PACKAGES

PACKAGES

Package is the collection of related classes and interfaces as a single unit.
Package is a folder contains .class files representing related classes and interfaces

In Java applications, packages are able to provide the following advantages.

- Modularity
- Abstraction
- Security
- Sharability
- Reusability

1) Modularity:

- Dividing a requirement set into no of pieces, providing solutions to the individual parts and combining all individual solutions to a single is called as "Modularization".
- In Java applications, Module is a folder contains .class files representing related classes and interfaces as a single unit.
- Due to the above module definition, we can conclude that Packages are able to improve modularity.

2) Abstraction:

If we declare classes and interfaces in a module or in a package then that classes and interfaces are not visible in outside of the package by default, so that, packages are able to improve Abstraction.

3) Security:

In Java applications, packages are able to provide Abstraction and Encapsulation, where both are able to improve Security.

4) Sharability:

In Java applications, if we declare packages one time then we are able to share that package content to any number of applications or modules at a time.

5) Reusability

In Java applications, if we declare a package one time then we are able to reuse any number of times within a single application or in more than one application.

There are 2 types of packages in Java .

- Predefined Packages
- User defined Packages

1) Predefined Packages:

These packages are provided by JAVA programming language along with Java software.

EX 1: java.lang:

- This package is default package in java applications, no need to import this package to the present java file, when we save java file with .java extension then automatically java.lang package is imported internally.
- This package is able to provide all fundamental classes and interfaces which are required to prepare basic java applications.
- java.lang package includes the following classes and interfaces to prepare java applications.

String, StringBuffer, StringBuilder,....
Object, System, Class,.....
Exception , ArithmeticException, NullPointerException,...
Thread, Runnable, Cloneable, Comparable.....
Integer, Byte, Short, Float, Long,.....

EX2: java.io:

This package is able to provide predefined classes and interfaces in order to perform Input and Output operations in Java.

This package includes the following classes and interfaces in java applications.

InputStream, ByteArrayInputStream, FileInputSTream,.....
OutputStream, ByteArrayOutputStream, FileOutputStream,.....
Reader, CharArrayReader, FileReader, ...
Writer , CharArrayWriter, FileWriter,.....
Serializable, Externalizable,.....

EX3: java.util:

This package is able to provide all predefined classes and interfaces which are representing data structures .

This package is able to provide classes and interfaces like below.

Collection, List, Set,Queue
ArrayList, Vector, Stack, LinkedList.
HashSet, LinkedHashSet, SortedSet, NavigableSet, TreeSet
Queue, PriorityQueue, BlockingQueue, LinkedBlockingQueue,.....
Map, HashMap, LinkedHashMap, IdentityHashMap, WeakHashMap,.....

EX 4: java.awt:

This package is able to provide predefined classes and interfaces representing all GUI components in order to prepare GUI applications.

This package has provided the following classes and interfaces to prepare GUI applications.

Component, Label, TextField, TextArea, Button, CheckBox, List, Choice, Frame,.....

EX 5: javax.swing:

This package is able to provide predefined library to prepare GUI applications.

Q) What are the differences between *AWT [java.awt]* and *SWING [javax.swing]?*

1. AWT provided GUI components are platform dependent GUI components.
SWING provided GUI components are Platform Independent GUI Components.

2. AWT provided GUI components are heavy weight GUI components.
SWING GUI components are light weight GUI components.

3. AWT provided GUI COmponents are basic GUI components.
EX: TextField, TextArea, Label, Button,....
SWING provided GUI components are most advanced GUI components.
EX: JColorChooser, JFileChooser, JTable, JTree,....

4. AWT provided GUI components are able to reduce application performance.
SWING provided GUI components are able to improve application performance.

javax.swing package has provided the following predefined classes and interfaces to design GUI applications.

JComponent, JTextField, JPasswordField, JButton, JCheckBox, JRadioButton, JFrame, JColorChooser, JFileChooser,.....

EX 6: java.net:

If we prepare any java application without using Client-Server Arch then that java application is called as Standalone Application.

If we prepare any java application on the basis of Client-Server Arch then that java application is called as Distributed application.

To prepare distributed applications, JAVA has provided the following distributed technologies.

- Socket Programming
- RMI [Remote Method Invocation]
- CORBA [Common Object Request Broker Arch/Agent]
- EJBs [Enterprise Java Beans]
- Web Services

java.net package has provided predefined library to prepare distributed applications by using Socket Programming.

Socket
ServerSocket
URL, URI, URN
URLConnection

EX 7: java.rmi:

java.rmi package is able to provide predefined library to prepare Distributed applications on the basis RMI .

