Packages ⇒

⇒ Naming & visibility mechanism of Java.

⇒ In Java packages are implemented through folder, but folder doesn't mean it will always be a package.

```
Package P1;
Class A{
    PSVM(--){
        SOPC("Hello");
    }
}
```

x.java ⇓ Let's say it is Stored in {documents/Java}

In this scenerio we need to make P1 and manully shift A.Class to that folder.

—OR→

```
--/documents/Java >javac -d . x.java
```

-d → Cereate a folder

. → In current folder.

{This will create a P1 folder automatically and A.class will be shifted to P1 automatically}

Mac

```
--/documents/Java > jovac -d user/downloads/Testing x.java.
```

Win

```
C:/ Test > javac -d e:/xyz x.java
```

After creating P1 using -d in any folder

```
C:/test > java A      } error
documents/Java > java A      } error
```

If we have used (-d .)

```
C:/ Test > java P1.A
```

```
C:/ Test > javac -d e:/xyz x.java

C:/ Test > set classpath = %Classpath%; e:/xyz
                ↓
C:/Test > java P1.A
```

Let's say we want to ~~create~~ a transfer packages, for that we want to create zip file, for that in java we have java archieve — jar file

```
C:/ Test > jar -cf myjar.jar *
```

* → All files

myjar.jar ↓ {In myjar.jar file}

In different PC ↘

Lets we put that into e:/xyz

```
d:/abc > set classpath = %Classpath%; e:/xyz/myjar.jar
d:/abc > java myclass.
```

Visibility ⇒

	Private	No modifier /default	Protected	Public
Same Class	✓	✓	✓	✓
same package	✗	✓	✓	✓
Same package subclass	✗	✓	✓	✓
different Package subclass	✗	✗	✓	✓
Different package Non SubClass	✗	✗	✗	✓

22/3/24

Import ⇒

import java.lang.*;

⇒ All public classes of java.lang can be used.

import java.utils.Scanner;

⇒ Only Scanner class of java.util.

Static Import ⇒

⇒ All athe packages of java are known as subpackage of either java or javax.

import java.sql.*;

All classes.

Package Subpackage

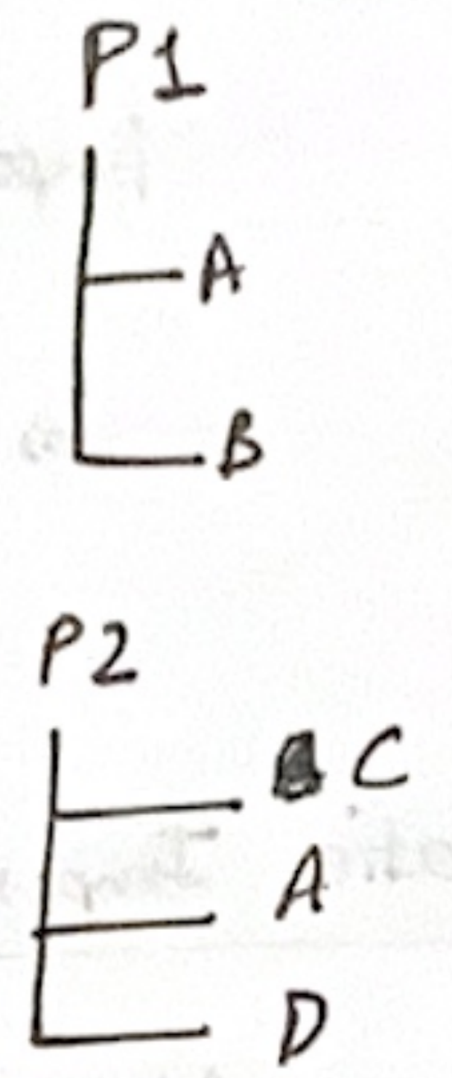
We can do this as well.

package P1.P2.P3;

* importing with * form, may increase a very small compile time, but it will not affect the runtime performance of java programs.

- you can import more than one packages containing classes with same name and it will not produce any error until we use a class which is there in more than ~~a~~ one packages which are imported with star form.

```
import P1.*;
import P2.*;
!
B b; ✓
D d; ✓
A a1; ✗ error
!
!
P2.A a1 = new P2.A(); ✓
```



{we have to use fully qualified name.}

ex=>

```
java.util.Scanner sc = new java.util.Scanner();
```

↳ By this we can avoid import statements.

```
Package P1;
Class A {
  -
  -
  -
}
```

A.Java

```
Package P2;
  import P1.*;
Class B{
  PSVM(-){
  A a1 = new A();
  }
}
```

↳ error as A is default class.

{We would need to make A public to access it in B.}

```
Package P1;
Public Class A{
    int i=1;
    Private int j=2;
    Protected int K=3;
    Public int m =4;
  }
}
```

```
Package P2;
import P1.*;
Class B extends A {
  PSVM(-){
    SOP(i); ✗
    SOP(j)  ✗
    SOP(k)  ✗
    SOP(m); ✗
  }
}
```

non-static to static access.

```
Package P2;
import P1.*;
Class B extends A{
  Void m1(){
    SOP(i). ✗
    SOP(j); ✗
    SOP(k); ✓
    SOP(m); ✓
  }
}
```

```
Package P2;
import P1.*;
Class B {
  PSVM(-){
  A a1 = new A();
  SOP(a1.i)  ✗
  SOP(a1.j); ✗
  SOP(a1.k); ✗
  SOP(a1.m); ✓
  }
}
```