Java Packages

→ Packages is collection of java classes, interface or other packages

⇒ If you have large number of classes, interface you can arrange them together and form a package

* there exist .class for each class and interface, so java classes or interfaces lies in .class file

.Java → (Java Compiler) → .class files

→ Package is nothing but a folder and you store .class files there in the package

⇒ Java can have usedefined packages along with predefined Packages (Built in ~~classes~~ packages) which contain lot of predefined classes and interfaces, & .Java file

Ex:) we `import java.lang.*;` from java.lang. ⊕

`java.lang` ↑ Package

⊕ ↑ Import everything

⇒ you have lib file folder and in src.zip you have list of all packages

⇒ A package contains .class, .Java file which can be very much helpfull

Package is a folder subpackages is subfolder

⇒ A package can have subpackages (subfolder) inside each another.

keyword to import the package

⊕ `import java.util.scanner;` → $.class1.Java file inside Package

java ↑ Package, util ↑ subpackage of java

⇒ with use of import we can create & use previously built code & use them.

⇒ Packages will save lot of time as we can directly import previously written code with ease

⇒ Also with * you can import all files from package or you can get a specific file also Ex: import java.util.scanner;

⊛ you can import files at start with import

import java.lang.string; or without importing at start

string z = new string(); Java.lang.string z = new java.lang.string();

Int A = new Int[5];

⇒ Ex: Arrays.sort(A)

But importing at start will save your efforts as you dont have to write long code each time.

⊛ So As we learned about how can we use predefined package but we can also will able to create own packages.

Creating Own packages

So to create a java package it is very easy thing so for your info you have create a java file

~~myclass.c~~

Hell.java

```
Package NamethePackagYouWantToCreate;
        Package of this name will created

Public Hell {
        public void function() {
                S.O.P("Hello world");
        }
}
```

This will create .class file ~~Hell~~ Hell.java and it will get stored inside name~~of~~Packagename & it contains Hell.class file

⇒ the public Hell is main class here and if you compile normally you will get .class file But if you do

Javac -d locationat which you want Javafile;
Package

Just name the package in which you want to create Package

Package Name; in Java file

and compile with -d flag and location & that file & you will get Package

Java -d ./ filename
location

⇒ you can add multiple class inside same package

Also the class you want to insert in package should not have main class

Public main class{ → Access specifier also has measure role while accessing through a Package

}

You can also will able to create subpackage by specify subpackage name in top of java file

hellclass.java

Package mainfolder.subpackage;

public hellclass{

}

and when you do

Javac -d ./ hellclass.java

⇒ a subpackage will create inside mainfolder package and it is named as subpackage (name you specified)

⇒ After creation of .class file ~~you~~ in package you can delete .java file still you can import the functionality through package (import)

⇒ Package are really helpfull in many case when you need lot of work to do.

with Package name; command at start and compile file with

javac -d [location] filename

you can easily able to create packages & subpackage.

⇒ Another time note that main class should not have main() function if you want to add it in package

⇒ you can add any numbe of ~~Java fi~~ :class files inside a Java package

if you use

import package.nameclass.youcreated;

and you can access code, functions from it.

Hell.java

Ex:

```
Package ghansham;
//main class
Public Hell {

public void hello(){
  S.o.p("Hello from
        ghansham");

  }

}
```

// compile & create package

Javac -d ./ Hell.java

If create package called ghansham & inside it you have Hell.class

now import

another.java

```
import ghansham.Hell;
//another class to test package
Public another{

main() {

ghansham.Hello();
}
```

//Output

Hello from ghansham

Such like that you can create and access a package or subpackage

⇒ ~~naming convechions~~

#naming conventions for packages

If you have lot of classes so & you have to sum up all package and give name

Ex: Roll no. collegeid, marks subject

⇒ Packet student

such like that give domain name

→ you have avoid same classname conflict

say your url is www.university.com

& In university packet you have classes

student
account
Book
& Registration

⇒ you show name package like

com.university.student

or com.university.Book

Basically in reverse order that of url