Remote
RemoteException
Naming

EX 8: java.sql:

The process of interacting with database from java application is called as JDBC [Java Database Connectivity].

To prepare JDBC applications, java has provided predefined library in the form of java.sql package.

java.sql package has provided the following classes and interfaces to prepare JDBC applications.

Driver, DriverManager, Connection, Statement, PreparedStatement, CallableStatement, ResultSet, ResultSetMetaData, DatabaseMetaData, ...

java.Text -
it able to provide some predefined classes and interfaces to provide Internationalization I18N service in java application
eg -

NumberFormat
DateFormat

2) User Defined Packages:

These packages are defined by the developers as per their application requirements.
Syntax: package package_Name;

If we want to use package declaration statement in java applications then we have to use the following two conditions.

- Package Declaration Statement must be first statement.
- Package name must be unique, it must not be sharable and it must not be duplicated.

Q) Is It Possible To Provide More Than One Package Declaration Statement Within A Single Java File?

- No, it is not possible to provide more than one package declaration statement within a single java file, because, package declaration statement must be provided as first statement in java files.
- To provide package names in java applications, JAVA has provided a convention like to include our company domain name in reverse.

EX:
package com.durgasoft.icici.transactions.deposit;
com.durgasoft---> Company Domain name in reverse.
icici -----> client/project name
transactions---> Module name
deposit ------> sub module name

If we declare classes and interfaces in a package and if we want to use these classes and interfaces in the present java file then we have to import the respective package to the present java file.

To import packages to the present java file we have to use the following syntaxes.

import package_Name.*;
--> It able to import all the classes and interfaces of the specified package.

EX: import java.util.*;

import package_Name.member_Name;
It able to import only the specified member from the specified package.

EX: import java.util.ArrayList;

Note: In java files, we are able to provide at most one package declaration statement, but, we are able to provide any number of import statements.

Q) Is It Possible To Use Classes And Interfaces of A Particular Package without Importing The Respective Package?

Yes, it is possible to use classes and interfaces of a particular package without importing that package, but by using fully qualified names of the respective classes and interfaces.

Note: Specifying class name or interface name along with package name is called as "Fully Qualified Name"

EX: java.io.BufferedReader
java.util.ArrayList

A Java program with import statement:

```
import java.io.*;
---
BufferedReader br = new BufferedReader(new InputStreamReader( System.in));
```

A Java program without import statement:

```
java.io.BufferedReader br = new java.io.BufferedReader(new
java.io.InputStreamReader(System.in));
```

Application-1:

D:\abc
Employee.java

C:\xyz
com
|----durgasoft
|----core
|-----Employee.class

D:\javaapps\ packages
Test.java
Test.class

Employee.java

```
1)  package com.durgasoft.core;
2)  public class Employee
3)  {
4)     String eid;
5)     String ename;
6)     float esal;
7)     String eaddr;
8)     public Employee(String eid, String ename, float esal, String eaddr)
9)     {
10)       this.eid=eid;
11)       this.ename=ename;
12)       this.esal=esal;
13)       this.eaddr=eaddr;
14)    }
15)    public void getEmpDetails()
16)    {
17)       System.out.println("Employee Details");
18)       System.out.println("----------------------");
19)       System.out.println("Employee Id     :"+eid);
20)       System.out.println("Employee Name   :"+ename);
21)       System.out.println("Employee Salary :"+esal);
22)       System.out.println("Employee Address:"+eaddr);
23)    }
24) }
```

On Command Prompt:

```
D:\abc>javac  -d  C:\xyz  Employee.java
```

Test.java

```
1)  import com.durgasoft.core.*;
2)  public class Test
3)  {
4)     public static void main(String[] args)
5)     {
6)        Employee emp=new Employee("E-111", "Durga", 50000.0f, "Hyderabad");
7)        emp.getEmpDetails();
8)     }
9)  }
```

On Command Prompt:

```
D:\javaapps\packages>set classpath=C:\xyz;.;
D:\javaapps\packages>javac Test.java
D:\javaapps\packages>java Test
```

--- Employee Details----

classpath: To specify the location where package is existed.
import com.durgasoft.core.* : To specify in which package Employee class is existed.

JAR Files in JAVA:

In java applications development, moving .class files from one machine to another machine in the network, uploading .class files directly to the websites,.... are not suggestible. Always, it is suggestible to prepare JAR files for the .class files in order to move in the network and to upload applications in the websites.

To create JAR files we have to use the following Command.
jar -cvf File_Name.jar *.*

EX:
D:\javaapps>jar -cvf durgaapp.jar *.*

To extract JAR file we have to use the following command on command prompt.
jar -xvf file_Name.jar

EX:
D:\javaapps>jar -xvf durgaapp.jar

If we want to access packages or classes or interfaces from JAR file then we have to set classpath path environment variable to JAR file directly.

