Packages

Packages:

The packages are the age old style of writing java programs in a sequential pattern to run and execute. In order to learn any programming language, we have to learn concepts, syntax and its library. If we want to learn c language then we want learn the concepts of “c” along with its syntax’s and library. A library is a collection of pre-defined header files. A header file is a collection of pre-defined functions. The pre-defined functions are nothing but, that are already registered functions to run the classes directly by writing the program with the class under a package.

For example when we write a class we

- Package CALENDAR
- Package SYSTEM
- Package MACHINE_CODE
- UNCHECKED_DEALLOCATION
- UNCHECKED_CONVERSION
- Package SEQUENTIAL_IO
- Package DIRECT_IO
- Package TEXT_IO

- Package IO_EXCEPTIONS
- Package LOW_LEVEL_IO

In c based software development, if we want to give any function as a common function for many “c" programmers, then we place the common functions into the header file. Otherwise it is not possible to use those common functions. Hence every common feature must belongs to header file. A header file in c language is common in java programming language too.

Similarly learning java is nothing but to learn its concepts: oops (i.e. Object Oriented programming software/language), syntax and API (Application programming Interface) to call the name of the library.

Define of API?

An API is collection of package. An API, or Application Programming Interfaces defines the way software's communicate each other over the Internet. APIs are increasingly the way in which companies exchange data, content, and digital resources, both internally, externally with partners, and openly with the public. An API allows anyone to open up data and other digital resources, for access by public developers, businesses, or even between departments and locations within a company.

What is package?

A package is a collection of classes, interfaces and sub packages. A sub package in terms contains collection of classes, interface and sub-sub packages etc.

The purpose of packages concept is to place common classes and interfaces. In other words, in java software development if we want to give any

class (or) interface to many number of java programmers, then keep common classes and interfaces into the package. Hence, all the classes and interfaces of a particular package (or) common for every java programmer.

Advantages of packages:

If we develop any java application with the concept of packages, then such applications will get following advantages.

1. Application development time is less.

2. Application memory space is less.

3. Application execution time is less.

4. Application performance is enhanced.

5. Application redundancy of code is reduced so that we can have consistent results and less storage cost.

6. We will be able to get the slogan of java (WORA=Write Once Read Anywhere)

Difference between inheritance and package?

Inheritance concept always make us to understand how to reuse the features with in the program between class, interface to interface and interface to class but not possible to access across the programs.

Packages concept makes us to understand how to reuse the features of both within and across the programs between class to class, interface to interface and interface to class.

Types of Packages

In java, we have three types of packages they are

1. Pre-defined/Built-in packages
2. User/Programmer/Secondary defined packages
3. Third party packages

Let us briefly learn about the packages:

Pre-defined packages:

As we discussed about the term “pre-defined”, in which packages are developed by sun micro system developers and supplied as part of Java software and are always used for dealing with Universal Requirements. These packages are useful to class to run the program without any code errors. We can check these pre-defined packages with code in the library files of JDK (java development kit) folder. We can view packages in the src, which will be seen as zip folder in the JDK. To view, open the src and see the folder where every package is written for pre-defined packages.

Some of the examples of pre-defined packages are:

- import java.io.*; // imports all the classes and interfaces
- import java.awt.evnet.*; // imports all the classes and interfaces
- import java.net.Socket; // imports only Socket class
- import java.awt.event.WindowEvent; // imports only WindowEvent class
- import java.sql;

- import java.applet;
- import java.awt;

User-defined package:

These user-defined packages are developed by java programmers and supplied as a part of their project to deal with common requirements. In most of the MNC Company the projects are ran with their common packages. It is the one of the most and best platform programming software to run the require projects. These user-defined packages are given by experience people in the company to the new trainers.

Third party:

Third party packages are developed by third party software vendors and supplied as a part of third party software and they are used for communicating with third party software products.

Types of Pre-defined packages:

Pre-defined packages are classified into three types. They are

a) J2SE package:

These packages are used for developing client side applications such as standalone, desktop and two-tier applications. These packages are the old concept which starts from the core java. JSE package can be abbreviated as (java standalone edition).

