Java means DURGA SOFT..

CORE JAVA

Material

Java™

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Packages

- Package is a collection of related classes and interfaces as a single unit.

--diagram--(Packages.png)[package1]

- Package is a folder contains .class files representing related classes and interfaces.

--diagram--(Packages.png)[package2]

In Java applications,packages will provide the following advantages.

1.Modularity:

- In enterprise application development,we will prepare modules by declaring packages only,so that,packages will improve modularity.

2.Abstraction:

- If we declare classes and interfaces in a package then that classes and interfaces are not visible in out side,so that,packages will provide abstraction.

3.Security:

- In Java applications,packages will provide abstraction and encapsulation so that packages will provide security.

  Security=Encapsulation+Abstraction

4.Sharability:

- In Java applications,if we declare one package then we can share that package to any no.of modules at a time.

5.Reusability:

- In Java applications,if we declare a package one time then that we can reuse any no. of times at any module.

  --diagram--(Pacakages1.png)

  There are two types of packages in Java.

  1.Predefined packages

  2.User defined packages

1.Predefined Packages:

- These are the packages defined by Java programming language along with Java software.

Ex:

1.java.lang:

- It is a default package in every java file,no need to import this package.
- java.lang package contains the basic classes and interfaces which are required for basic Java applications.
- java.lang.package contains the classes and interfaces like

String,StringBuffer,StringBuilder

Thread,Runnable

Exception,ArithmeticException,NullPointerException.....

Byte,Short,Integer,Long....

Object,Cloneable,Comparable.....

2.java.io:

- This package contains all the predefined classes and interfaces which are required to perform,IO Operations in Java.

  This package contains the predefined classes and interfaces like

  InputStream,ByteArrayInputStream,FileInputStream....

  OutputStream,ByteArrayOutputStream,FileOutputStream...

  Reader,FileReader,InputStreamReader,BufferedReader.....

  Writer,FileWriter,BufferedWriter.....

  Serializable,Externalizable.....

3.java.util:

- This package will provide all the predefined classes and interfaces to represent data structures in java applications.

  This package will provide the predefined classes and interfaces like
  List,ArrayList,Vector,Stack,LinkedList

Set,HashSet,LinkedHashSet,TreeSet,NavigableSet,SortedSet,

Map,HashMap,

LinkedHashMap,TreeMap,NavigableMap....

4.java.awt:

- This package will provide all the predefined classes and interfaces to design GUI applications.

  This package contains the classes and interfaces like

  Component,Label,TextFiled,TextArea,Button,Check box,List..

  FlowLayout,BorderLayout,GridLayout,GridBagLayout,CardLayout....

5.javax.swing:

- This package will provide predefined classes and interfaces to design GUI applications.

  javax.swing package is able to provide no.of GUI components which are light weight GUI components, platform independent GUI components,able to consume less memory and less execution time...

  when compared with AWT provided GUI components.

  This package contains the predefined classes and interfaces like

  JComponent,JTextField,JLabel,JButton,JPasswordField,JList,JCheckBox,JRadioButton,JColorChooser, JFileChooser.....

6.java.net:

- This package will provide all the predefined classes and interfaces to prepare Distributed applications.

  This package will provide the predefined classes and interfaces like Socket, ServerSocket, URL, URLConnection, INetAddress

7.java.sql:

- This package will provide predefined classes and interfaces to prepare JDBC applications.

  This package will provide the predefined classes and interfaces like

  Driver

  DriverManager

  Connection

  Statement

  PreparedStatement

  CallableStatement

  ResultSet

2.User Defined packages:

- These are the packages defined by the developers as per their application requirements.

  To declare user defined packages in Java applications,we have to use the following syntax.

package package_Name;

where package declaration statement should be first statement in Java file after comment section.

NOTE: It is not possible to provide more than one package declaration statement with in a single java file.

Where package name should be unique,it should not be duplicated and it should not be shareable.

To provide package names,Java has given a convention like to include our company domain name in reverse.

EX:

package com.durgasoft.axis.loan;

com.durgasoft-->Company Domain name in reverse

axis-->Client name/project name

loan-->module name

- If we want to use a particular package content in present java file then we have to import the respective package in present java file.
- To import a package in java file,we have to use the following syntax.

  import package_Name.*;

- It will import all the classes and interfaces from the specified package.

  Ex:import java.util.*;

  import package_Name.Member_Name;

- It will import only the specified member from the specified package.

  Ex:import java.util.ArrayList;

Example:

D:\abc

Employee.java

C:\abc

com

|----durgasoft

|------core

|----Employee.class

D:\java7

MainApp.java

D:\abc\Employee.java:

```
package com.durgasoft.core;(D:\abc\Employee.java)
public class Employee{
String eid;
String ename;
String eaddr;
```

```
public Employee(String eid,String ename,String eaddr){
this.eid=eid;
this.ename=ename;
this.eaddr=eaddr;
}
public void displayEmployee(){
System.out.println("Employee Details");
System.out.println("-----------------");
System.out.println("Employee Id :"+eid);
System.out.println("Employee Name :"+ename);
System.out.println("Employee Address :"+eaddr);
}
}
```


compilation:D:\abc>javac -d c:\abc Employee.java

D:\java7\MainApp.java:

```
import com.durgasoft.core.*;
public class MainApp{
public static void main(String args[]){
```

```
Employee emp=new Employee("E-111","Durga","Hyd");
emp.displayEmployee();
}
}
```

compilation:D:\java7>set classpath=C:\abc;.;

D:\java7>javac MainApp.java

D:\java7>java MainApp

jar files in Java:

As part of Java applications development,after implementing the java application it is not suggestable to send the application in the form of .class files individually to the clients.

