by using import statement.*
- *If the application required two classes (System,String) then We are able to import the classes in two ways*
  - ***Importing all classes.***
    *Import java.lang.*;*
  - ***Importing application required classes***
    *Import java.lang.System;*
    *Import java.lang.String;*

*In above two approaches second approach is recommended because it is importing application required classes.*

***Error-2:-***

- *Whenever we are using other package classes then that classes must be public otherwise compiler generate error message.*

***Error:class is not public we are unable to access outside package.***

***Public modifier:-***

- ✓ *Public modifier is applicable for variables,methods,classes.*
- ✓ *All packages are able to access public members.*

***Default modifier:-***

- ➢ It is applicable for variables,methods,classes.
- ➢ We are able to access default members only within the package and it is not possible to access outside package .
- ➢ Default access is also known as package level access.
- ➢ The default modifier in java is default.

***Error-3:-***

- ❖ *Whenever we are using other package class member that members also must be public.*

***Note : When we declare class as public the corresponding members are not public, if we want access public class members that members also must be public.***

***File-1: Sravya.java***

```
package com.sravya.states.info;
public class Sravya
{       public void ts(){System.out.println("jai telengana");}
        public void ap(){System.out.println("jai andhra");}
        public void others(){System.out.println("jai jai others");}
}
```



***File-2: Tcs.java***

```
packagecom.tcs.states.requiredinfo;
import com.sravya.states.info.*;
class  Tcs
{       public static void main(String[] args)
        {       Sravya s = new Sravya();
```

```
            s.ts();         s.ap();         s.others();
        }
}
```

*E:\>javac -d . Sravya.java* ***compilation of Sravya***
*E:\>javac -d . Tcs.java* ***compilation of Tcs***
*E:\>java com.tcs.states.requiredinfo.Tcs* ***execution of Tcs***

```
jaitelengana
jaiandhra
jaijai others
```

***Example:-***

***Private modifier:-***

- ✓ privatemodifier applicable for methods and variables.
- ✓ We are able to access private members only within the class and it is not possible to access even in child classes.

```
class Parent
{       privateint a=10;
};
class Child extends Parent
{       void m1()
        {       System.out.println(a);  //a variables is private Child class unable to access
        }
        public static void main(String[] arghs)
        {       Child c = new Child();
                c.m1();
        }
};
```

***error: a has private access in Parent***

***Note :- the most accessable modifier in java Is public & most restricted modifier in java is private.***

***Example :-***

***Protected modifier:-***

- ✓ Protected modifier is applicable for variables,methods.
- ✓ We are able access protected members with in the package and it is possible to access outside packages also but only in child classes.
- ✓ But in outside package we can access protected members only by using child reference. If wetry to use parent reference we will get compile time error.

***A.java:-***

```
package   app1;
public class A
{       protectedint fee=1000;
};
```

***B.java:-***

```
package app2;
import app1.*;
public class B extends A
```

```
{       public static void main(String[] args)
        {       B    b = new B( );
                System.out.println(b.fee);
        }
};
```

| ***Parent-class method*** | ***child-class method*** | |
|---|---|---|
| ***Default*** | *default* | *-->valid (same level)* |
| | *protected , public* | *--> valid (increasing permission )* |
| | *Private* | *--> invalid (decreasing permission)* |
| ***Public*** | *public* | *--> valid (same level)* |
| | *Default,private,protected* | *-->invalid(decreasing permission)* |
| ***Protected*** | *protected* | *--->valid (same level)* |
| | *Public* | *--->valid (increasing permission)* |
| | *Default,private* | *---->invalid (decreasing permission)* |



***Case 1:- same level*** ***[default-default]***

```
abstract class Test
{       abstract void m1();          // default modifier
};
class Test1 extends Test
{       void m1(){System.out.println("m1 method");}      //default modifier
}
```

***Case 2:- increasing permission*** ***[protected-public]***

```
abstract class Test
{       protected abstract void m1();     // protected modifier access
};
class Test1 extends Test
{       public void m1(){System.out.println("m1 method");}       //public modifier access
};
```

***Case3 :- decreasing permission*** ***[public-protected]***

```
abstract class Test
{       public abstract void m1();          // public  modifier
};
```

```
class Test1 extends Test
{       protected void m1(){System.out.println("m1 method");}  //protected  modifier
};
```

***Summary of variables:-***

| ***modifier*** | ***Private*** | ***no-modifier*** | ***protected*** | ***public*** |
|---|---|---|---|---|
| ***Same class*** | *yes* | *yes* | *yes* | *yes* |
| ***Same package sub class*** | *no* | *yes* | *yes* | *yes* |
| ***Same package non sub class*** | *no* | *yes* | *yes* | *yes* |
| ***Different package sub class*** | *no* | *no* | *yes* | *yes* |
| ***Different package non sub class*** | *no* | *no* | *no* | *yes* |