Example of J2SE/JSE program:

In your favorite editor, create a file called HelloIndia.java with the following contents:

```
Class HelloIndia
{
    Public Static Void Main (Strings args [])
    {
        System.Out.Println (“Hello India”)
    }
    }
```

From a command line, the command to compile this program is as follows:

javac HelloIndia.java

For this to work, the javac must be in your shell’s path, or you must explicitly specify the path to the program (such as c:\j2se\bin\javac HelloIndia.java). If the compilation is successful, javac will quietly end and return you to a command prompt. If you look in the directory, there will be HelloIndia.class file. This .class file is the compiled generated program of your compiled program.

To run the program, simply use the following Java command:

java HelloIndia

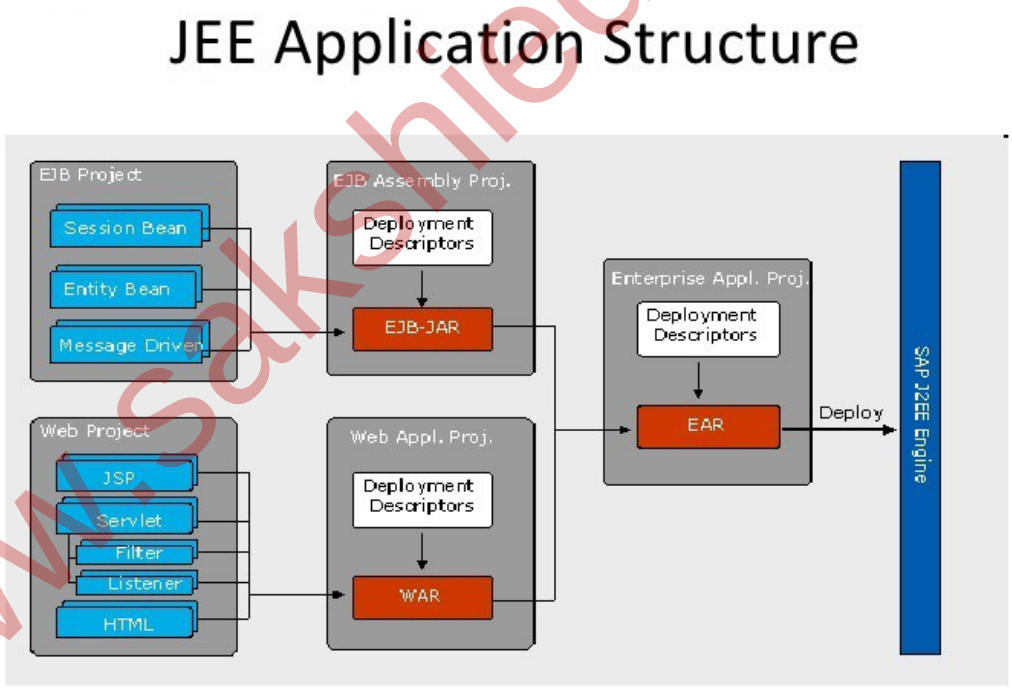
Ans: Hello Inida

b) J2EE package:

These packages are to be used for developing server side applications such as a web applications, distributed applications & enterprise applications. This package is the most important and running edition in most of the real-time industries. Most of the MNC's companies run the programs with JEE packages. J2EE/JEE can be abbreviated as (Java enterprise edition).

Example for J2EE/JEE program.

To create a JEE Program install the jdk with IDE (Environment project). These JEE is mainly is used to connect the local drivers and run the program. We can connect the Oracle database to Java Database (odbc.jdbc) by using these JEE application package.



Javax.servlet package can connect with .XML web app to run the application.

```
Javax.servlet.io;
{
        *//////////////////////
        //////////*
}
```

c) J2ME Package:

These packages are used for developing wireless or mobile applications. Java plays a main role in developing the mobile based applications. These are same as the web based applications but the difference can be seen only in the mobile view. For example applications like banking, recharges & social networking apps. These particular applications are used to develop with the JME (Java Mobile/Micro Edition).

