19-August-2017

NESTED PACKAGE → Package Within Package

package p1.sub; → If package p1 is available then (child i.e. Sub) will be created

→ If parent 'p1' is not there then it will create 'p1' first then child 'sub'.

Example:-

D:\f1>

Temp1.java

```
package p1.sub.sub1

public class Temp1
{
public void show()
{
Sop("Temp1");
}
}
```

```
package p2;
import p1.*
↳ Using this gives error
import p1.sub.sub1.*;

class Temp2
{
psvm()
{
new Temp1().show();
}
}
```

✓ Child packages must be imported explicitly.

ACCESS MODIFIERS

- Only present in OOPS based language.
  Not in Procedure Programming Language.
- Access Modifiers defines the scope of any class and their parts.
- Works only at package level.

Imp.
Java Rule:

public = protected = default = within same package.

Example:-

D:\f1
p2 → Temp1

```
public class Temp
{
Temp()
{
}
public void show()
{
Sop("show");
}
```

```
package p2
import p1.*;
class Temp2 extends Temp
{
psvm()
{
new Temp1().show();
Temp1 t = null;
}
}
```

Can't work as constructor is not public

This will work as here no object is created here 't' is just reference variable.

default

- This 'default' is different from 'default' of interfaces
- Can only be used within a package.

RULE
A class can't be protected

Data f^n's, Member & Constructors can be protected

Protected.

- protected things of a class can be used outside the package only via inheritance, association is not allowed.

Examples:-

D:\ f1>
p1 → Temp1

```
public class Temp1
{
 public Temp()
 { }
protected void show()
 {
Sop("pkg p1");
}}
```

E:\ f2>
p2 → Temp2

```
par class Temp2
{
  psvm
 {
 new Temp1().show();
 }
}
```

- Error, as 'show()' fn is protected here.

Example:-

D:\f1>
p1 → Temp1

Exactly as Above.

E:\f2>

```
package p2;
import p1.*;

Class Temp2 extends Temp1
{
psvm
{
new Temp2().show()
}
}
```

↳ This will execute correctly as here we have created object of Temp2, Obj. of Temp1 is not allowed here also.

Example :-

D:\f1>
p1 → Temp1

```
protected Temp1()
{
}
public void show()
{
 Sop ("pkg p1");
}
```

Private

Rule

1. Normal class can't be private but Nested class can be private.

2. In OOPS, Private Things of a class can't be used outside the class.
But in JAVA, this is is partially true.

Rule

If you make the constructor of any class private, then you can't create object of this class, outside the class.
This is also Partially True in JAVA.