**Example :-**

***Test.java:-***

```
package app1;
public class Test
{       public void m1(){System.out.println("app1.Test class m1()");}
}
```

***A.java:-***

```
package app1.corejava;
public class A
{       public void m1(){System.out.println("app1.corejava.A class m1()");}
}
```

***Ratan.java:-***

```
import app1.Test;
import app1.corejava.A;
classRatan
{       public static void main(String[] args)
        {       Test t = new Test();
                t.m1();
                A a =new A();
                a.m1();
        }
}
```


***Example :-***

***Test.java:-***

```
package app1;
public class Test
{       public void m1(){System.out.println("app1.Test class m1()");}
}
```

**X.java:-**

```
package app1.corejava;
public class X
{       public void m1(){System.out.println("app1.core.X class m1()");}
}
```

***Y.java:-***

```
package app1.corejava.advjava;
public class Y
{       public void m1(){System.out.println("app1.corejava.advjava.Y class m1()");}
}
```

***Z.java:-***

```
Package app1.corejava.advjava.structs;
public class Z
{       public void m1(){System.out.println("app1.corejava.advjava.structs.Z class m1()");}
}
```

***Ratan.java:-***

```
import app1.Test;
import app1.corejava.X;
import app1.corejava.advjava.Y;
import app1.corejava.advjava.structs.Z;
classRatan
{       public static void main(String[] args)
        {       Test t = new Test();  t.m1();
                X x = new X();  x.m1();
                Y y = new Y();   y.m1();
                Z z = new Z(); z.m1();
        }
};
```

*Note :- applicable modifiers on constructors*

1) *Public*
2) *Private*
3) *Default (if we are not providing modifiers)*
4) *Protected*

***Private constructors:-***

```
class Parent
{       private Parent(){}//private constructor
}
class Child extends Parent
{       Child()
        {super();} //we are calling parent class private constructor it is not possible
};
D:\>javac Test.java
Test.java:6: Parent() has private access in Parent
```

***Static import:-***

1. *This concept is introduced in 1.5 version.*
2. *if we are using the static import it is possible to call static variables and static methods of a particular class directly to the application without using class name.*
   a. ***import static java.lang.System.*;***

   *The above line is used to call all the static members of System class directly into application without using class name.*

***Ex:-without static mport***

```
importjava.lang.*;
class Test
{       public static void main(String[] args)
        {
        System.out.println("Hello World!");
        System.out.println("Hello World!");
        System.out.println("Hello World!");
        }
}
```

***Ex :- with static import***

```
import static java.lang.System.*;
class Test
{       public static void main(String[] args)
        {
        out.println("ratan world");
        out.println("ratan world");
        out.println("ratan world");
        }
};
```

***Example:-***

```
packagecom.dss.java.corejava;
public class Sravya
{       public static  int fee=1000;
        public static void course()
        {       System.out.println("core java");
        }
        public static void duration()
        {       System.out.println("1-month");
        }
        public static void trainer()
        {       System.out.println("ratan");
        }
};
```

***without static import***

```
packagecom.tcs.course.coursedetails;
importcom.dss.java.corejava.*;
classTcs
{       public static void main(String[] args)
        {       System.out.println(Sravya.fee);
                Sravya.course();
                Sravya.duration();
                Sravya.trainer();
        }
}
```

***with static import***

```
packagecom.tcs.course.coursedetails;
import static com.dss.java.corejava.Sravya.*;
classTcs
{       public static void main(String[] args)
        {       System.out.println(fee);
                course();
                duration();
                trainer();
        }
}
```


***Example :-***

- When you import main package we are able to access only package classes, it is not possible to access sub package classes, if we want sub package classes must import sub packages also.

***Ex:-***

***com***
***|-->sravya***
*|--->A.class*
*|--->B.class*
*|--->C.class*
*|--->**ratan***
*|--->D.class*

*In above example when we import **com.sravya.*** it is possible to access only three classes(A,B,C) but it is not possible to access sub package classes **(ratan package D class)** if we want sub package classes must import sub package(**import com.sravya.ratan.***).*

***File-1: A.java***

```
packagejav.corejava;
public class A
{       public void m1()
        {System.out.println("core java World!");
        }
}
```

***File-2: B.java:***

```
packagejav.corejava.advjava;
public class  B
{       public void m1()
        {System.out.println("Adv java World!");
        }
}
```