Example for J2ME/JME

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
{
        public class HelloMIDlet extends MIDlet
        *//////////////////////////////
        ////////////////////*
        }
```

List of pre-defined packages:

In JSE/J2SE we have 8 essential languages which are useful for every class to run the program.

1) Java.lang.*:

This package is used for achieving language functionalities (or) basic services of the language.

a) Accepting command line arguments

b) Displaying the result of the console

c) Providing garbage collection

d) Data conversion techniques

e) Development of web based applications.

For every java program, this package will be imported default. Hence this package is known as the default package.

Program on Java.lang.*;

```
package java.lang;

public

class AbstractMethodError extends IncompatibleClassChangeError {

    private static final long serialVersionUID = -1654391082989018462L;

    public AbstractMethodError()

{
```

```
        super();
        {
    public AbstractMethodError(String s) {
        super(s);
        }
}
```

2) Java.awt.*;

This is also called as abstract windowing tool kit. The aim of the package is to design GUI application which will be look & feel application.

To design a GUI application we require GUI components such as label, check box, Button, text-filed user interfaces. In java programming all these GUI components are given in the form of pre-defined class creating a GUI component is nothing but creating an object of appropriate pre-defined class.

Example:

Create a button with a name server for an GUI

Button B1=new button (“save”);

Checkbox C1=check box (“C”)

Here you can see the button B1 is the appearance of the save button and Check box C1 is the function of “c”.

Example Program on java.awt.*;

package java.awt;

```
import java.awt.peer.CheckboxPeer;

import java.awt.event.*;

import java.util.EventListener;

import java.io.ObjectOutputStream;

import java.io.ObjectInputStream;

import java.io.IOException;

import javax.accessibility.*;

public class Checkbox extends Component implements ItemSelectable, Accessible

{

        static

        {

                */////////////////////////////*

        }

        }
```

3) Java.awt.event.*:

Here event is the sub package of awt package. The package is to provide functionality to the GUI application.

Define of event.

Change in the state of object is known as an event. In order to develop a complete GUI application we must import java,awt.*; and java.awt.event.*; we

can say to create an GUI application both “function” and “design” are important. Here java.awt.*; is the design and java.awt.event.*; is the function.

Example Program on java.awt.event.*;

```
package java.awt;

import java.awt.event.*;

import java.io.*;

public class Event implements java.io.Serializable
{

//*************************//
}
```

4) Java.io.*;

This is also called as file programming or stream-handling package. The purpose of this package is to achieve the data persistence which will be storing the data permanently.

The flow of data between main memory and secondary memory in the bytes/bits is known as **stream.**

Example program on java.io.*:

```
package java.io;

public class FileNotFoundException extends IOException {

        private static final long serialVersionUID =
-8978569738237104 92L;
```

```
public FileNotFoundException() {
super();
{
    Public FileNotFoundException(String s) {
    super (s);
    {
        Private FileNotFoundException(String path, String reason) {
        super (path + ((reason == null)
    }
}
```

5) Java.applet.*;

This package is used for applet programming. The purpose of this package is to develop disturbed applications. We know that distributed applications will be run in the context of browser/www and whose result is sharable across the globe.

In the initial data of java development “Sun Micro Systems” developers develop a concept called applets for development of distributed applicatons. To full fill the concepts of applets, we have pre-defined class called **applet,** it presents in package called java.applet.*; this package contains only one class who’s fully qualified name is **java.applet.applet**.

Definition of applet:

An applet is a java program that runs in the context of browser/www and who's result are sharable accessible across the globe.

Example Program on Java.applet.*;

```
package java.applet;

import java.awt.Image;

import java.awt.Graphics;

import java.awt.image.ColorModel;

import java.net.URL;

import java.util.Enumeration;

import java.io.InputStream;

import java.io.IOException;

import java.util.Iterator;

public interface AppletContext

{

        AudioClip getAudioClip(URL url);

        {

            *//////////////////////////////////////*

        }

    }
```

1) Java.net.*;

This package is also called as network programming. The purpose of this package is to develop client-side application.

In client server application development we have two applications. They are

a) Client-side applications

b) Server-side applications

Network programming

For development of client-side application and server-side application, we have pre-defined classes and interfaces which are present in java.net.*:

Example program on java.net.*;