Application-2:

D:\abc
Account.java

C:\xyz
com
|---durgasoft
|----icici
|-----accounts
|------Account.class

C:\xyz
durga_icici.jar

Move durga_icici.jar from C:\xyz location to D:\xyz location

D:\xyz
durga_icici.jar

D:\javaapps\packages
Test.java
Test.class

Account.java

```
1)  package com.durgasoft.icici.accounts;
2)  public class Account
3)  {
4)     String accNo;
5)     String accName;
6)     String accType;
7)     String accBranch;
8)     String accBank;
9)     public Account(String accNo, String accName, String accType, String accBranch,
       String accBank)
10)    {
11)       this.accNo=accNo;
12)       this.accName=accName;
13)       this.accType=accType;
14)       this.accBranch=accBranch;
15)       this.accBank=accBank;
16)    }
17)    public void getAccountDetails()
18)    {
19)       System.out.println("Account Details");
20)       System.out.println("--------------------");
21)       System.out.println("Account Number   :"+accNo);
22)       System.out.println("Account Name     :"+accName);
23)       System.out.println("Account Type     :"+accType);
24)       System.out.println("Account Branch   :"+accBranch);
25)       System.out.println("Account Bank     :"+accBank);
26)    }
27) }
```

On Command Priompt:
D:\abc>javac -d C:\xyz Account.java
C:\xyz>jar -cvf durga_icici.jar *.*

Move durga_icici.jar from C:\xyz location to D:\xyz location.

Test.java

```
1)  import com.durgasoft.icici.accounts.*;
2)  public class Test
3)  {
4)     public static void main(String[] args)
5)     {
6)        Account acc=new Account("abc123", "Durga", "Savings", "S R Nagar", "ICICI");
7)        acc.getAccountDetails();
8)     }
9)  }
```

On Command Prompt:

```
D:\javaapps\packages>set classpath=D:\xyz\durga_icici.jar;.;
D:\javaapps\packages>javac Test.java
D:\javaapps\packages>java Test

---- Account Details-----
```

MANIFEST File in JAR:

MANIFEST.MF file is created by "jar" command at the time of creating JAR file under "META-INF" folder and it will provide metadata about the jar file in the form of Key-Value pairs.

Executable Jar Files:

If we prepare any JAR file with main class and main() method then that jar file is called as "Executable Jar" file.

To prepare Executable JAR file we have to use the following steps.

- **Prepare Java application as per the requirement and Compile Java application:**

EX: D:\javaapps\packages\ Test.java

```
1)  import java.awt.*;
2)  class LogoFrame extends Frame
3)  {
4)     LogoFrame()
5)     {
6)        this.setVisible(true);
7)        this.setSize(900,300);
8)        this.setBackground(Color.green);
9)        this.setTitle("Logo Frame");
10)    }
11)    public void paint(Graphics g)
```

```
12)   {
13)       Font f=new Font("arial", Font.BOLD, 40);
14)       g.setFont(f);
15)       this.setForeground(Color.red);
16)       String logo="DURGA SOFTWARE SOLUTIONS";
17)       g.drawString(logo, 100,150);
18)   }
19) }
20) class Test
21) {
22)    public static void main(String[] args)
23)    {
24)       LogoFrame lf=new LogoFrame();
25)    }
26) }
```

On Command Prompt D:\javaapps\packages>javac Test.java

- Prepare a text file with "Main-Class" attribute with Main class name:

D:\javaapps\packages\ abc.txt
Main-Class: Test

Note: The main intention to specify "Main-Class" attribute in text file is to send Main-Class attribute information to MANIFEST.MF file.

- Create JAR file with the MANIFEST.MF file, it must include

Main-Class attributes which we specified in text file.
On Command Prompt D:\javaapps\packages>jar -cvfm durgaapp.jar abc.txt *.*

- Execute JAR File

On Command prompt D:\javaapps\packages>java -jar durgaapp.jar

Internal Flow:

- JVM will recognize -jar option and JVM will take jar file name from command prompt.
- JVM will search for jar file at current location, if it is available then JVM will go to MANIFEST.MF file which is available META-INF folder.
- In MANIFEST.MF file, JVM will search for Main-Class attribute, if it is available then JVM will take its value that is Main Class name.
- After getting Main Class name from MANIFEST.MF file JVM will execute Main Class and generate output.

If we want to execute the above application through batch file, first, we have to prepare batch file with the required execution command then double click on that batch.

durga.bat [Take Text file and save with .bat extension]

```
java  -jar  durgaapp.jar
```

NOTE: We must save both jar file and bat file at the same location.