***File-3: C.java:-***

```
packagejav.corejava.advjava.structs;
public class  C
{       public void m1()
        {System.out.println("Structs World!");
        }
```

***Package structure:-***

```
jav
|-->corejava
        |--->A.class
```

```
jav
|-->corejava
        |--->A.class
        |--->advjava
                |--->B.class
```

```
}
```

***File-4:- MainTest.java***

***Package structure :-***

```
jav
|-->corejava
```

**|--->A.class** **|--->structs**
**|--->advjava** **|--->C.class**
**|--->B.class**

```
importjav.corejava.A;
importjav.corejava.advjava.B;
importjav.corejava.advjava.structs.C;
classMainTest
{       public static void main(String[] args)
        {       A a = new A(); a.m1();
                B b = new B(); b.m1();
                C c = new C(); c.m1();
        }
}
```


***Example :-*** ***in java it is not possible to use predefined package names as a user defined packages.***

```
packagejava.lang;
class Test
{       public static void main(String[] args)
        {       System.out.println("Ratan World!");
        }
}
class A
{       };
```

***D:\DP>javac -d . Test.java***
***D:\DP>java java.lang.Test***
***Exception in thread "main" java.lang.SecurityException: Prohibited package name: java.lang***

***Applicable modifiers on constructors:-***

1) *Public*
2) *Private*

3) *Protected*
4) *default*

***we are achieving singleton class creation by using private constructors in java:-***

- *when we declare constructor with private modifier we can't create object outside of the class.*
- *Singleton class allows to create only one object of particular class and we are achieving singleton class creation by using private constructors.*
- *In some scenarios it is appropriate to have exactly one instance of class like,*
  - *Window manager*
  - *File systems*
  - *Project manager*
  - *Admin*

  *These type of objects are called singleton objects.*

***file-1:-***

```
package com.dss.st;
class Test
{       public static Test t=null;
        private Test(){}
        public static Test getInstance()
        {       if (t==null)
                {       t = new Test();
                }
                return t;
        }
        public void disp()
        {       System.out.println("this is ratan singleton class");
        }
};
```

***File-2:-***

```
Package com.dss;
Import com.dss.st.Test;
class Test1
{       public static void main(String[] args)
        {       //Test t = new Test(); compilation error  Test() has private access in Test
                Test t1 = Test.getInstance();
                Test t2 = Test.getInstance();
                System.out.println(t1.hashCode());//31168322
                System.out.println(t2.hashCode());//31168322
        }
};
```


**Source file Declaration rules:-**

The source file contains the fallowing elements

1) Package declaration---→optional-----→at most one package(0 or 1)--→1st statement
2) Import declaration-----→optional-----→any number of imports--------→2nd statement
3) Class declaration--------→optional-----→any number of classes---------→3rd statement
4) Interface declaration---→optional----→any number of interfaces-----→3rd statement
5) Comments declaration-→optional----→any number of comments----→3rd statement

a. The package must be the first statement of the source file and it is possible to declare at most one package within the source file .
b. The import session must be in between the package and class statement. And it is possible to declare any number of import statements within the source file.
c. The class session is must be after package and import statement and it is possible to declare any number of class within the source file.
    i. It is possible to declare at most one public class.
    ii. It is possible to declare any number of non-public classes.
d. The package and import statements are applicable for all the classes present in the source file.
e. It is possible to declare comments at beginning and ending of any line of declaration it is possible to declare any number of comments within the source file.

**Preparation of user defined API (application programming interface document):-**

1. API document nothing but user guide.
2. Whenever we are buying any product the manufacturing people provides one document called user guide. By using user guide we are able to use the product properly.
3. James gosling is developed java product whenever james gosling is delivered the project that person is providing one user document called API(application programming interface) document it contains the information about how to use the product.
4. To prepare user defined apidocument for the user defined projects we must provide the description by using documentation comments that information is visible in API document.


LEARN FROM EXPERTS ...
COMPLETE JAVA
CORE JAVA, ADV. JAVA, ORACLE, STRUTS, HIBERNATE, SPRING, WEB SERVICES,...
COMPLETE .NET
C#.NET, ASP.NET, SQL SERVER, MVC 5 & WCF
TESTING TOOLS
MANUAL + SELENIUM
ORACLE
D2K
MSBI
SHARE POINT
HADOOP
ANDROID
C, C++, DS, UNIX
CRT & APTITUDE TRAINING
AN ISO 9001:2008 CERTIFIED
DURGA
Software Solutions®
# 202, 2nd Floor, HUDA Maitrivanam,
Ameerpet, Hyd. Ph: 040-64512786,
9246212143, 8096969696
www.durgasoft.com