```
package java.net;

import java.io.FileDescriptor;

import java.io.IOException;

import java.io.InterruptedIOException;

import java.util.Enumeration;

abstract class AbstractPlainDatagramSocketImpl extends DatagramSocketImpl

{

    *////////////////////////////////////

    ////////////////////////////////////////*
```

```
}
```

7) java .util.*;

This package is also called as collection framework. The purpose of this package is to provide higher performance to the java applications.

It is one of the standardized mechanisms for grouping multiple values either of same type or defined type or both the types in a single variable with dynamic size in a single variable. This single variable is known as collection frame work variable.

Example Program on java.util.*;

```
package java.util;

public abstract class AbstractCollection<E> implements Collection<E>

{

   protected AbstractCollection()

{

        public abstract Iterator<E> iterator();

        public abstract int size();

        }

/*//////////////////////////////////////////*

}
```

8) java.text.*;

The main aim of this package is for formatting. We can use this package for

- Formatting of dates
- Formatting of times
- Manipulation of numerical operations etc.,

Mainly these packages are used for “Report generation pages”.

Example program on Java.text.*;

```
package java.text;
public class Annotation
{
    public Annotation(Object value)
{
        this.value = value;
        }
        public Object getValue()
{
                return value;
        }
public String toString()
{
                return getClass().getName() + "[value=" + value + "]";
        }
```

private Object value;

}

What is meant by fully classified name?

For example, for calling a class or to run the program developer should provide an full class name in the IDE. If we are writing a program on ***lang*** with string we provide a class package called java.lang.string which is called as fully classified package.

In java point of language the **lang** is one of the sub-package and string is one of the class.

Operating view:

Here, java is the package folder and lang is the sub package. In that sub package string is the class file.

Diagram View:

String.class

Guidelines for creating user defined packages:

Syntax for creating package:

Package pack1[pack2[.....[pack.n]]]

In above syntax package is a keyword used for developing user-defined packages. Here pack1,pack2,packn represents java valid variable names treated as user-defined package names. Pack2....Pack n represents the names of sub packages of the structure.

Example:

Package p1;

Or

Package p1, p2;

Guidelines:

1) Choose an appropriate package for placing common classes and interfaces and ensure that the package statement must be the first executable statement of the program.

2) Choose an appropriate class name/interface name and ensure whose modifier must be public.

3) The modifier of the constructor of the class of the package must be public. This step is not acceptable for concept of interfaces.

4) The modifier of the method of the class/interface which is present in the package must be public.

5) Whichever class or interface present in the package must be given a title name with an extension .java

6) At any point of time it is recommend to the java programmer either to define a class in package

(Or)

It is not recommended to define both interface and package definition in the window because we get file recognition problem.

Programs

Creating a package test display (tp) and place a class called work.

Program:

```
//Work.java----------Rule 5
Package tp;----------Rule 1
Public class test ----------Rule 2
{
        Public work ()----------Rule 3
        {
                System.Out.Println (“work-----Dc”);                -----〈Rule 6
        }
        Public void disp ()----------Rule 4
        {
                System.Out.Println (“Test----------Display”);
        }
}///Work
```

Create a package tp and place an interface 2test

Program:

```
//2test.java
Package tp;
Public interface 2test
{
```

```
        Void show ();
}
```

Example program to put an interface in the package animals:

```
/* File name : Birds.java */
package birds;
interface birds
 {
          public void eat();
         public void travel();
}
```

Now, put an implementation in the same package animals:

```
package birds;
/* File name : FlyingInt.java */
public class FlyingInt implements Birds
{
public void eat()
{
              System.out.println("Flying birds eats");
        }
```

```
public void travel()
{
    System.out.println("Flying birds travels");
        }
        public int noOfWings()
{
         return 0;
         }
public static void main(String args[])
{
                FlyingInt f= new FlyingInt();
                f.eat();
                f.travel();
        }
}
```

Now, you compile these two files and put them in a sub-directory called “Birds" and try to run as follows:

Flying birds eats

Flying birds travels