21-July-2017

# CONDITIONS BASED ON PACKAGES

**Condition-1**: Two different folders having a same package with the same class.

**Example**

D:\f1>
p1 → Temp1
public void show()
{Sop ("package p1 from f1");}

D:\f3>
p1 → Temp1
public void show()
{Sop ("package p1 from f3");}

E:\f2>
Package p2;
import p1.*;

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

**Steps**

(i) Compiling Temp1 in both 'f1' & 'f3' folder.

(ii) (a) E:\f2 > set classpath= D:\f1 ; D:\f3;
(b) E:\f2 > javac -d . * . java E:\f2>java p2.Temp2
Output : package p1 from f1.

(iii) (a) E:\f2 > set classpath = D:\f3 ; D:\f1
E:\f2 > java ____ p2. Temp2
Output : package p1 from f3

→ No need to compile again as we just linked diff. classpath

Imp: Compiler will link to that p1 which is written first in while setting classpath.

## Condition-2: Two different folders having a same package with different classes

Example:-

D:\f1>

p1 → Temp1

```
public void show()
{SOp
SOp("pkg p1 from f1");
}
```

D:\f3>

p1 → Temp3

```
public void show()
{
Sop("pkg p1 from f3")
}
```

E:\f2>

```
package p2;

import p1.*;

Class Temp2
{
 psvm()
{
new Temp1().show();
new Temp3().show();
 }
}
```

STEPS

(i) Compile (a) Temp 1 in 'f1'
(b) Temp3 in 'f3'

(ii) (a) E:\f2 > set classpath = D:\f1 ; D:\f3;

(b) E:\f2 > javac -d . *.java.

(c) E:\f2 > java p2.Temp2

Output: pkg p1 from f1
pkg p1 from f3

IMP So, compiler linked to both the p1 & displayed output.

## Condition : 3 – Two different folders having different packages with a same class.

D:\f1>
p1 → Temp1
public void show()
{
Sop("Temp1 of pkg1 from f1");
}

D:\f3>
p3 → Temp1
public void show()
{
Sop("Temp1 of pkg3 from f3");
}

E:\f2>
package p2;

import p3.*;
import p1.*;

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

### Steps

(i) Compile both 'Temp1' in 'f1' & 'f3'

(ii) Compile 'Temp2' in 'E:\f2' by setting classpath.

Output: This program will give Ambiguity error

- This case also arises in Java itself as, 'util' & 'sql' have same 'Date' class
  ↳ packages

Imp. So, to remove error, we access a class using package name.

P.T.O.

Now, Rewriting

E:\f2

```
package p2;

import p1.*;
import p3.*;

class Temp2
{
psvm()
{
new p1.Temp1().show()
```

Output: Temp1 of pkg1 from f1

OR `new p3.Temp1().show()`

Output: Temp1 of pkgB from fB

```
}
}
```

Evergreen
Page No.
Date: / /201

## Condition: 4 — Using the class of one package into the class of default package

• P → To → D

### Rule

Never keep any java file parallel to any package because while using a package in default package we ~~give~~ a .java file, default package will search entire folder

### Example:

```
import p1.*;
```
↳ To use/keep parallel java file we have to write exact class name like 'p1.Temp1'

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

### Steps

(i) Create & Compile 'Temp1' of 'p1' first.

(ii) (a) Set Classpath

(b) Compile Temp2.

(iii) Run Temp2 of 'default' package.

Output :- ~~pack~~ pkg p1

## Condition 5 :- One class using another Class which is parallel to it and it is getting same class from another package

D:\f1>

p1 → Temp1

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

E:\f2>

Temp1.java

```
package p2;

Class Temp1
{
void show()
{
Sop("Temp1 from same pkg ie p2");
}
}
```

Here we didn't need to make class and 'show()' f^n public as we are using in same package.

Temp2.java

```
package p2;
import p1.*;
class Temp2
{
psvm();
{
new Temp1().show();
```

Output: Temp1 of pkg p2

OR

```
new p1.Temp1().show();
```

Output: Temp1 of p1

**Rule:**

Parallel class will be given priority.

- Whenever you encounter duplicacy error, access class by package name.

**Steps**

(i) Compile Temp1 in p1
(ii) Compile Temp1 in E:\f2
(iii)(a) Compile Temp2 in E:\f2
(b) Run Temp in p2

## Condition-6:- Using class of one package into class of another package without setting classpath.

D:/f1>
p1 → Temp1

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

E:/f2>
p2 → Temp2

```
package p2;
import p1.*;

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

### Steps

(i) E:/f2> javac -d . *.java
This gives error.

(ii) (a) E:/f2> javac -classpath D:/f1; -d . *.java
This will compile successfully.

(b) E:/f2> java -classpath D:/f1; p2.Temp2
Output: pkg p1

### FROM JDK 6

(a) E:/f2> javac -cp D:/f1; -d . *.java
(b) E:/f2> java -cp D:/f1; p2.java

↳ Shortform of '-classpath'

## Condition-7 :→ Setting classpath during Runtime that means via java program

- This is usually done by JVM while importing packages i.e ~~not jar~~ that are pre-build like 'util', sql()

### Example:

D:\f1>
p1 → Temp1

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

E:\f2 > Temp2.java

```
Class SetClassPath
{
psvm()
{
String cp=System.getProperty
        ("java.class.path");
Sop(cp);

System.setProperty("java.
        class.path",
        "D:\\f1");

String cp1=System.getProperty
        ("java.class.path");

System.out.println(cp1);
}
}
```

Imp:

- ~~Use~~ This method is used when we have to load class dynamically.
- Classpath is set as long as program runs.
- Changes back to original when program ~~run~~ ends.

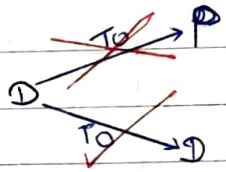
This program is used / classpath is set dynamically when we have to load classes dynamically.

Evergreen
Page No.
Date : / /201

## Condition-8 :- Using the class of 'default' package outside that 'default' package.

### Rule

The 'default' package can only be used at other location rather than its original location if ~~its~~ they are imported only to 'default' package.




### Example :-

D:\f1 > Temp1.java

```
public void show()
{
Sop("package p1");
}
```

E:\f2 > Temp2.java

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

### Steps

(i) Compile Temp1.java.

(ii) E:\f2 > set classpath = D:\f1;

(iii) E:\f2 > javac Temp2.java
java Temp2