Java Packages

What is a package ?

- Collection of functionally similar classes & interfaces , grouped together.

Need ?

1. To group functionally similar classes together.
2. Avoids name space collision (allows duplicate class names in different packages)
3. Finer control over access specifiers.

Steps

1. To create class A under package “p1” , for simplicity
   - Create folder <p1> -- under <src>
     (Package name must match with the folder name)
2. Add package statement as the 1st statement in Java source file.
   - Eg : package p1; (the class/classes in this src file will be part of package p1)
3. To compile , from command prompt , from <src> folder
   - `javac -d ..\bin p1\A.java`
4. Since Package names are mapped to folder names,
   Java compiler will automatically create the sub-folder <p1> under the <bin> folder & place A.class within <p1>

5. **NOTE** : It’s not **mandatory** to create java sources(.java) under package named folder. BUT its mandatory to store packged compiled classes(.class) under package named folders
   - Earlier half is just maintained for convenience , so that java compile can then automatically detect the dependencies & compile classes.

6. How to launch / run packaged java classes?
   cd ..\<bin>
   java FullyQualifiedClassName containing main method
   Eg. java p1.A

7. To run the packaged java classes from any folder
   - You will need to set the classpath

- Eg java –cp p1.A
  OR
  You can set classpath | CLASSPATH as Java specific environment variable.
  It is used mainly by JVM's classloader : to locate & load the classes.
  Classloader will try to locate the classes from classpath entries , to resolve & load Java classes.

8. What should be value of classpath ?
   - It must be set to top of packaged class hierarchy(eg : bin folder)
     set classpath=d:\dac\day2\bin;.;
     Using command line invocation OR set it from environment variables.

Package related Rules

1. If the class is part of a package, the package statement must be the first line in the source code file, before any of import statements or class declaration.

2. import and package statements apply to all classes within a source code file.
In other words, there's no way to declare multiple classes in a file and have
them in different packages, or use different imports.

2. Access specifiers in detail
- Refer to a diagram : Access specifiers.png
- Refer to a diagram : understanding access specifiers.png
- Set it up & confirm the table.

3. Java Arrays
An array is a **collection** of **similar** types of data.
It's a **data structure**, which is **fixed** in size.
Eg. if we want to store the names of 100 people then we can create an array of the string type that can store 100 names.

String[] array = new String[100];
Here, the above array cannot store more than 100 names. The number of values in a Java array is always fixed.

In Java, arrays are created as **objects**.
Like objects, arrays are dynamically created & stored on the **heap**.
Before creating an array object , JVM will load the array class in the method area.

How to **create arrays** in Java?

- dataType[] arrayName;
- dataType - it can be primitive data types like int, char, double, byte, etc. or reference type
- arrayName - it is an identifier
- Eg .   double[] data;//memory is   allocated for array type of referecne variable .
  Here, data is an array that can hold values of type double.
- To define the number of elements that an array can hold, we have to allocate memory for the array in Java.
- Allocate memory n create array object use new keyword
  Eg. data = new double[10];
  Here, the array can store 10 elements. We can also say that the size or length of the array is 10.
- To declare and allocate the memory of an array in one single statement
  Eg. double[] data = new double[10];
  Array object has length as the data member , to get the size of the array.
- To Initialize Arrays in Java , during declaration ?
  Eg. int[] ages {12, 4, 5, 2, 5};
- You can use for loop as well as for each loop to iterate over an array.

for-each loop(enhanced for loop)

- **Syntax**

for (dataType varName : array)

{ body of for each loop }

eg : To display array data using for-each loop

```
for(double d : data) //implicit assignment  : d=data[0],d=data[1]......d=data[data.length-1]
 System.out.println(d);
```

for-each loop limitations

1. It can only iterate from 1st element to last element , with step size +1

2. It works on a copy of array elements

(i.e. it can't be used for modifying array elements).

- You can also create **Array of references** in Java , where references are stored in an array and actual objects are stored elsewhere in the heap.

Java Multidimensional Arrays

A multidimensional array is an array of arrays. Each element of a multidimensional array is an array itself. For example,

```
int[][] a = new int[3][4];
```

Here, we have created a multidimensional array named a. It is a 2-dimensional array, that can hold a maximum of 12 elements.

4. static keyword in java

Usages

4.1 static data member

- Memory is allocated only once at the class loading time. (also known as class variables)

- Its metadata (i.e data type , access specifier ,name etc.) is stored in metaspace along with the loaded class information but it’s value is stored in special area of heap (but not along with the instances)

- one single copy of static data member is shared across all objects of the same class.

- It is initialized to the default values(eg --double --0.0,char -0, boolean -false,reference-null)

- How to refer ? -- className.memberName

eg -- class Emp {

public static int idCounter;

private String name;

private int empId;

...

}

If we hire 10 emps (i.e 10 emp objs created in the heap)

How many copies of

idCounter : 1

name : 10

empId : 10

4.2 static methods

- Can be accessed without creating instance. (ClassName.methodName(....))
- You Can't access 'this' or 'super' keywords from within static method.

Note :

1. Can static methods access other static members directly(without creating instance) : YES

2. Can static methods access other non-static members directly(without creating instance) : NO

3. Can non static methods access static members directly ? -- YES

4.3 . static import

- Using this syntax, you can directly access static members from the specified class, without the class name.

eg .

To access directly , ALL static members of the System class

```
import static java.lang.System.*;

public static void main(String[] args)

{

    out.println(“testing static import”);

    Scanner sc=new Scanner(in);

    gc();

    exit(0);

    sc.close();

}
```

4.4 static initializer block

syntax -

static {

1. It gets called only once at the class loading time , by JVM's classloader
2. Usage
   - to initialize all static data members
   - to achieve singleton pattern , loading DB drivers , loading frameworks (for the functionality -which HAS to be called precisely once.

3. A class can have multiple static initilaizer blocks(legal BUT not recommended)

4. It appears , within a class definition & can access only static members directly

Regarding non-static initializer blocks(instance initializer block)

syntax

{

1. It will be called once per instantiation, before matching constructor
2. Better alternative is to use a parameterized constructor.

}

5. **static nested classes** ---

In Java , you can create a statically nested class within an outer class.

eg --

```
class Outer {

// can contain static & non-static members

   static class Nested

   {

       //can access ONLY static members of the outer class
DIRECTLY(without creating instance)

   }

}
```