Always,it is suggestable to manage the generated .class files in the form of jar[Java Archive] files and it is suggestable to send that jar files to the clients or to the customers.

To prepare jar file in java applications,we have to use the following command on command prompt.

jar -cvf file_name.jar *.*

CVF --->Create Verboz file

. --->all files and folders

To extract jar file,we have to use the following command.

jar -xvf file_Name.jar

If we want to use the content of jar file in our java application then we have to set "classpath" environment variable to jar file.

D:\abc

Employee.java

Student.java

C:\abc

com

|-----durgasoft

|-------emp

| |----Employee.class

|-------std

| |----Student.class

C:\xyx

durga.jar

D:\java7

MainApp.java

MainApp.class

D:\abc\Employee.java:

```
package com.durgasoft.emp;
public class Employee{
String eid;
String ename;
String eaddr;
public Employee(String eid,String ename,String eaddr){
this.eid=eid;
this.ename=ename;
this.eaddr=eaddr;
}
public void displayEmployee(){
System.out.println("Employee Details");
System.out.println("-----------------");
System.out.println("Employee Id :"+eid);
System.out.println("Employee Name :"+ename);
System.out.println("Employee Address :"+eaddr);
}
}
```

D:\abc\Student.java:

```
package com.durgasoft.std;
public class Student{
String sid;
String sname;
String saddr;
public Student(String sid,String sname,String saddr){
this.sid=sid;
this.sname=sname;
this.saddr=saddr;
}
public void displayStudent(){
System.out.println("Student Details");
System.out.println("----------------");
System.out.println("Student Id  :"+sid);
System.out.println("Student Name :"+sname);
```

```
System.out.println("Student Address :"+saddr);
}
}
```

Compilation Process:

```
D:\abc>javac   -d C:\abc  *.java
C:\abc>jar -cvf durga.jar *.*
```

Copy the generated durga.jar from C:\abc and past at C:\xyz location for testing purpose.

D:\java7\MainApp.java

```
import com.durgasoft.emp.*;
import com.durgasoft.std.*;
public class MainApp{
public static void main(String args[]){
Employee emp=new Employee("E-111","Durga","Hyd");
emp.displayEmployee();
System.out.println();
Student std=new Student("S-111","Anil","Sec");
std.displayStudent();
}
```

```
}
```

Compilation Process:

```
D:\java7>set classpath=C:\xyz\durga.jar;.;

D:\java7>javac MainApp.java

D:\java7>java MainApp
```

In Java applications,when we create a Jar file,jar command will provide MANIFEST.MF file under META-INF folder inside jar file along with our current location content.Here MANIFEST.MF file is able to store the data about jar file in the form of key-value pairs.

Ex:

```
D:\abc> jar -cvf durga.jar *.*
```

Diagram(Packages2.png)

Steps to prepare Executable jar file and batch file:

Executable Jar file is a jar file which contains main class directly.

1)Prepare java application as per the requirement.

D:\abc\MainApp.java

```
import java.awt.*;

class LogoFrame extends Frame{

LogoFrame(){
```

```
this.setVisible(true);
this.setSize(700,500);
this.setTitle("Logo Frame");
this.setBackground(Color.green);
}

public void paint(Graphics g){
Font f=new Font("airal",Font.BOLD+Font.ITALIC,35);
g.setFont(f);
this.setForeground(Color.red);
g.drawString("DURGA SOFTWARE SOULTIONS",50,100);
}
}
class MainApp{
public static void main(String args[]){
LogoFrame lf=new LogoFrame();
}
}
```

2)Prepare a text file with "Main-Class" attribute:

D:\abc\abc.txt

Main-Class:MainApp

The main intention to prepare this text file is to keep Main-Class attribute data in the generated MANIFEST.MF file.

3)Compile Java file and prepare jar file with user provided data in MANIFEST.MF file:

Syntax:

jar -cvfm file_Name.jar file_name.txt *.*

Ex:

D:\abc>javac MainApp.java

D:\abc>jar -cvfm durga.jar abc.txt *.*

When we execute the above command on command prompt then "jar" command will prepare jar file with MainApp.class and MANIFEST.MF file with "Main-Class:MainApp" attribute data.

4.Execute jar file:

To execute jar file directly we have to use the following command.

java -jar file_name.jar

EX:

D:\abc>java -jar durga.jar

If we use the above command on command prompt then JVM will perform the following actions

a)JVM will search for durga.jar file at current location

b)If durga.jar file is available then JVM will goto META-INF folder and recognize MANIFEST.MF file.

c)JVM will get "Main-Class" attribute value from MANIFEST.MF file that is main class name that is "MainApp".

d)JVM will search for main class .class file inside durga.jar file

e)If MainApp.class file is available in durga.jar file then JVM will execute main class.

Diagram(Packages3.png)

If we want to execute the above application with batch file then we have to prepare batch file with the required "java" command,keep both batch file and jar file at the same location and double click on batch file.


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